Rational Rose 2000e Rose Extensibility Reference

Contents

List of Figures xxxv

List of Tables xxxvii

Preface xliii

Prerequisites xliii

How This Manual Is Organized xliv

Online Help xlv

Online Manual xlv

Related Documentation xlv

File Names xlv

Chapter 1 Rational Rose Extensibility Interface Reference 1

AbstractState Class 1
- AbstractState Class Properties 2
  - AbstractState.SubActivities Property 2
  - AbstractState.SubDecisions Property 3
  - AbstractState.SubStates Property 3
  - AbstractState.SubSynchronizations Property 4
- AbstractState Class Methods 5
  - AbstractState.AddDoAction Method 6
  - AbstractState.AddEntryAction Method 6
  - AbstractState.AddExitAction Method 7

AbstractState.AddStateMachine Method 7
AbstractState.AddUserDefinedEvent Method 8
AbstractState.DeleteAction Method 8
AbstractState.DeleteStateMachine Method 9
AbstractState.DeleteUserDefinedEvent Method 9
AbstractState.GetActions Method 10
AbstractState.GetAllSubActivities Method 10
AbstractState.GetAllSubDecisions Method 11
AbstractState.GetAllSubStates Method 11
AbstractState.GetAllSubSynchronizations Method 12
AbstractState.GetDoActions Method 12
AbstractState.GetEntryActions Method 13
AbstractState.GetExitActions Method 13
AbstractState.GetStateMachines Method 14
AbstractState.GetUserDefinedEvents Method 14

Action Class 15
Action Class Properties 15
Action.Arguments Property 15
Action.Target Property 16
Action Class Methods 16

Activity Class 17
Activity Class Properties 17
Activity Class Methods 18

ActivityView Class 18
ActivityView Class Properties 18
ActivityView Class Methods 19
ActivityView.GetActivity Method 19

Application Class 20
Application Class Properties 20
Application.AddInManager Property 21
Application.ApplicationPath Property 22
Application.CommandLine Property 22
Application.CurrentModel Property 23
Application.Height Property 23
Application.IsInitialized Property 23
Application.Left Property 24
Application.PathMap Property 24
Application.ProductName Property 25

Application.Top Property 25
Application.Version Property 26
Application.Visible Property 26
Application.Width Property 27
Application Class Methods 27
Application.CompileScriptFile Method 29
Application.ExecuteScript Method 30
Application.Exit Method 30
Application.FreeScript Method 31
Application.GetLicensedApplication Method 32
Application.GetObject Method 32
Application.GetProfileString Method 33
Application.GetRoseIniPath Method 34
Application.LoadScript Method 35
Application.NewModel Method 36
Application.NewScript Method 36
Application.OpenExternalDocument Method 37
Application.OpenModel Method 38
Application.OpenModelAsTemplate Method 39
Application.OpenRoseModel Method 40
Application.OpenScript Method 41
Application.OpenURL Method 41
Application.Save Method 42
Application.SaveAs Method 43
Application.WriteErrorLog Method 43
Application.WriteProfileString Method 44

Association Class 45
Association Class Properties 45
Association.Constraints Property 46
Association.Derived Property 46
Association.LinkClass Property 46
Association.ParentCategory Property 47
Association.Role1 Property 47
Association.Role2 Property 48
Association.Roles Property 48
Association Class Methods 49
Association.ClearRoleForNameDirection Method 49
Association.GetCorrespondingRole Method 50
Association.GetOtherRole Method 50
Association.GetRoleForNameDirection Method 51

Association.NameIsDirectional Method 51
Association.SetLinkClassName Method 52
Association.SetRoleForNameDirection Method 52

Attribute Class 53
Attribute Class Properties 53
Attribute.Containment Property 54
Attribute.Derived Property 54
Attribute.ExportControl Property 55
Attribute.InitValue Property 56
Attribute.ParentClass Property 56
Attribute.Static Property 56
Attribute.Type Property 57
Attribute Class Methods 57

Category Class 58
Category Class Properties 58
Category.Associations Property 59
Category.Categories Property 59
Category.ClassDiagrams Property 60
Category.Classes Property 60
Category.Global Property 61
Category.ParentCategory Property 61
Category.ScenarioDiagrams Property 62
Category.UseCases Property 62
Category Class Methods 63
Category.AddCategory Method 65
Category.AddCategoryDependency Method 65
Category.AddClass Method 66
Category.AddClassDiagram Method 66
Category.AddDataModelDiagram Method 67
Category.AddScenarioDiagram Method 68
Category.AddUseCase Method 68
Category.DeleteCategory Method 69
Category.DeleteCategoryDependency Method 69
Category.DeleteClass Method 70
Category.DeleteClassDiagram Method 70
Category.DeleteScenarioDiagram Method 71
Category.DeleteUseCase Method 71
Category.GetAllCategories Method 72
Category.GetAllClasses Method 72
Category.GetAllUseCases Method 73

Category.GetAssignedSubsystem Method 73
Category.GetCategoryDependencies Method 74
Category.HasAssignedSubsystem Method 74
Category.RelocateAssociation Method 75
Category.RelocateCategory Method 75
Category.RelocateClass Method 76
Category.RelocateClassDiagram Method 76
Category.RelocateScenarioDiagram Method 77
Category.SetAssignedSubsystem Method 77
Category.TopLevel Method 78

CategoryDependency Class 78
CategoryDependency Class Properties 78
CategoryDependency Class Methods 79
CategoryDependency.GetContextCategory Method 79
CategoryDependency.GetSupplierCategory Method 80

Class Class 80
Class Class Properties 81
Class.Abstract Property 82
Class.Attributes Property 82
Class.Cardinality Property 82
Class.ClassKind Property 83
Class.Concurrency Property 84
Class.ExportControl Property 85
Class.FundamentalType Property 86
Class.Operations Property 86
Class.Parameters Property 87
Class.ParentCategory Property 87
Class.ParentClass Property 88
Class.Persistence Property 88
Class.Space Property 88
Class.StateMachine Property 89
Class Class Methods 89
Class.AddAssignedModule Method 92
Class.AddAssociation Method 92
Class.AddAttribute Method 93
Class.AddClassDependency Method 93
Class.AddHas Method 94
Class.AddInheritRel Method 94
Class.AddInstantiateRel Method 95
Class.AddNestedClass Method 95

Class.AddOperation Method 96
Class.AddParameter Method 97
Class.AddRealizeRel Method 99
Class.CreateStateMachine Method 99
Class.DeleteAssociation Method 100
Class.DeleteAttribute Method 100
Class.DeleteClassDependency Method 101
Class.DeleteHas Method 101
Class.DeleteInheritRel Method 102
Class.DeleteInstantiateRel Method 102
Class.DeleteNestedClass Method 103
Class.DeleteOperation Method 103
Class.DeleteParameter Method 104
Class.DeleteRealizeRel Method 105
Class.DeleteStateMachine Method 105
Class.GetAllNestedClasses Method 106
Class.GetAssignedLanguage Method 106
Class.GetAssignedModules Method 107
Class.GetAssociateRoles Method 107
Class.GetAssociations Method 110
Class.GetClassDependencies Method 111
Class.GetClients Method 112
Class.GetHasRelations Method 113
Class.GetInheritRelations Method 113
Class.GetInstantiateRelations Method 114
Class.GetLinkAssociation Method 114
Class.GetNestedClasses Method 115
Class.GetRealizeRelations Method 115
Class.GetRoles Method 116
Class.GetSubClasses Method 116
Class.GetSuperclasses Method 117
Class.IsALinkClass Method 117
Class.IsNestedClass Method 118
Class.RemoveAssignedModule Method 118

ClassDependency Class 119
ClassDependency Class Properties 119
ClassDependency.ClientCardinality Property 119
ClassDependency.ExportControl Property 120
ClassDependency.InvolvesFriendship Property 121
ClassDependency.SupplierCardinality Property 121
ClassDependency Class Methods 121

ClassDiagram Class 122
ClassDiagram Class Properties 122
ClassDiagram.ParentCategory Property 122
ClassDiagram Class Methods 123
ClassDiagram.AddAssociation Method 124
ClassDiagram.AddCategory Method 124
ClassDiagram.AddClass Method 125
ClassDiagram.AddUseCase Method 125
ClassDiagram.GetAssociations Method 126
ClassDiagram.GetCategories Method 126
ClassDiagram.GetClasses Method 127
ClassDiagram.GetClassView Method 127
ClassDiagram.GetObject Method 128
ClassDiagram.GetSelectedCategories Method 128
ClassDiagram.GetSelectedClasses Method 129
ClassDiagram.GetUseCases Method 129
ClassDiagram.IsDataModelingDiagram Method 130
ClassDiagram.IsUseCaseDiagram Method 130
ClassDiagram.RemoveAssociation Method 131
ClassDiagram.RemoveCategory Method 131
ClassDiagram.RemoveClass Method 132
ClassDiagram.RemoveUseCase Method 132

ClassRelation Class 133
ClassRelation Class Properties 133
ClassRelation Class Methods 133
ClassRelation.GetContextClass Method 134
ClassRelation.GetSupplierClass Method 134

ClassView Class 135
ClassView Class Properties 135
ClassView.AutomaticResize Property 136
ClassView.ShowAllAttributes Property 136
ClassView.ShowAllOperations Property 137
ClassView.ShowOperationsSignature Property 137
ClassView.SuppressAttributes Property 138
ClassView.SuppressOperations Property 138
ClassView.Visibility Property 139
ClassView Class Methods 139
ClassView. GetDisplayedAttributes Method 140
ClassView.GetDisplayedOperations Method 141

ClientRelKind Enumeration 142

ClientRelType Enumeration 143

Collection Classes 144
- Collection Class Properties 145
  - Collection.Count Property 145
- Methods for All Collections 146
  - Collection.Exists Method 147
  - Collection.FindFirst Method 147
  - Collection.FindNext Method 148
  - Collection.GetAt Method 149
  - Collection.GetFirst Method 149
  - Collection.GetObject Method 150
  - Collection.GetWithUniqueID Method 150
  - Collection.IndexOf Method 151
- Methods for User-Defined Collections 151
  - Collection.Add Method 152
  - Collection.AddCollection Method 152
  - Collection.Remove Method 153
  - Collection.RemoveAll Method 153

ComponentView Class 153
- ComponentView Class Properties 154
- ComponentView Class Methods 154
  - ComponentView.GetComponent Method 155

ConnectionRelation Class Overview 155
- ConnectionRelation Class Properties 156
  - ConnectionRelation. Characteristics Property 156
  - ConnectionRelation. SupplierIsDevice Property 157
- ConnectionRelation Class Methods 157

ContextMenuItem Class 158
- ContextMenuItem Class Properties 158
  - ContextMenuItem.Caption Property 158
  - ContextMenuItem.InternalName Property 159
  - ContextMenuItem.MenuID Property 159
  - ContextMenuItem.MenuState Property 160
- ContextMenuItem Class Methods 160

ContextMenuItemType Enumeration 161

ControllableUnit Class 162
ControllableUnit Class Properties 163
ControllableUnit Class Methods 163
ControllableUnit.Control Method 165
ControllableUnit.GetAllSubUnitItems Method 165
ControllableUnit.GetFileName Method 166
ControllableUnit.GetSubUnitItems Method 166
ControllableUnit.IsControlled Method 167
ControllableUnit.IsLoaded Method 167
ControllableUnit.IsLocked Method 168
ControllableUnit.IsModifiable Method 168
ControllableUnit.IsModified Method 169
ControllableUnit.Load Method 169
ControllableUnit.Lock Method 170
ControllableUnit.Modifiable Method 170
ControllableUnit.NeedsRefreshing Method 171
ControllableUnit.Refresh Method 171
ControllableUnit.Save Method 172
ControllableUnit.SaveAs Method 172
ControllableUnit.Uncontrol Method 173
ControllableUnit.Unload Method 173
ControllableUnit.Unlock Method 174

Decision Class 174
Decision Class Properties 175
Decision Class Methods 175

DecisionView Class 175
DecisionView Class Properties 176
DecisionView Class Methods 176
DecisionView.GetDecision Method 176

DefaultModelProperties Class 177
DefaultModelProperties Class Properties 177
DefaultModelProperties Class Methods 178
DefaultModelProperties.AddDefaultProperty Method 179
DefaultModelProperties.CloneDefaultPropertySet Method 180
DefaultModelProperties.CreateDefaultPropertySet Method 181
DefaultModelProperties.DeleteDefaultProperty Method 182
DefaultModelProperties.DeleteDefaultPropertySet Method 183
DefaultModelProperties.FindDefaultProperty Method 184

DefaultModelProperties.GetDefaultPropertySet Method 185
DefaultModelProperties.GetDefaultSetNames Method 186
DefaultModelProperties.GetToolNamesForClass Method 187
DefaultModelProperties.IsToolVisible Method 188
DefaultModelProperties.SetToolVisibility Method 188

DependencyRelation Class 189
DependencyRelation Class Properties 189
DependencyRelation Class Methods 190

DeploymentDiagram Class 190
DeploymentDiagram Class Properties 190
DeploymentDiagram Class Methods 191
DeploymentDiagram.AddDevice Method 192
DeploymentDiagram.AddProcessor Method 193
DeploymentDiagram.GetDevices Method 193
DeploymentDiagram.GetProcessors Method 194
DeploymentDiagram.RemoveDevice Method 194
DeploymentDiagram.RemoveProcessor Method 195

DeploymentUnit Class 195
DeploymentUnit Class Properties 196
DeploymentUnit Class Methods 196

Device Class 196
Device Class Properties 197
Device.Characteristics Property 197
Device. Connections Property 198
Device Class Methods 198
Device.AddDeviceConnection Method 199
Device.AddProcessorConnection Method 199
Device.GetConnectedDevices Method 200
Device.GetConnectedProcessors Method 200
Device.RemoveDeviceConnection Method 201
Device.RemoveProcessorConnection Method 201

Diagram Class 202
Diagram Class Properties 202
Diagram.Documentation Property 202
Diagram.ExternalDocuments Property 203
Diagram.Items Property 203

Diagram.ItemViews Property 204
Diagram.Visible Property 204
Diagram.ZoomFactor Property 205
Diagram Class Methods 205
Diagram.Activate Method 207
Diagram.AddExternalDocument Method 208
Diagram.AddNoteView Method 209
Diagram.AddRelationView Method 209
Diagram.DeleteExternalDocument Method 210
Diagram.Exists Method 210
Diagram.GetNoteViews Method 211
Diagram. GetParentContext Method 211
Diagram.GetSelectedItems Method 212
Diagram.GetViewFrom Method 212
Diagram.Invalidate Method 213
Diagram.IsActive Method 213
Diagram.Layout Method 214
Diagram.RemoveItemView Method 214
Diagram.RemoveNoteView Method 215
Diagram.Render Method 215
Diagram.RenderEnhanced Method 216
Diagram.RenderEnhancedToClipBoard Method 216
Diagram.RenderToClipboard Method 217
Diagram.Update Method 217

Element Class 218
Element Class Properties 218
Element.Application Property 218
Element.Model Property 219
Element.Name Property 219
Element Class Methods 219
Element.CreateProperty Method 221
Element.FindDefaultProperty Method 222
Element.FindProperty Method 223
Element.GetAllProperties Method 223
Element.GetCurrentPropertySetName Method 224
Element.GetDefaultPropertyValue Method 224
Element.GetDefaultSetNames Method 225
Element. GetIconIndex Method 225
Element.GetPropertyClassName Method 226
Element.GetPropertyValue Method 226
Element.GetQualifiedName Method 227

Element.GetToolNames Method 228
Element.GetToolProperties Method 228
Element.GetUniqueId Method 229
Element.InheritProperty Method 229
Element.IsDefaultProperty Method 230
Element.IsOverriddenProperty Method 231
Element.OverrideProperty Method 232
Element. RenderIconToClipboard Method 233
Element.SetCurrentPropertySetName Method 234

Event Class 234
Event Class Properties 235
Event.Arguments Property 235
Event.GuardCondition Property 235
Event.Name Property 236
Event Class Methods 236
Event.GetAction Method 236

ExternalDocument Class 237
ExternalDocument Class Properties 237
ExternalDocument.ParentCategory Property 238
ExternalDocument. ParentDiagram Property 238
ExternalDocument. ParentItem Property 239
ExternalDocument.Path Property 239
ExternalDocument.URL Property 240
ExternalDocument Class Methods 240
ExternalDocument. GetIconIndex Method 241
ExternalDocument.IsURL Method 241
ExternalDocument.Open Method 242
ExternalDocument. RenderIconToClipboard Method 243

HasRelationship Class 244
HasRelationship Class Properties 244
HasRelationship.ClientCardinality Property 244
HasRelationship.Containment Property 245
HasRelationship.ExportControl Property 246
HasRelationship.Static Property 247
HasRelationship.SupplierCardinality Property 247
HasRelationship Class Methods 247

InheritRelation Class 248
InheritRelation Class Properties 248
InheritRelation.ExportControl Property 249
InheritRelation.FriendshipRequired Property 250
InheritRelation.Virtual Property 250
InheritRelation Class Methods 250

InstanceView Class 251
InstanceView Class Properties 251
InstanceView Class Methods 251
InstanceView.GetInstance Method 252

InstantiateRelation Class 252
InstantiateRelation Class Properties 252
InstantiateRelation Class Methods 253

LineVertex Class 253
LineVertex Class Properties 253
LineVertex Class Methods 254
LineVertex.GetXPosition Method 254
LineVertex.GetYPosition Method 255
LineVertex Sample Rational Rose Script 256

Link Class 258
Link Class Properties 258
Link.LinkRole1 Property 259
Link.LinkRole2 Property 259
Link.LinkRole1Shared Property 259
Link.LinkRole2Shared Property 260
Link.LinkRole1Visibility Property 261
Link.LinkRole2Visibility Property 262
Link Class Methods 263
Link.AddMessageTo Method 264
Link.AssignAssociation Method 265
Link.DeleteMessage Method 265
Link.GetAssociation Method 266
Link.GetMessages Method 266
Link.UnAssignAssociation Method 267

MenuState Enumeration 268

Message Class 270
Message Class Properties 270
Message.Frequency Property 270
Message.Synchronization Property 271
Message Class Methods 272
Message.GetLink Method 272
Message.GetOperation Method 273
Message.GetReceiverObject Method 273
Message.GetSenderObject Method 274
Message. GetSequenceInformation Method 275
Message.IsMessageToSelf Method 276
Message.IsOperation Method 276

Model Class 277
Model Class Properties 278
Model.DefaultLanguage Property 279
Model.DefaultProperties Property 279
Model.DeploymentDiagram Property 280
Model.DeploymentUnit Property 280
Model.Notation Property 281
Model.RootCategory Property 281
Model.RootSubsystem Property 282
Model.RootUseCaseCategory Property 282
Model.UseCases Property 283
Model Class Methods 283
Model.AddDevice Method 286
Model.AddProcessor Method 286
Model.DeleteDevice Method 287
Model.DeleteProcessor Method 287
Model.FindCategories Method 288
Model.FindCategoryWithID Method 288
Model.FindClasses Method 289
Model.FindClassWithID Method 289
Model. FindDiagramWithID Method 290
Model.FindItems Method 290
Model.FindItemWithID Method 291
Model.GetActiveDiagram Method 291
Model.GetAllCategories Method 292
Model.GetAllClasses Method 292
Model.GetAllDevices Method 293
Model.GetAllModules Method 293

Model.GetAllProcessors Method 294
Model.GetAllSubsystems Method 294
Model.GetAllUseCases Method 295
Model.GetObject Method 295
Model.GetSelectedCategories Method 296
Model.GetSelectedClasses Method 296
Model.GetSelectedDiagrams Method 297
Model.GetSelectedExternalDocuments Method 297
Model.GetSelectedItems Method 298
Model.GetSelectedModules Method 299
Model.GetSelectedSubsystems Method 299
Model.GetSelectedUseCases Method 300
Model.Import Method 300
Model.LoadControlledUnits Method 301
Model.ResolveReferences Method 301

Module Class 302
Module Class Properties 302
Module.Declarations Property 303
Module.OtherPart Property 303
Module.ParentSubSystem Property 304
Module.Part Property 305
Module.Path Property 306
Module.AssignedLanguage Property 306
Module.Type Property 307
Module Class Methods 308
Module.AddRealizeRel Method 309
Module.AddVisibilityRelationship Method 310
Module.DeleteRealizeRel Method 310
Module.DeleteVisibilityRelationship Method 311
Module.GetAllDependencies Method 311
Module.GetAssignedClasses Method 312
Module.GetDependencies Method 312
Module.GetRealizeRelations Method 313
Module.GetSubsystemDependencies Method 313

ModuleDiagram Class 314
ModuleDiagram Class Properties 314
ModuleDiagram.ComponentViews Property 314
ModuleDiagram.ParentSubsystem Property 315
ModuleDiagram.SubsystemViews Property 315

ModuleDiagram Class Methods 316
ModuleDiagram.AddComponentView Method 317
ModuleDiagram.AddSubsystemView Method 318
ModuleDiagram.GetModules Method 319
ModuleDiagram.GetSelectedModules Method 319
ModuleDiagram.GetSelectedSubsystems Method 320
ModuleDiagram.GetSubsystems Method 320
ModuleDiagram.RemoveComponentView Method 321
ModuleDiagram.RemoveSubsystemView Method 322

ModuleVisibilityRelationship Class 322
ModuleVisibilityRelationship Class Properties 323
ModuleVisibilityRelationship.ContextModule Property 323
ModuleVisibilityRelationship.ContextSubsystem Property 324
ModuleVisibilityRelationship.SupplierModule Property 324
ModuleVisibilityRelationship.SupplierSubsystem Property 325
ModuleVisibilityRelationship Class Methods 325

NotationTypes Enumeration 326

NoteView Class 327
NoteView Class Properties 327
NoteView.Text Property 327
NoteView Class Methods 328
NoteView. AddAttachmentToView Method 329
NoteView. GetDiagramLink Method 330
NoteView.GetNoteViewType Method 330

ObjectFlow Class Overview 331
ObjectFlow Class Properties 331
ObjectFlow Class Methods 332

ObjectInstance Class 332
ObjectInstance Class Properties 332
ObjectInstance.ClassName Property 333
ObjectInstance.MultipleInstances Property 333
ObjectInstance.Links Property 333
ObjectInstance.Persistence Property 334

ObjectInstance Class Methods 334
ObjectInstance.AddLink Method 335
ObjectInstance.DeleteLink Method 336
ObjectInstance.GetClass Method 336
ObjectInstance.IsClass Method 337

Operation Class 337
Operation Class Properties 338
Operation.Concurrency Property 339
Operation.Exceptions Property 339
Operation.ExportControl Property 340
Operation.Parameters Property 341
Operation.ParentClass Property 341
Operation.Postconditions Property 342
Operation.Preconditions Property 342
Operation.Protocol Property 342
Operation.Qualification Property 343
Operation.ReturnType Property 343
Operation.Semantics Property 343
Operation.Size Property 344
Operation.Time Property 344
Operation.Virtual Property 344
Operation Class Methods 345
Operation.AddParameter Method 345
Operation.DeleteParameter Method 346
Operation.RemoveAllParameters Method 346

Package Class 347
Package Class Properties 347
Package Class Methods 347
Package.IsRootPackage Method 348

Parameter Class 348
Parameter Class Properties 348
Parameter.Const Property 349
Parameter.InitValue Property 349
Parameter.Type Property 349
Parameter Class Methods 350

PathMap Class 350
PathMap Class Properties 350
PathMap Class Methods 351
PathMap.AddEntry Method 352
PathMap.DeleteEntry Method 352
PathMap.GetActualPath Method 353
PathMap.GetActualPathWithContext Method 353
PathMap.GetObject Method 355
PathMap.GetVirtualPath Method 355
PathMap.GetVirtualPathWithContext Method 356
PathMap.HasEntry Method 357

Process Class 358
Process Class Properties 358
Process.MyProcessor Property 358
Process.Priority Property 359
Process Class Methods 359

Processor Class 359
Processor Class Properties 360
Processor.Characteristics Property 360
Processor. Connections Property 361
Processor.Processes Property 361
Processor.Scheduling Property 362
Processor Class Methods 363
Processor.AddDeviceConnection Method 364
Processor.AddProcess Method 364
Processor.AddProcessorConnection Method 365
Processor.DeleteProcess Method 365
Processor.GetConnectedDevices Method 366
Processor.GetConnectedProcessors Method 366
Processor.RemoveDeviceConnection Method 367
Processor.RemoveProcessorConnection Method 367

Property Class 368
Property Class Properties 368
Property.Name Property 368
Property.ToolName Property 369
Property.Type Property 369
Property.Value Property 370

Property Class Methods 370
Property.GetObject Method 370

RealizeRelation Class 371
RealizeRelation Class Properties 371
RealizeRelation Class Methods 371
RealizeRelation.GetContextClass Method 372
RealizeRelation.GetContextComponent Method 372
RealizeRelation.GetSupplierClass Method 373
RealizeRelation.GetSupplierComponent Method 373

Relation Class 374
Relation Class Properties 374
Relation.SupplierName Property 374
Relation Class Methods 375
Relation.GetClient Method 375
Relation.GetSupplier Method 376
Relation.HasClient Method 376
Relation.HasSupplier Method 377

Role Class 377
Role Class Properties 377
Role.Aggregate Property 378
Role.AssociateItem Property 379
Role.Association Property 379
Role.Cardinality Property 379
Role.Class Property 380
Role.Constraints Property 380
Role.Containment Property 381
Role.ExportControl Property 382
Role.Keys Property 383
Role.Navigable Property 383
Role.Static Property 383
Role.UseCase Property 384
Role.Friend Property 384
Role Class Methods 385
Role.DeleteKey Method 385
Role.AddKey Method 386
Role.GetClassName Method 386

RoseAddIn Class 387
  RoseAddIn Class Properties 387
    RoseAddIn.CompanyName Property 388
    RoseAddIn.Copyright Property 388
    RoseAddIn.EventHandler Property 389
    RoseAddIn.FundamentalTypes Property 389
    RoseAddIn.HelpFilePath Property 390
    RoseAddIn.InstallDirectory Property 390
    RoseAddIn.MenuFilePath Property 391
    RoseAddIn.Name Property 391
    RoseAddIn.PropertyFilePath Property 392
    RoseAddIn.RootRegistryPath Property 392
    RoseAddIn.ServerName Property 393
    RoseAddIn.ToolNames Property 393
    RoseAddIn.Version Property 394
  RoseAddIn Class Methods 394
    RoseAddIn.Activate Method 395
    RoseAddIn.AddContextMenuItem Method 395
    RoseAddIn.Deactivate Method 397
    RoseAddIn.ExecuteScript Method 397
    RoseAddIn.GetContextMenuItems Method 398
    RoseAddIn.GetDisplayName Method 399
    RoseAddIn.IsActive Method 399
    RoseAddIn.IsLanguageAddIn Method 400
    RoseAddIn.ReadSetting Method 400
    RoseAddIn.WriteSetting Method 401

RoseAddInEventTypes Enumeration 401

RoseAddInManager Class 402
  RoseAddInManager Class Properties 402
    RoseAddInManager.AddIns Property 402
  RoseAddInManager Class Methods 403
    RoseAddInManager.DisableEvents Method 403
    RoseAddInManager.EnableEvents Method 404

RoseItem Class 405
  RoseItem Class Properties 405
    RoseItem.Documentation Property 405
    RoseItem.ExternalDocuments Property 406

RoseItem.LocalizedStereotype Property 406
RoseItem.StateMachineOwner Property 407
RoseItem.Stereotype Property 407
RoseItem Class Methods 408
RoseItem.AddExternalDocument Method 409
RoseItem.DeleteExternalDocument Method 410
RoseItem.GetRoseItem Method 410
RoseItem.OpenCustomSpecification Method 411
RoseItem.OpenSpecification Method 411

RoseItemView Class 412
RoseItemView Class Properties 412
RoseItemView.Height Property 413
RoseItemView.LineVertices Property 414
RoseItemView.Name Property 414
RoseItemView.StereotypeDisplay Property 415
RoseItemView.Width Property 415
RoseItemView.XPosition Property 416
RoseItemView.YPosition Property 416
Values for RoseItemView Color Properties 416
RoseItemView.FillColor.Blue Property 417
RoseItemView.FillColor.Green Property 418
RoseItemView.FillColor.Red Property 418
RoseItemView.FillColor.Transparent Property 419
RoseItemView.Font.Blue Property 419
RoseItemView.Font.Bold Property 420
RoseItemView.Font.FaceName Property 420
RoseItemView.Font.Green Property 421
RoseItemView.Font.Italic Property 421
RoseItemView.Font.Red Property 422
RoseItemView.Font.Size Property 422
RoseItemView.Font.StrikeThrough Property 422
RoseItemView.Font.Underline Property 423
RoseItemView.Item Property 423
RoseItemView.LineColor.Blue Property 424
RoseItemView.LineColor.Green Property 424
RoseItemView.LineColor.Red Property 425
RoseItemView.ParentDiagram Property 425
RoseItemView.ParentView Property 426
RoseItemView.SubViews Property 426

RoseItemView Class Methods 427
RoseItemView.Invalidate Method 428
RoseItemView.GetAttachedNotes Method 428
RoseItemView.GetDefaultHeight Method 429
RoseItemView.GetDefaultWidth Method 429
RoseItemView.GetMinHeight Method 430
RoseItemView.GetMinWidth Method 430
RoseItemView.HasItem Method 431
RoseItemView.HasParentView Method 431
RoseItemView.IsSelected Method 432
RoseItemView.PointInView Method 432
RoseItemView.SetSelected Method 433
RoseItemView.SupportsFillColor Method 433
RoseItemView.SupportsLineColor Method 434

RoseObject Class 434
RoseObject Class Properties 434
RoseObject Class Methods 435
RoseObject.CanTypeCast Method 435
RoseObject.GetObject Method 437
RoseObject.IdentifyClass Method 437
RoseObject.IsClass Method 438
RoseObject.TypeCast Method 439

ScenarioDiagram Class 440
ScenarioDiagram Class Properties 440
ScenarioDiagram.InstanceViews Property 440
ScenarioDiagram Class Methods 441
ScenarioDiagram.AddInstance Method 442
ScenarioDiagram.AddInstanceView Method 443
ScenarioDiagram.CreateMessage Method 444
ScenarioDiagram.CreateObject Method 445
ScenarioDiagram.DeleteInstance Method 445
ScenarioDiagram.GetDiagramType Method 446
ScenarioDiagram.GetMessages Method 446
ScenarioDiagram.GetObjects Method 447
ScenarioDiagram.GetSelectedLinks Method 447
ScenarioDiagram.GetSelectedMessages Method 448
ScenarioDiagram.GetSelectedObjects Method 448
ScenarioDiagram.RemoveInstanceView Method 449

State Class 449
State Class Properties 450
State.History Property 450
State.StateKind Property 451
State Class Methods 452
State.AddState Method 452
State.DeleteState Method 453
State.RelocateState Method 453

StateDiagram Class 454
StateDiagram Class Properties 455
StateDiagram.IsActivityDiagram Property 455
StateDiagram.Parent Property 455
StateDiagram Class Methods 456
StateDiagram.AddActivityView Method 458
StateDiagram.AddDecisionView Method 459
StateDiagram.AddStateView Method 459
StateDiagram.AddSwimLaneView Method 460
StateDiagram.AddSynchronizationView Method 461
StateDiagram.GetActivityViews Method 462
StateDiagram.GetDecisionViews Method 463
StateDiagram.GetDiagramActivityViews Method 464
StateDiagram.GetDiagramDecisionViews Method 464
StateDiagram.GetDiagramStateVertexViews Method 465
StateDiagram.GetDiagramSwimLaneViews Method 465
StateDiagram.GetDiagramSynchronizationViews Method 466
StateDiagram.GetSelectedActivities Method 466
StateDiagram.GetSelectedActivityViews Method 467
StateDiagram.GetSelectedDecisionViews Method 467
StateDiagram.GetSelectedDecisions Method 468
StateDiagram.GetSelectedStates Method 468
StateDiagram.GetSelectedStateVertices Method 469
StateDiagram.GetSelectedStateViews Method 469
StateDiagram.GetSelectedSwimLaneViews Method 470
StateDiagram.GetSelectedSynchronizations Method 470
StateDiagram.GetSelectedSynchronizationViews Method 471
StateDiagram.GetSelectedTransitions Method 471
StateDiagram.GetStateVertexViews Method 472
StateDiagram.GetStateView Method 473
StateDiagram.GetStateViews Method 473
StateDiagram.GetSwimLaneViews Method 474

StateDiagram.GetSynchronizationViews Method 475
StateDiagram.RemoveActivityView Method 476
StateDiagram.RemoveDecisionView Method 477
StateDiagram.RemoveStateView Method 478
StateDiagram.RemoveSwimLaneView Method 479
StateDiagram.RemoveSynchronizationView Method 480

StateMachine Class 480
StateMachine Class Properties 481
StateMachine.AbstractStates Property 482
StateMachine.Activities Property 483
StateMachine.Decisions Property 483
StateMachine.Diagrams Property 484
StateMachine.Documentation Property 484
StateMachine.ExternalDocuments Property 485
StateMachine.LocalizedStereotype Property 485
StateMachine.ParentClass Property 486
StateMachine.StateMachineOwner Property 486
StateMachine.States Property 487
StateMachine.StateVertices Property 487
StateMachine.Stereotype Property 488
Description 488
Syntax 488
Property Type 488
StateMachine.SwimLanes Property 488
StateMachine.Synchronizations Property 489
StateMachine Class Methods 490
StateMachine.AddActivity Method 492
StateMachine.AddActivityDiagram Method 492
StateMachine.AddDecision Method 493
StateMachine.AddExternalDocument Method 494
StateMachine.AddState Method 495
StateMachine.AddStateChartDiagram Method 495
StateMachine.AddSwimLane Method 496
StateMachine.AddSynchronization Method 496
StateMachine.DeleteExternalDocument Method 497
StateMachine.DeleteState Method 497
StateMachine.DeleteStateVertex Method 498
StateMachine.DeleteSwimLane Method 498
StateMachine.GetAllAbstractStates Method 499
StateMachine.GetAllActivities Method 499
StateMachine.GetAllDecisions Method 500

StateMachine.GetAllDiagrams Method 500
StateMachine.GetAllStates Method 501
StateMachine.GetAllStateVertices Method 501
StateMachine.GetAllSynchronizations Method 502
StateMachine.GetAllTransitions Method 502
StateMachine.GetTransitions Method 503
StateMachine.OpenCustomSpecification Method 503
StateMachine.OpenSpecification Method 504
StateMachine.RelocateState Method 504

StateMachineOwner Class 505

StateMachineOwner Class Properties 505
StateMachineOwner.StateMachines Property 506

StateMachineOwner Class Methods 506
StateMachineOwner.CreateStateMachine Method 507
StateMachineOwner.DeleteStateMachine Method 508
StateMachineOwner.GetParentItem Method 508

StateVertex Class 509

StateVertex Class Properties 509
StateVertex.Parent Property 510
StateVertex.ParentStateMachine Property 510
StateVertex.Transitions Property 511

StateVertex Class Methods 511
StateVertex.AddTransition Method 512
StateVertex. AddTransitionToVertex Method 513
StateVertex.DeleteTransition Method 514
StateVertex.GetSwimLanes Method 514

StateView Class 515

StateView Class Properties 515

StateView Class Methods 515
StateView.GetState Method 516

Subsystem Class 516

Subsystem Class Properties 516
Subsystem.ModuleDiagrams Property 517
Subsystem.Modules Property 517
Subsystem.ParentSubsystem Property 518
Subsystem.Subsystems Property 518

Subsystem Class Methods 519
Subsystem.AddModule Method 520
Subsystem.AddModuleDiagram Method 521
Subsystem.AddSubsystem Method 521
Subsystem.DeleteModule Method 522
Subsystem.DeleteSubsystem Method 522
Subsystem.GetAllModules Method 523
Subsystem.GetAllSubsystems Method 523
Subsystem.GetAssignedCategories Method 524
Subsystem.GetAssignedClasses Method 524
Subsystem.GetSubsystemDependencies Method 525
Subsystem.GetVisibilityRelations Method 525
Subsystem.GetVisibleSubsystems Method 528
Subsystem.RelocateModule Method 528
Subsystem.RelocateModuleDiagram Method 529
Subsystem.RelocateSubSystem Method 529
Subsystem.TopLevel Method 530

SubsystemView Class 530
SubsystemView Class Properties 530
SubsystemView Class Methods 531
SubsystemView.GetSubsystem Method 531

SwimLane Class 532
SwimLane Class Properties 532
SwimLane Class Methods 532
SwimLane.GetStateVertices Method 533

SwimLaneView Class 533
SwimLaneView Class Properties 534
SwimLaneView Class Methods 534

SyncItem Class 534
SyncItem Class Properties 535
SyncItem Class Methods 535

SyncItemView Class 535
SyncItemView Class Properties 536
SyncItemView.Horizontal Property 536
SyncItemView Class Methods 537
SyncItemView.GetSynchronization Method 537

Transition Class 538
Transition Class Properties 538
Transition Class Methods 538
Transition.GetSendAction Method 539
Transition.GetSourceState Method 540
Transition. GetSourceStateVertex Method 540
Transition.GetTargetState Method 541
Transition. GetTargetStateVertex Method 541
Transition.GetTriggerAction Method 542
Transition.GetTriggerEvent Method 542
Transition.RedirectTo Method 543

UseCase Class 543
UseCase Class Properties 543
UseCase.Abstract Property 544
UseCase.ClassDiagrams Property 544
UseCase.ParentCategory Property 545
UseCase.Rank Property 545
UseCase.ScenarioDiagrams Property 546
UseCase.StateMachine Property 546
UseCase Class Methods 548
UseCase.AddAssociation Method 549
UseCase.AddClassDiagram Method 550
UseCase.AddInheritRel Method 550
UseCase.AddScenarioDiagram Method 551
UseCase.AddUseCaseDiagram Method 552
UseCase.CreateStateMachine Method 553
UseCase.DeleteAssociation Method 553
UseCase.DeleteClassDiagram Method 554
UseCase.DeleteInheritRel Method 554
UseCase.DeleteScenarioDiagram Method 555
UseCase.GetAssociations Method 555
UseCase.GetInheritRelations Method 556
UseCase.GetRoles Method 556
UseCase.GetSuperUseCases Method 557

Appendix A REI Inheritance Diagrams 559

Introduction 559

RoseBase Inheritance Diagram 560

RoseObject Inheritance Diagram 561

Element Inheritance Diagram 562

Diagram Inheritance Diagram 563

RoseItem Inheritance Diagram 564

ControllableUnit Inheritance Diagram 565

Deployment Classes 566

Related Logical Classes 567

Relation Inheritance Diagram 568

ClassRelation Inheritance Diagram 569

Scenario Classes 570

State Classes 571

RoseItemView Inheritance Diagram 572

SecondaryRoseObject Inheritance Diagram 573

Appendix B REI Events 575

Introduction 575

OnActivate Event 576
- Description 576
- Registry and Server Requirements 576
- Syntax 576
- See Also 576

OnAppInit Event 577
- Description 577
- Registry and Server Requirements 577
- Syntax 577

OnBrowseBody Event 577
- Description 577
- Registry and Server Requirements 578
- Syntax 578
- See Also 578

OnBrowseHeader Event 579
- Description 579
- Registry and Server Requirements 579
- Syntax 580
- See Also 580

OnCancelModel Event 580
- Description 580
- Registry and Server Requirements 580
- Syntax 581
- See Also 581

OnCloseModel Event 581
- Description 581
- Registry and Server Requirements 581
- Syntax 582
- See Also 582

OnDeactivate Event 582
- Description 582
- Registry and Server Requirements 583
- Syntax 583
- See Also 583

OnDeletedModelElement Event 583
- Description 583
- Registry and Server Requirements 584
- Syntax 584
- See Also 584

OnEnableContextMenuItems Event 585
Description 585
Registry and Server Requirements 585
Syntax 585
Return Type 585
See Also 586
OnGenerateCode Event 586
Description 586
Registry and Server Requirements 587
Syntax 587
See Also 587
OnModifiedModelElement Event 587
Description 587
Assigned Language Changed: 587
Name Changed: 588
Registry and Server Requirements 589
Syntax 589
See Also 589
OnNewModel Event 590
Description 590
Registry and Server Requirements 590
Syntax 590
See Also 590
OnNewModelElement Event 591
Description 591
Registry and Server Requirements 592
Syntax 592
See Also 592
OnOpenModel Event 592
Description 592
Registry and Server Requirements 592
Syntax 593
See Also 593

OnPropertySpecOpen Event 593
  Description 593
  Registry and Server Requirements 594
  Syntax 594
  Return Type 594
  See Also 594

OnSaveModel Event 595
  Description 595
  Registry and Server Requirements 595
  Syntax 595
  See Also 596

OnSelectedContextMenuItem Event 596
  Description 596
  Registry and Server Requirements 596
  Syntax 596
  Return Type 597
  See Also 597

Index 599

List of Figures

Figure 1 Example: Comparing GetAssociateRoles and GetRoles 109
Figure 2 LineVertex Objects 253
Figure 3 Example: Retrieving Line Vertices 257
Figure 4 Example: Retrieving Module Visibility Relationships 527
Figure 5 RoseBase Inheritance Diagram 560
Figure 6 RoseObject Inheritance Diagram 561
Figure 7 Element Inheritance Diagram 562
Figure 8 Diagram Inheritance Diagram 563
Figure 9 RoseItem Inheritance Diagram 564
Figure 10 ControllableUnit Inheritance Diagram 565
Figure 11 Deployment Classes 566
Figure 12 Related Logical Classes 567
Figure 13 Relation Inheritance Diagram 568
Figure 14 ClassRelation Inheritance Diagram 569
Figure 15 Scenario Classes 570
Figure 16 State Classes 571
Figure 17 RoseItemView Inheritance Diagram 572
Figure 18 SecondaryRoseObject Inheritance Diagram 573

List of Tables

Table 1 AbstractState Class Properties Summary 2
Table 2 AbstractState Class Methods Summary 5
Table 3 Action Class Properties Summary 15
Table 4 Action Class Methods Summary 16
Table 5 Activity Class Properties Summary 17
Table 6 Activity Class Methods Summary 18
Table 7 ActivityView Class Properties Summary 18
Table 8 ActivityView Class Methods Summary 19
Table 9 Application Class Properties Summary 20
Table 10 Application Class Methods Summary 27
Table 11 Association Class Properties Summary 45
Table 12 Association Class Methods Summary 49
Table 13 Attribute Class Properties Summary 53
Table 14 Attribute.Containment Rich Data Types 54
Table 15 Attribute.ExportControl Rich Data Types 55
Table 16 Attribute Class Methods Summary 57
Table 17 Category Class Properties Summary 58
Table 18 Category Class Methods Summary 63
Table 19 CategoryDependency Class Properties Summary 78
Table 20 CategoryDependency Class Methods Summary 79
Table 21 Class Class Properties Summary 81
Table 22 Class.ClassKind Rich Data Types 83
Table 23 Class.Concurrency Rich Data Types 84
Table 24 Class.ExportControl Rich Data Types 85
Table 25 Class Class Methods Summary 89

Table 26 ClassDependency Class Properties Summary 119
Table 27 ClassDependency.Export Control Rich Data Types 120
Table 28 ClassDependency Class Methods Summary 121
Table 29 ClassDiagram Class Properties Summary 122
Table 30 ClassDiagram Class Methods Summary 123
Table 31 ClassRelation Class Properties Summary 133
Table 32 ClassRelation Class Methods Summary 133
Table 33 ClassView Class Properties Summary 135
Table 34 ClassView Class Methods Summary 139
Table 35 ClientRelKind Enumeration Valid Values 142
Table 36 ClientRelType Enumeration Valid Values 143
Table 37 Collection Class Properties Summary 145
Table 38 Collection Class Methods Summary (for all collections) 146
Table 39 User-Defined Collection Class Methods Summary 151
Table 40 ComponentView Class Properties Summary 154
Table 41 ComponentView Class Methods Summary 154
Table 42 ConnectionRelation Class Properties Summary 156
Table 43 ConnectionRelation Class Methods Summary 157
Table 44 ContextMenuItem Class Properties Summary 158
Table 45 ContextMenuItem Class Methods Summary 160
Table 46 ContextMenuItemType Enumeration Valid Values 161
Table 47 ControllableUnit Class Properties Summary 163
Table 48 ControllableUnit Class Methods Summary 163
Table 49 Decision Class Properties Summary 175
Table 50 Decision Class Methods Summary 175
Table 51 DecisionView Class Properties Summary 176
Table 52 DecisionView Class Methods Summary 176
Table 53 DefaultModelProperties Class Properties Summary 177
Table 54 DefaultModelProperties Class Methods Summary 178
Table 55 DependencyRelation Class Properties Summary 189
Table 56 DependencyRelation Class Methods Summary 190
Table 57 DeploymentDiagram Class Properties Summary 190
Table 58 DeploymentDiagram Class Methods Summary 191
Table 59 DeploymentUnit Class Properties Summary 196
Table 60 DeploymentUnit Class Methods Summary 196
Table 61 Device Class Properties Summary 197

Table 62 Device Class Methods Summary 198
Table 63 Diagram Class Properties Summary 202
Table 64 Diagram Class Methods Summary 205
Table 65 Element Class Properties Summary 218
Table 66 Element Class Methods Summary 219
Table 67 Event Class Properties Summary 235
Table 68 Event Class Methods Summary 236
Table 69 ExternalDocument Class Properties Summary 237
Table 70 ExternalDocument Class Methods Summary 240
Table 71 HasRelationship Class Properties Summary 244
Table 72 HasRelationship.Containment Rich Data Types 245
Table 73 HasRelationship.Export Control Rich Data Types 246
Table 74 Has Relationship Class Methods Summary 247
Table 75 InheritRelation Class Properties Summary 248
Table 76 InheritRelation.Export Control Rich Data Types 249
Table 77 InheritRelation Class Methods Summary 250
Table 78 InstanceView Class Properties Summary 251
Table 79 InstanceView Class Methods Summary 251
Table 80 InstantiateRelation Class Properties Summary 252
Table 81 InstantiateRelation Class Methods Summary 253
Table 82 LineVertex Class Methods Summary 254
Table 83 Link Class Properties Summary 258
Table 84 Link.LinkRole1Visibility Rich Data Types 261
Table 85 Link.LinkRose2Visibility Rich Data Types 262
Table 86 Link Class Methods Summary 263
Table 87 MenuState Enumeration Valid Values 268
Table 88 Message Class Properties Summary 270
Table 89 Message Class Methods Summary 272
Table 90 Model Class Properties Summary 278
Table 91 Model Class Methods Summary 283
Table 92 Module Class Properties Summary 302
Table 93 Module.Part Rich Data Types 305
Table 94 Module.Type Rich Data Types 307
Table 95 Module Class Methods Summary 308
Table 96 ModuleDiagram Class Properties Summary 314
Table 97 ModuleDiagram Class Methods Summary 316

Table 98 ModuleVisibilityRelationship Class Properties Summary 323
Table 99 ModuleVisibilityRelationship Class Methods Summary 325
Table 100 NotationTypes Enumeration Valid Values 326
Table 101 NoteView Class Properties Summary 327
Table 102 NoteView Class Methods Summary 328
Table 103 ObjectFlow Class Properties Summary 331
Table 104 ObjectInstance Class Properties Summary 332
Table 105 ObjectFlow Class Methods Summary 332
Table 106 ObjectInstance Class Methods Summary 334
Table 107 Operation Class Properties Summary 338
Table 108 Operation.Concurrency Rich Data Types 339
Table 109 Operation.ExportControl Rich Data Types 340
Table 110 Operation Class Methods Summary 345
Table 111 Package Class Properties Summary 347
Table 112 Package Class Methods Summary 347
Table 113 Parameter Class Properties Summary 348
Table 114 Parameter Class Methods Summary 350
Table 115 PathMap Class Methods Summary 351
Table 116 Process Class Properties Summary 358
Table 117 Process Class Methods Summary 359
Table 118 Processor Class Properties Summary 360
Table 119 Processor.Scheduling Rich Data Type 362
Table 120 Processor Class Methods Summary 363
Table 121 Property Class Properties Summary 368
Table 122 Property Class Methods Summary 370
Table 123 RealizeRelation Class Properties Summary 371
Table 124 RealizeRelation Class Methods Summary 371
Table 125 Relation Class Properties Summary 374
Table 126 Relation Class Methods Summary 375
Table 127 Role Class Properties Summary 377
Table 128 Role.Containment Rich Data Types 381
Table 129 Role.ExportControl Rich Data Types 382
Table 130 Role Class Methods Summary 385
Table 131 RoseAddIn Class Properties Summary 387
Table 132 RoseAddIn Class Methods Summary 394
Table 133 RoseAddInEventTypes Enumeration Valid Values 401

Table 134 RoseAddInManager Class Properties Summary 402
Table 135 RoseAddInManager Class Methods Summary 403
Table 136 RoseItem Class Properties Summary 405
Table 137 RoseItem Class Methods Summary 408
Table 138 RoseItemView Class Properties Summary 412
Table 139 RoseItemView Class Methods Summary 427
Table 140 RoseObject Class Methods Summary 435
Table 141 ScenarioDiagram Class Properties Summary 440
Table 142 ScenarioDiagram Class Methods Summary 441
Table 143 State Class Properties Summary 450
Table 144 State.StateKind Rich Data Types 451
Table 145 State Class Methods Summary 452
Table 146 StateDiagram Class Properties Summary 455
Table 147 StateDiagram Class Methods Summary 456
Table 148 StateMachine Class Properties Summary 481
Table 149 StateMachine Class Methods Summary 490
Table 150 StateMachineOwner Class Properties Summary 505
Table 151 StateMachineOwner Class Methods Summary 506
Table 152 StateVertex Class Properties Summary 509
Table 153 StateVertex Class Methods Summary 511
Table 154 StateView Class Properties Summary 515
Table 155 StateView Class Methods Summary 515
Table 156 Subsystem Class Properties Summary 516
Table 157 Subsystem Class Methods Summary 519
Table 158 SubsystemView Class Properties Summary 530
Table 159 SubsystemView Class Methods Summary 531
Table 160 SwimLane Class Properties Summary 532
Table 161 SwimLane Class Methods Summary 532
Table 162 SwimLaneView Class Properties Summary 534
Table 163 SwimLaneView Class Methods Summary 534
Table 164 SyncItem Class Properties Summary 535
Table 165 SyncItem Class Methods Summary 535
Table 166 SyncItemView Class Properties Summary 536
Table 167 SyncItemView Class Methods Summary 537
Table 168 Transition Class Properties Summary 538
Table 169 Transition Class Methods Summary 538

Table 170 UseCase Class Properties Summary 543
Table 171 UseCase Class Methods Summary 548

Preface

The *Rational Rose 2000e, Rose Extensibility Reference* provides complete descriptions and syntax for all of the classes, properties and methods that comprise the Rational Rose Extensibility Interface (REI) model.

The REI extensions work in conjunction with BasicScript, allowing you to create Rational Rose Scripts. You can also use the REI extensions in an automation environment (controlling Rational Rose from within your normal development environment). See the online help or the *Rational Rose 2000e, Rose Extensibility User's Guide* for more information.

Prerequisites

This manual assumes that you are familiar with the Windows 95 Windows 98 or Windows NT 4.0 operating environment, object oriented design concepts, and how to use Rational Rose.

If you are unfamiliar with Rational Rose or object oriented design concepts, you should refer to the *Rational Rose 2000e, Using Rose* manual, as well as run the Rational Rose tutorial, which is included on your product CD.

Also note that you may need to adapt the syntax listed for each REI property and method to your particular programming language. If the listed syntax, does not meet your needs, consult your programming environment's help, programming language books, and outside documentation on the subject.

How This Manual Is Organized

The content of this manual is organized in alphabetical order, first by class and then by property or method within each class. So, for example, if you want to look up a property for the add-in class, you know that you should look very close to the beginning of the manual. Conversely, properties and methods for the Use-Case class are last in the manual.

For each REI class you will find, in order:

- A brief explanation of the class and its use
- A list of all properties defined for the class, including inherited properties
- A detailed description of each non-inherited property in the property list
- A list of all methods defined for the class, including inherited methods
- A detailed description of each non-inherited method in the method list

Note: *The detailed descriptions of inherited properties and methods are not repeated for each class. For detailed information on inherited properties and methods, refer to the class from which they are inherited.*

As with most reference documents, you can use the guide words at the top of each page to find your place.

At the end of the Reference section, you will find an appendix, which shows a series of screen shots of the Logical View of the Rational Rose Extensibility Model. You may find it helpful to see the packages that comprise the model and the classes contained in each package.

Online Help

Rational Rose includes comprehensive online help with hypertext links and a two-level index. The online help includes all of the information found in this manual, as well as all of the information contained in the *Rational Rose 2000e, Rose Extensibility User's Guide.*

Online Manual

Rational Rose includes all the user manuals online. Please refer to the Readme.txt file (found in the Rational Rose installation directory) for more information.

Related Documentation

Please review any **`readme.txt`** files and **`Release Notes`** to ensure that you have the latest information about the product before you begin using Rational Rose and the Extensibility Interface. The release notes are included with your product documentation and are available online from the **Start** menu. The release notes also list the new and updated classes, properties, and methods. This information allows existing users to quickly discover what has changed since the last version of Rose.

For additional resources, refer to the *Rational Rose 2000e, Using Rose* guide and online help. If you are new to Rational Rose, visual modeling, or the Unified Modeling Language (UML), you may also want to read the book, *Visual Modeling with Rational Rose and UML*, included with your product documentation.

File Names

Where file names appear in examples, Windows syntax is depicted. To obtain a legal UNIX file name, eliminate any drive prefix and change the backslashes to slashes:

```
c:\project\username
```

becomes

```
/project/username
```

Chapter 1

Rational Rose Extensibility Interface Reference

AbstractState Class

The AbstractState class is an abstract class that exposes properties and methods common to state and activity functionality in the extensibility interface. These common properties and methods involve actions, events, state machines, and substates. With the properties and methods of the AbstractState class, you can:

- Retrieve information about abstract states (states, activities), such as name, parent application, parent model, documentation, stereotypes, external documents, and state machine owners, parent state or activity, parent state machine, transitions
- Retrieve objects associated with abstract states (states, activities) such as child activities, decisions, states, and synchronizations
- Create and retrieve tool and property settings for abstract states (states, activities)
- Add and delete external documents
- Open specification sheets for abstract states (states, activities)
- Add, delete, and retrieve do, entry, and exit actions
- Add, delete, and retrieve state machines
- Add, delete, and retrieve user-defined events

The AbstractState class does not directly correspond to anything in the Rose user interface. Through inheritance, however, states and activities are abstract states.

AbstractState Class Properties

The following table describes the AbstractState Class properties.

Table 1 AbstractState Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
StateVertex Properties	Inherits all StateVertex class properties
SubActivities	Specifies the collection of child activities belonging to the abstract state
SubDecisions	Specifies the collection of child decisions belonging to the abstract state
SubStates	Specifies the collection of child states belonging to the abstract state
SubSynchronizations	Specifies the collection of child synchronizations belonging to the abstract state

AbstractState.SubActivities Property

Description

This property specifies the collection of child activities nested within the abstract state. The subactivity collection includes only immediate child activities; it does not include grandchildren or any other descendants. To retrieve all subactivities, use the AbstractState.GetAllSubDecisions method.

This property is read-only.

Syntax

`Set` *`myActivityCollection`* = *`myAbstractState`*.**`SubActivities`**

Property Type

ActivityCollection

AbstractState.SubDecisions Property

Description

This property specifies the collection of child decisions nested within the abstract state. The subdecision collection includes only immediate child decisions; it does not include grandchildren or any other descendants. To retrieve all subdecisions, use the AbstractState.GetAllSubDecisions method.

Note: *This property is read-only.*

Syntax

```
Set myDecisionCollection = myAbstractState.SubDecisions
```

Property Type

DecisionCollection

AbstractState.SubStates Property

Description

This property specifies the collection of child states nested within the abstract state. The substate collection includes only immediate child states; it does not include grandchildren or any other descendants. To retrieve all substates, the AbstractState.GetAllSubDecisions method. The collection specified by this property also includes initial and final states. To exclude initial or final states, add code to check the State.StateKind property of each state in the collection before working with the state.

Note: *This property is read-only.*

Syntax

```
Set myStateCollection = myAbstractState.SubStates
```

Property Type

StateCollection

AbstractState.SubSynchronizations Property

Description

This property specifies the collection of child synchronizations nested within the abstract state. The subsynchronization collection includes only immediate child synchronizations; it does not include grandchildren or any other descendants. To retrieve all subsynchronizations, use the AbstractState.GetAllSubSynchronizations method.

Note: *This property is read-only.*

Syntax

```
Set mySyncItemCollection =
      myAbstractState.SubSynchronizations
```

Property Type

SyncItemCollection

AbstractState Class Methods

The following table describes the AbstractState Class methods.

Table 2 AbstractState Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
StateVertex Methods	Inherits all StateVertex class methods
AddDoAction	Adds a do action
AddEntryAction	Adds an entry action
AddExitAction	Adds an exit action
AddStateMachine	Adds a state machine
AddUserDefinedEvent	Adds a user-defined event
DeleteAction	Deletes an action
DeleteStateMachine	Deletes a state machine
DeleteUserDefinedEvent	Deletes a user-defined event
GetActions	Retrieves actions
GetAllSubActivities	Recursively retrieves all subactivities
GetAllSubDecisions	Recursively retrieves all subdecisions
GetAllSubStates	Recursively retrieves all substates
GetAllSubSynchronizations	Recursively retrieves all subsynchronizations
GetDoActions	Retrieves all do actions
GetEntryActions	Retrieves all entry actions
GetExitActions	Retrieves all exit actions
GetStateMachines	Retrieves all state machines
GetUserDefinedEvents	Retrieves all user-defined events

AbstractState.AddDoAction Method

Description

This method adds a do action to an abstract state.

Syntax

`Set` *`theAction`* `=` *`myAbstractState`*`.`**`AddDoAction`** `(`*`theActionName`*`)`

Element	**Description**
theAction As Action	Returns the do action added to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) to which to add the do action
theActionName As String	Name of the do action to be added to the abstract state

AbstractState.AddEntryAction Method

Description

This method adds an entry action to an abstract state.

Syntax

`Set` *`theAction`* `=` *`myAbstractState`*`.`**`AddEntryAction`** `(`*`theActionName`*`)`

Element	**Description**
theAction As Action	Returns the entry action added to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) to which to add the entry action
theActionName As String	Name of the entry action to be added to the abstract state

AbstractState.AddExitAction Method

Description

This method adds an exit action to an abstract state.

Syntax

Set *theAction* = *myAbstractState*.**AddExitAction**
    (*theActionName*)

Element	Description
theAction As Action	Returns the exit action added to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) to which to add the exit action
theActionName As String	Name of the exit action to be added to the abstract state

AbstractState.AddStateMachine Method

Description

This method adds a state machine to an abstract state. For another way to add a state machine, see the StateMachineOwner.CreateStateMachine method.

Syntax

Set *theStateMachine* = *myAbstractState*.**AddStateMachine**
    (*theName*)

Element	Description
theStateMachine As StateMachine	Returns the state machine added to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) to which to add the state machine
theName As String	Name of the state machine to be added to the abstract state

AbstractState.AddUserDefinedEvent Method

Description

This method adds a user-defined event to an abstract state.

Syntax

```
Set theEvent = myAbstractState.AddUserDefinedEvent
      (theEventName, theActionName)
```

Element	Description
theEvent As Event	Returns the event added to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) to which to add the event
heEventName As String	Name of the event to be added to the abstract state
theActionName As String	Name of the action to be added to the abstract state

AbstractState.DeleteAction Method

Description

This method deletes an action from an abstract state.

Syntax

```
isActionDeleted = myAbstractState.DeleteAction (theAction)
```

Element	Description
isActionDeleted As Boolean	Returns True if the action is successfully deleted from the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to delete the action
theAction As Action	Action to be deleted from the abstract state

AbstractState.DeleteStateMachine Method

Description

This method deletes a state machine from an abstract state. For other ways to delete state machines, see the Class.DeleteStateMachine and StateMachineOwner.DeleteStateMachine methods.

Syntax

```
isStateMachineDeleted = myAbstractState.DeleteStateMachine
      (theStateMachine)
```

Element	Description
isStateMachineDeleted As Boolean	Returns True if the state machine is successfully deleted from the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to delete the state machine
theStateMachine As StateMachine	State machine to be deleted from the abstract state

AbstractState.DeleteUserDefinedEvent Method

Description

This method deletes a user-defined event from an abstract state.

Syntax

```
isUserDefinedEventDeleted =
      myAbstractState.DeleteUserDefinedEvent (theEvent)
```

Element	Description
isUserDefinedEventDeleted As Boolean	Returns True if the event is successfully deleted from the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to delete the event
theEvent As Event	Event to be deleted from the abstract state

AbstractState.GetActions Method

Description

This method retrieves the collection of actions belonging to an abstract state. To retrieve specific actions, see the AbstractState.GetDoActions, AbstractState.GetEntryActions and AbstractState.GetExitActions methods.

Syntax

```
Set theActionCollection = myAbstractState.GetActions ()
```

Element	Description
theActionCollection As ActionCollection	Returns the collection of actions belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of actions

AbstractState.GetAllSubActivities Method

Description

This method recursively retrieves the collection of activities belonging to an abstract state.

Syntax

```
Set theActivityCollection =
      myAbstractState.GetAllSubActivities ()
```

Element	Description
theActivityCollection As ActivityCollection	Returns the collection of activities belonging to the abstract state and any substates or subactivities
myAbstractState As AbstractState	Abstract state (state or activity) from which to recursively retrieve the collection of activities

AbstractState.GetAllSubDecisions Method

Description

This method recursively retrieves the collection of decisions belonging to an abstract state.

Syntax

```
Set theDecisionCollection =
      myAbstractState.GetAllSubDecisions ()
```

Element	Description
theDecisionCollection As DecisionCollection	Returns the collection of decisions belonging to the abstract state and any substates or subactivities
myAbstractState As AbstractState	Abstract state (state or activity) from which to recursively retrieve the collection of decisions

AbstractState.GetAllSubStates Method

Description

This method recursively retrieves the collection of substates belonging to an abstract state.

Syntax

```
Set theStateCollection = myAbstractState.GetAllSubStates ()
```

Element	Description
theStateCollection As StateCollection	Returns the collection of substates belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of substates

AbstractState.GetAllSubSynchronizations Method

Description

This method recursively retrieves the collection of synchronizations belonging to an abstract state.

Syntax

```
Set theSyncItemCollection =
      myAbstractState.GetAllSubSynchronizations ()
```

Element	Description
theSyncItemCollection As SyncItemCollection	Returns the collection of synchronizations belonging to the abstract state and any substates or subactivities
myAbstractState As AbstractState	Abstract state (state or activity) from which to recursively retrieve the collection of synchronizations

AbstractState.GetDoActions Method

Description

This method retrieves the collection of do actions belonging to an abstract state. To retrieve all types of actions, see the AbstractState.GetActions method.

Syntax

```
Set theActionCollection = myAbstractState.GetDoActions ()
```

Element	Description
theActionCollection As ActionCollection	Returns the collection of do actions belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of do actions

AbstractState.GetEntryActions Method

Description

This method retrieves the collection of entry actions belonging to an abstract state. To retrieve all types of actions, see the AbstractState.GetActions method.

Syntax

```
Set theActionCollection = myAbstractState.GetEntryActions
       ()
```

Element	Description
theActionCollection As ActionCollection	Returns the collection of entry actions belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of entry actions

AbstractState.GetExitActions Method

Description

This method retrieves the collection of exit actions belonging to an abstract state. To retrieve all tpes of actions, see the AbstractState.GetActions method.

Syntax

```
Set theActionCollection = myAbstractState.GetExitActions ()
```

Element	Description
theActionCollection As ActionCollection	Returns the collection of exit actions belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of exit actions

AbstractState.GetStateMachines Method

Description

This method retrieves the collection of state machines belonging to an abstract state.

Syntax

```
Set theStateMachineCollection =
      myAbstractState.GetStateMachines ()
```

Element	Description
theStateMachineCollection As StateMachineCollection	Returns the collection of state machines belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of state machines

AbstractState.GetUserDefinedEvents Method

Description

This method retrieves the collection of user-defined events which have actions belonging to an abstract state. To retrieve the collection of actions for an event, use the AbstractState.GetActions method.

Syntax

```
Set theEventCollection =
      myAbstractState.GetUserDefinedEvents ()
```

Element	Description
theEventCollection As EventCollection	Returns the collection of events belonging to the abstract state
myAbstractState As AbstractState	Abstract state (state or activity) from which to retrieve the collection of events

Action Class

An action is an operation that:

- Is associated with a transition
- Takes an insignificant amount of time to complete
- Is considered to be non-interruptible

Action Class Properties

The following table summarizes the Action Class properties:

Table 3 Action Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
RoseItem Properties	Inherits all RoseItem Class properties
Arguments	Defines the content of the action
Target	Specifies the object that is the target of the action

Action.Arguments Property

Description

Specifies the arguments for the action.

Syntax

Action.`Arguments`

Property Type

String

Action.Target Property

Description

Specifies the object that is the target for the action; for example, the object to receive a message.

Syntax

Action.`Target`

Property Type

String

Action Class Methods

The following table summarizes the Action Class methods.

Table 4 Action Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
RoseItem Methods	Inherits all RoseItem class methods

Activity Class

The Activity class is an abstract class that exposes Rose's activity functionality in the extensibility interface. With the properties and methods of the Activity class, you can:

- Retrieve information about activities, such as name, documentation, stereotype
- Retrieve objects associated with activities such as parent activities, parent states, parent state machines, child activities, child decisions, child states, child synchronizations, outgoing transitions, and swimlanes
- Create and retrieve tool and property settings for activities
- Open specification sheets for activities
- Add, delete, and retrieve an activity's actions, state machines, and events
- Add and delete transitions

The Activity class corresponds to activities in the Rose user interface.

Activity Class Properties

The following table describes the Activity Class properties.

Table 5 Activity Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
StateVertex Properties	Inherits all StateVertex class properties
AbstractState Properties	Inherits all AbstractState class properties

Activity Class Methods

The following table describes the Activity Class methods.

Table 6 Activity Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
StateVertex Methods	Inherits all StateVertex class methods
AbstractState Methods	Inherits all AbstractState class methods

ActivityView Class

The ActivityView class is an abstract class that exposes Rose's activity view functionality in the extensibility interface. With the properties and methods of the ActivityView class, you can:

- Retrieve information about the activity represented by the activity view, including the activity object
- Retrieve objects associated with the activity view such as the diagram it is on, any parent or child views, and line vertices
- Retrieve physical information about the activity view such as position, height, width, fill color, line color, font
- Create and retrieve tool and property settings for activity views

The ActivityView class corresponds to activities on diagrams in the Rose user interface.

ActivityView Class Properties

The following table describes the ActivityView Class properties.

Table 7 ActivityView Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItemView Properties	Inherits all RoseItemView class properties

ActivityView Class Methods

The following table describes the ActivityView Class methods.

Table 8 ActivityView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItemView Methods	Inherits all RoseItemView class methods
GetActivity	Returns the activity object represented by the activity view object

ActivityView.GetActivity Method

Description

This method retrieves the activity represented by the activity view.

Syntax

```
Set theActivity = myActivityView.GetActivity ()
```

Element	**Description**
theActivity As Activity	Returns the activity object represented by the activity view object
myActivityView As ActivityView	Activity view object from which to retrieve the activity object

Application Class

Use the application class to:

- Create a new model
- Select an existing model as the current model
- Determine the characteristics of the Rational Rose application being controlled by your script

Here are a few of the application characteristics you can control with application class properties and methods:

- How (and if) the Rational Rose application appears on the computer screen while the script is running
- The size and position of the Rational Rose application window
- Whether to write errors to the error log

Application Class Properties

The following table summarizes the Application Class properties:

Table 9 Application Class Properties Summary

Property	**Description**
AddInManager	Rational Rose AddInManager belonging to the currently active Rational Rose executable
ApplicationPath	Full path and name of the currently active Rational Rose executable. You can retrieve this value, but you cannot set it
CommandLine	String that is passed to the Rational Rose application when the Rational Rose executable is run
CurrentModel	Currently open Rational Rose model
Height	Height of the main window
IsInitialized	Indicates whether the application is fully initialized
Left	Distance between the left side of the main window and the left side of the screen
PathMap	Path map for the currently active Rational Rose application

Property	**Description**
ProductName	Name of the currently active Rational Rose application
Top	Distance between the top of the main window and the top of the screen
Version	Version of the currently active Rational Rose product
Visible	Controls whether the Rational Rose application is visible on the screen
Width	Width of the main window

Application.AddInManager Property

Description

Specifies the Rational Rose AddIn Manager belonging to the currently active Rational Rose executable.

Note: *This property is read-only.*

Syntax

`Application.AddInManager`

Property Type

RoseAddInManager

Application.ApplicationPath Property

Description

Specifies the path to the Rational Rose application to execute.

Note: *This property is read-only.*

Syntax

`Application`.**`ApplicationPath`**

Property Type

String

Example

```
Application.c:\Programs\Rational\Rose.exe.ApplicationPath
```

Application.CommandLine Property

Description

Returns the command line option string that is passed when the Rational Rose executable is run.

Note: *This property is read-only.*

Syntax

`Application`.**`CommandLine`**

Property Type

String

Example

```
Application.“c:\My Models\My Model 1.mdl”
```

or

```
Application.c:\MyModels\Mymodel1.mdl
```

Note: *Enclose the string with quotation marks if it contains any spaces.*

Application.CurrentModel Property

Description

Specifies the model that is currently open in Rational Rose.

Note: *This property is read-only.*

Syntax

`Application`.**`CurrentModel`**

Property Type

Application

Application.Height Property

Description

Specifies the height of the main window.

Syntax

`Application`.**`Height`**

Property Type

Integer

Application.IsInitialized Property

Description

This property indicates whether the specified Rose application is fully initialized.

Note: *This property is read-only.*

Syntax

```
blnIsInitialized = objApplication.IsInitialized
```

Property Type

Boolean

Application.Left Property

Description

Specifies the distance between the left side of the main window and the left side of the screen.

Syntax

`Application``.`**`Left`**

Property Type

Integer

Application.PathMap Property

Description

Returns the path map defined for the current Rational Rose application.

Note: *This property is read-only.*

Syntax

`Application``.`**`PathMap`**

Property Type

PathMap

Application.ProductName Property

Description

Returns the product name for the currently active Rational Rose application.

Note: *This property is read-only.*

Syntax

`Application`.**`ProductName`**

Property Type

String

Application.Top Property

Description

Specifies the distance between the top of the main window and top of the screen.

Syntax

`Application`.**`Top`**

Property Type

Integer

Application.Version Property

Description

Returns the version of the currently active Rational Rose application. Corresponds to the information provided when you select About from the Help menu in Rational Rose.

Note: *This property is read-only.*

Syntax

Application.**Version**

Property Type

String

Application.Visible Property

Description

Controls whether the Rational Rose application is visible on the computer screen.

Syntax

Application.**Visible**

Property Type

Boolean

Application.Width Property

Description

Specifies the width of the main window.

Syntax

`Application``.`**`Width`**

Property Type

Integer

Application Class Methods

The following table summarizes the Application Class Methods.

Table 10 Application Class Methods Summary

Method	Description
CompileScriptFile	Compiles a script file
ExecuteScript	Executes a source or compiled image of a script
Exit	Exits the Rational Rose application
FreeScript	Unloads a previously loaded script from memory
GetLicensedApplication	Retrieves an instance of the application given the application's licensing key
GetObject	Retrieves the OLE interface object associated with the application
GetProfileString	Retrieves the string associated with an entry in the Rose.ini file
GetRoseIniPath	Retrieves the path to the Rose.ini file for the current user
LoadScript	Loads a script into memory so that it can be called by other scripts
NewModel	Creates a new Rational Rose model

Method	Description
NewScript	Opens a blank window for script creation in the script editor
OpenExternalDocument	Opens an external document given its file name
OpenModel	Opens a Rational Rose model and all of its subunits
OpenModelAsTemplate	Opens an existing model to use as a template for creating a new one
OpenRoseModel	Opens a Rational Rose model with or without prompting the user whether to open all of its subunits
OpenScript	Opens an existing script in the script editor window
OpenURL	Opens an internet document, given its URL
Save	Saves the current Rational Rose model
SaveAs	Saves the current Rational Rose model under a new name
WriteErrorLog	Writes a message to the log window
WriteProfileString	Writes an entry and its associated string in the Rose.ini file

Application.CompileScriptFile Method

Description

This subroutine compiles the script contained in the specified file.

Syntax

```
theApplication.CompileScriptFile theFileName,
       theBinaryName, Debug
```

Element	**Description**
theApplication As Application	Instance of the Rational Rose application in which the script is being compiled
theFileName As String	Name of the file that contains the script being compiled; include the.ebs file extension
theBinaryName As String	Name of the binary file in which to save the compiled script; use the.ebx file extension
Debug As Boolean	Set to True to embed the script's source code in the compiled file. This allows the script debugger to display the source code when it enters external modules

Application.ExecuteScript Method

Description

This subroutine executes the source or compiled image of a script contained the specified file. You can specify the file without its extension. If the script is currently open in the script editor, Rational Rose will execute the open script. Otherwise, Rational Rose will search for the source script (.ebs) and execute it, if found. If not found, Rational Rose will search for and execute the compiled script (.ebx file).

Syntax

theApplication.**ExecuteScript** *theFileName*

Element	Description
theApplication As Application	Instance of the Rational Rose application in which the script is being executed
theFileName As String	Name of the file that contains the script to execute

Application.Exit Method

Description

This subroutine exits the Rational Rose application.

Syntax

theApplication.**Exit**

Element	Description
theApplication As Application	Instance of the Rational Rose application being exited

Application.FreeScript Method

Description

This subroutine unloads the source or compiled image of a script contained in the specified file. Specify the file without its extension and Rational Rose will free the source script (.ebs), if found. If not found, Rational Rose will free the compiled script (.ebx file).

Notes:

- This subroutine is only valid for Rational Rose Script; it does not exist in Rational Rose Automation
- Every LoadScript call should have a subsequent FreeScript call. See LoadScript Method for more information.

Syntax

theApplication.**FreeScript** *theFileName*

Element	**Description**
theApplication As Application	Instance of the Rational Rose from which the script is being unloaded
theFileName As String	The name of the file that contains script to unload. Do not specify a file extension

Application.GetLicensedApplication Method

Description

This method retrieves an instance of the licensed application given the application's licensing key.

Syntax

```
Set theInstance = theApplication.GetLicensedApplication
      (theKey)
```

Element	Description
theInstance As Application	Returns the instance of the licensed application
theApplication As Application	Currently active application
theKey As String	Licensing key for the application being retrieved

Application.GetObject Method

Description

This method retrieves the OLE automation interface object associated with the specified application.

Note: *This method is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

```
Set theOLEObject = theApplication.GetObject ()
```

Element	Description
theOLEObject As Object	Returns the OLE automation interface object associated with the application
theApplication As Application	Instance of the Rational Rose application whose OLE automation interface object is being returned

Application.GetProfileString Method

Description

This method retrieves a profile string entry in the Rose.ini file, given a section, entry, and default value.

Syntax

```
Set theProfileString = theApplication.GetProfileString
      (theSection, theEntry, theDefault)
```

Element	Description
theProfileString As String	Returns the profile string that corresponds to the given section, entry, and default value
theApplication As Application	Currently active application and therefore the application whose Rose.ini file entry is being retrieved
theSection As String	Name of the Rose.ini file section from which the profile string is being retrieved. For example, [PathMap]
theEntry As String	The name of the Rose.ini file entry whose profile string is being retrieved For example, $SCRIPT_PATH
theDefault As String	Default value of the entry being retrieved. In the [PathMap] $SCRIPT_PATH example, the default value is the path to the folder that contains the scripts being called by the application

Application.GetRoseIniPath Method

Description

This method retrieves the path to the `Rose.ini` file for the current user.

Syntax

`theIniPathString` = *`theApplication`*.**`GetRoseIniPath`** `()`

Element	Description
theIniPathString As String	Returns a string containing the path to and filename of the `Rose.ini` file for the current user For example, C:\Program Files\Rational \Rose 2000e\Rose.ini
theApplication As Application	Currently active application and therefore the application whose Rose.ini file path is being retrieved

Application.LoadScript Method

Description

This subroutine loads the source or compiled image of a script contained in the specified file. You can specify the file without its extension and Rational Rose will load the source script (.ebs), if found. If not found, Rational Rose will load the compiled script (.ebx file).

Notes:

- This subroutine is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.
- When finished with the script, you should make a call to FreeScript. Because scripts contain reference counting information, if you call LoadScript on a given script 10 times, you should subsequently call FreeScript 10 times; otherwise the script will not be unloaded.

Syntax

theApplication.**LoadScript** *theFileName*

Element	**Description**
theApplication As Application	Instance of the Rational Rose application in which the script is being loaded
theFileName As String	Name of the file that contains the script; Do not specify a file extension

Application.NewModel Method

Description

This method creates a new Rational Rose model and returns it as a model object.

Syntax

`Set` *`theModel`* `=` *`theApplication`*.**`NewModel`** `()`

Element	Description
theModel As Model	Contains the newly created Rational Rose model
theApplication As Application	Instance of the Rational Rose application in which the model is being created

Application.NewScript Method

Description

This subroutine opens a script editor window in which to create a new script.

Note: *This subroutine is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

`theApplication`.**`NewScript`**

Element	Description
theApplication As Application	Instance of the Rational Rose application in which the new script is being created

Application.OpenExternalDocument Method

Description

This method opens an external document, given a fully qualified name of the file that contains the document.

Syntax

`IsOpen` = *`theApplication`*.**`Open`** (*`theFileName`*)

Element	Description
IsOpen As Boolean	Returns a value of true when the specified document is successfully opened
theApplication As Application	Currently active application
theFileName As String	Fully qualified file name or the URL that contains the external document

Application.OpenModel Method

Description

This method opens a Rational Rose model and returns it as a model object.

Note: *Use the more flexible OpenRoseModel method instead of OpenModel.*

Syntax

Set *theModel* = *theApplication*.**OpenModel** (*theName*)

Element	Description
theModel As Model	Contains the model being opened
theApplication As Application	Instance of the Rational Rose application from which the model is being retrieved
theName As String	Name of the model being opened including path For example, To open theModel.mdl in C:/theSubdir, type: Set theModel=theApplication.OpenModel ("c:/theSubdir/theModel.mdl")

Application.OpenModelAsTemplate Method

Description

This method retrieves an existing model to be used as a template from which to create a new model.

Syntax

```
Set theModel = theApplication.OpenModelAsTemplate
      (FileName)
```

Element	Description
theModel As Model	Returns the model contained in the specified file
theApplication As Application	Currently active application
theFileName As String	Name of the file that contains the model being returned

Application.OpenRoseModel Method

Description

This method opens a Rational Rose model with or without prompting the user whether to open all of its subunits.

Note: *Use this method instead of the OpenModel method.*

Syntax

```
Set theModel = theApplication.OpenRoseModel (theModelPath,
        promptForSubunits)
```

Element	**Description**
theModel As Model	Returns the model contained in the specified path and file
theApplication *As* Application	Currently active application
theModelPath As String	Path and filename of an existing Rational Rose model to be loaded For example, “C:\My Models\myModel.mdl”
promptForSubunits As Boolean	Set this argument to True to load the Model and, if it has subunits, display the “Load Subunits” prompt. This gives the user a choice as to whether to load the subunits. Set this argument to False to load the model and all of its subunits without prompting the user.

Application.OpenScript Method

Description

This subroutine opens the source or compiled image of a script contained in the specified file in the script editor window. You can specify the file without its extension and Rational Rose will search for the source script (.ebs) and open it, if found. If not found, Rational Rose will search for and open the compiled script (.ebx file).

Note: *This subroutine is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

theApplication.**OpenScript** *FileName*

Element	Description
theApplication As Application	Instance of the Rational Rose application in which the script is being opened
FileName As String	Name of the script file being opened

Application.OpenURL Method

Description

This method opens a URL, given the URL string.

Syntax

IsOpen = *theApplication*.**Open** (*theURL*)

Element	Description
IsOpen As Boolean	Returns a value of true when the specified URL is successfully opened
theApplication As Application	Currently active application
theURL As String	URL that contains the external document

Application.Save Method

Description

This subroutine saves the current Rational Rose model.

Note: *This method is not valid if any of the following is true:*

The file containing the Rational Rose model is ReadOnly.

The file containing the Rational Rose model is unnamed.

SaveUnits is True and any Unit cannot be saved.

Syntax

`theApplication`.**`Save`** *`SaveUnits`*

Element	**Description**
theApplication As Application	Instance of the Rational Rose application whose current model is being saved
SaveUnits As Boolean	Indicates whether the current model is comprised of controlled units

Application.SaveAs Method

Description

This subroutine names and saves the current Rational Rose model.

Note: *This method is not valid under the following conditions:*

The file containing the Rational Rose model is ReadOnly.

The file containing the Rational Rose model is unnamed.

SaveUnits is True and any Unit cannot be saved.

Syntax

`theApplication.SaveAs theName, SaveUnits`

Element	Description
theApplication As Application	Instance of the Rational Rose application whose current model is being saved
theName As String	Name of the model being saved
SaveUnits As Boolean	Indicates whether the current model is comprised of controlled units

Application.WriteErrorLog Method

Description

This subroutine writes an error message to a log window.

Syntax

`theApplication.WriteErrorLog theMessage`

Element	Description
theApplication As Application	Instance of the Rational Rose application for which errors are being logged
theMessage As String	Message text to write to the error log window

Application.WriteProfileString Method

Description

This method retrieves a profile string entry in the Rose.ini file, given a section, entry, and default value.

Syntax

```
IsWritten = theApplication.WriteProfileString (theSection,
      theEntry, theValue)
```

Element	Description
IsWritten As Boolean	Returns a value of true when the specified ProfileString is successfully written to the Rose.ini file.
theApplication As Application	Currently active application and therefore the application whose Rose.ini file entry is being written.
theSection As String	Name of the Rose.ini file section to which the profile string is being written. For example, [PathMap]
theEntry As String	The name of the Rose.ini file entry whose profile string is being written. For example, $SCRIPT_PATH
theValue As String	Value of the entry being written. In the [PathMap] $SCRIPT_PATH example, the value is the actual path to the folder that contains the scripts being called by the application.

Association Class

An association is a connection, or a link, between classes. The association class exposes properties and methods that:

- Determine the characteristics of associations between classes
- Allow you to retrieve associations from a model

Association Class Properties

The following table summarizes the Association Class properties:

Table 11 Association Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Constraints	Specifies the association’s constraints
Derived	Identifies the association as derived
LinkClass	Identifies the association as a link class
ParentCategory	Specifies the package containing an association
Role1	Specifies a role as Role 1 in an association
Role2	Specifies a role as Role 2 in an association
Roles	Specifies the Role collection belonging to the association

Association.Constraints Property

Description

This property specifies any constraints (expressions of semantic conditions that must be preserved) on the association relationship.

Syntax

theConstraintText = *theAssociation*.**Constraints**

theAssociation.**Constraints** = "Must have a lot of money if university = Harvard; IQ must be high for university = ..."

Property Type

String

Association.Derived Property

Description

Indicates whether this object is derived from another object.

Syntax

Object.**Derived**

Property Type

Boolean

Association.LinkClass Property

Description

Identifies the association as a link class.

Syntax

Association.**LinkClass**

Property Type

Class

Association.ParentCategory Property

Description

This property specifies the package containing the association.

Note: *This property is read-only.*

Syntax

`Set` *`myPackage`* = *`myAssociation`*.**`ParentCategory`**

Property Type

Category

Association.Role1 Property

Description

Specifies an object as being Role 1 in an association.

Note: *This property is read-only.*

Syntax

`Association`.**`Role1`**

Property Type

Role

Association.Role2 Property

Description

Specifies an object as being Role 2 in an association.

Note: *This property is read-only.*

Syntax

```
Association.Role2
```

Property Type

Role

Association.Roles Property

Description

Specifies the collection of Roles belonging to the Association.

Note: *This property is read-only.*

Syntax

```
Association.Roles
```

Property Type

RoleCollection

Association Class Methods

The following table summarizes the Association class methods.

Table 12 Association Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
ClearRoleForNameDirection	Removes the name direction from a role belonging to the association
GetCorrespondingRole	Retrieves a corresponding role from an association
GetOtherRole	Retrieves another role from an association
GetRoleForNameDirection	Retrieves the role that is set as the name direction for the association
NameIsDirectional	Determines whether the association has a name direction
SetLinkClassName	Specifies the class that is the link class for the association
SetRoleForNameDirection	Sets a role as the name direction of the association

Association.ClearRoleForNameDirection Method

Description

This subroutine clears name direction setting for the association.

Syntax

theAssociation.**ClearRoleForNameDirection**

Element	**Description**
theAssociation As Association	Association whose name direction is being cleared

Association.GetCorrespondingRole Method

Description

This method retrieves the role associated with a specified class.

Syntax

Set *theRole* = *theObject*.**GetCorrespondingRole** (*theClass*)

Element	Description
theRole As Role	Returns the role that corresponds to the specified class
theObject As Association	Association from which the role is being retrieved
theClass As Class	The Class whose role is being returned

Association.GetOtherRole Method

Description

This method retrieves the role of a class associated with the specified class.

Syntax

Set *theRole* = *theAssociation***GetOtherRole** (*theClass*)

Element	Description
theAssociation As Association	Association from which the role is being retrieved
theRole As Role	Returns the role that corresponds to the other class in the association, not the specified class
theClass As Class	Class whose associated class' role is being returned

Association.GetRoleForNameDirection Method

Description

This method retrieves the role that is set as the name direction for the association.

Syntax

`Set theRole = theAssociation.GetRoleForNameDirection ()`

Element	Description
theRole As Role	Returns the role that is set as the association's name direction
theAssociation As Association	Association from which the role is being retrieved

Association.NameIsDirectional Method

Description

This method checks whether the association has a name directional role setting.

Syntax

`IsDirectional = theAssociation.NameIsDirectional ()`

Element	Description
IsDirectional As Boolean	Returns a value of True if the association has a name directional setting
theAssociation As Association	Association whose name direction setting is being checked

Association.SetLinkClassName Method

Description

This subroutine sets the link class name for an association.

Syntax

`theObject`.**`SetLinkClassName`** *`theName`*

Element	Description
theObject As Association	Association whose link class name is being set
theName As String	Name of the link class

Association.SetRoleForNameDirection Method

Description

This subroutine sets the role that is the name direction for the association.

Syntax

`theAssociation`.**`SetRoleForNameDirection`** *`theRole`*

Element	Description
theAssociation As Association	Association whose name direction role is being set
theRole As Role	Role being set as the association’s name direction

Attribute Class

Attributes define the characteristics of a class. Each object in a class has the same attributes, but the values of the attributes may be different.

The attribute class exposes properties and methods that determine the characteristics of these attributes and that allow you to retrieve them from a model.

Some of the characteristics determined by attribute class properties are:

- Type
- Initial value
- whether the attribute is static; whether it is derived
- Attribute visibility

Attribute Class Properties

The following table summarizes the Attribute class properties.

Table 13 Attribute Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Containment	Indicates a containment relationship
Derived	Defines the attribute as derived
ExportControl	Controls attribute visibility
InitValue	Initial value of the attribute
ParentClass	Specifies the Class to which the attribute belongs
Static	Defines the attribute as static
Type	Type of the attribute

Attribute.Containment Property

Description

The Containment property is a rich data type. The following table describes the valid forms of expressing the Containment rich data type.

Table 14 Attribute.Containment Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Unspecified ■ ByValue ■ ByReference

Note: *This property is read-only.*

Syntax

`Attribute``.`**`Containment`**

Property Type

Containment

Attribute.Derived Property

Description

Indicates whether the attribute is derived.

Syntax

`Attribute``.`**`Derived`**

Property Type

Boolean

Attribute.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the attribute object. The following table describes the valid forms of expressing the ExportControl rich data type for the Attribute class.

Table 15 Attribute.ExportControl Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible string values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControlproperty is read-only. Its Value and Name, however, are read/write.*

Syntax

theAttribute.**ExportControl.Name** = **"PrivateAccess"**

theAttribute.**ExportControl.Value** = **2**

theNameStr = *theAttribute*.**ExportControl.Name**

theValue = *theAttribute*.**ExportControl.Value**

Property Type

RichType or AttributeExportControl

Attribute.InitValue Property

Description

Indicates the initial value of the attribute object.

Syntax

`Attribute``.`**`InitValue`**

Property Type

String

Attribute.ParentClass Property

Description

Specifies the class to which the attribute belongs.

Note: *This property is read-only.*

Syntax

`Attribute``.`**`ParentClass`**

Property Type

Class

Attribute.Static Property

Description

Indicates whether the attribute is static.

Syntax

`Attribute``.`**`Static`**

Property Type

Boolean

Attribute.Type Property

Description

Indicates the data type of the attribute object.

Syntax

`Attribute.Type`

Property Type

String

Attribute Class Methods

The following table summarizes the Attribute class methods.

Table 16 Attribute Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
Rose Item Methods	Inherits all RoseItem methods

Category Class

The category class allows you to define and manipulate logical collections of classes. The category class exposes properties and methods that allow you to define and manipulate the characteristics of categories.

Category Class Properties

The following table summarizes the Category Class properties.

Table 17 Category Class Properties Summary

Property	**Description**
Element	Inherits all Element properties
RoseItem	Inherits all RoseItem properties
Associations	Collection that contains all associations belonging to the category
Categories	Collection that contains the categories that are children of the category
ClassDiagrams	Collection that contains the class diagrams belonging to the category
Classes	Collection that contains the classes belonging to the category
Global	Identifies the category as global
ParentCategory	Category that contains the category.
ScenarioDiagrams	Collection that contains the scenario diagrams belonging to the category
UseCases	Collection that contains the use cases belonging to the category

Category.Associations Property

Description

Specifies the associations that belong to the category.

Note: *This property is read-only.*

Syntax

`Category`.**`Associations`**

Property Type

AssociationsCollection

Category.Categories Property

Description

Specifies the categories that belong to the category.

Note: *This property is read-only.*

Syntax

`Category`.**`Categories`**

Property Type

CategoryCollection

Category.ClassDiagrams Property

Description

Specifies the class diagrams that belong to the category.

Note: *This property is read-only.*

Syntax

```
Category.ClassDiagrams
```

Property Type

ClassDiagramCollection

Category.Classes Property

Description

This property specifies the classes that belong to the category. This property does not, however, specify the classes that belong to child categories of the specified category. If you want to know all the classes belonging to a particular category and all of its child categories, please use the GetAllClasses method.

Note: *This property is read-only.*

Syntax

```
Set theClassCollection = theCategory.Classes
```

Property Type

ClassCollection

Category.Global Property

Description

Indicates whether the category is a global object.

Syntax

`Category``.`**`Global`**

Property Type

Boolean

Category.ParentCategory Property

Description

If the category is the root category, then the value of the parent category will be set to *Nothing.*

You can check its value by using the TopLevel method. You cannot set this value.

Note: *This property is read-only.*

Syntax

`Category``.`**`ParentCategory`**

Property Type

Category

Category.ScenarioDiagrams Property

Description

Specifies the scenario diagrams that belong to the category.

Note: *This property is read-only.*

Syntax

`Category`.**`ScenarioDiagrams`**

Property Type

ScenarioDiagramCollection

Category.UseCases Property

Description

Specifies the use cases that belong to the category.

Note: *This property is read-only.*

Syntax

`Category`.**`UseCases`**

Property Type

UseCaseCollection

Category Class Methods

The following table summarizes the Category Class methods.

Table 18   Category Class Methods Summary

Element	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
ControllableUnit Methods	Inherits all ControllableUnit class methods
Package Methods	Inherits all Package class methods
AddCategory	Adds a category to a category
AddCategoryDependency	Adds a category dependency to a category
AddClass	Adds a class to a category
AddClassDiagram	Adds a class diagram to a category
AddDataModelDiagram	Adds a data model diagram to a category
AddScenarioDiagram	Adds a scenario diagram to a category
AddUseCase	Adds a use case to a category
DeleteCategory	Deletes a category from a category
DeleteCategoryDependency	Deletes a category dependency from a category
DeleteClass	Deletes a class from a category
DeleteClassDiagram	Deletes a class diagram from a category
DeleteScenarioDiagram	Deletes a scenario diagram from a category
DeleteUseCase	Deletes a use case from a category
GetAllCategories	Retrieves all of the categories belonging to the category and all of the categories belonging to its children
GetAllClasses	Retrieves the classes belonging to the category and all of the classes belonging to all child categories of the category

Element	Description
GetAllUseCases	Retrieves the collection that contains the use cases belonging to the category
GetAssignedSubsystem	Retrieves the subsystem assigned to the category
GetCategoryDependencies	Retrieves the collection of category dependencies belonging to the category
HasAssignedSubsystem	Retrieves the subsystem assigned to the category
RelocateAssociation	Relocates or moves an association from its current package to the specified package
RelocateCategory	Relocates a category within a category
RelocateClass	Relocates a class within a category
RelocateClassDiagram	Relocates a class diagram within a category
RelocateScenarioDiagram	Relocates a scenario diagram within a category
SetAssignedSubsystem	Assigns the subsystem to the category
TopLevel	Indicates whether the category is the root category

Category.AddCategory Method

Description

This method creates a new category in a model and returns it in the specified *object.*

Syntax

```
Set theCategory = theCategory.AddCategory (theName)
```

Element	Description
theCategory As Category	Returns the newly created Category object
theName As String	Name of the category being created, expressed as a string
theCategory As Category	Instance of the object being created and added

Category.AddCategoryDependency Method

Description

This method adds a category dependency to a category.

Syntax

```
Set theCategoryDependency =
        theCategory.AddCategoryDependency (theName,
        theSupplierCategoryName)
```

Element	Description
theCategoryDependency As CategoryDependency	Returns the category dependency being added to the category
theCategory As Category	Category to which the dependency is being added
theName As String	Name of the category dependency to add
theSupplierCategoryName As String	Name of the Supplier Category of the category dependency

Category.AddClass Method

Description

This method creates a new class in a category and returns it in the specified object.

Syntax

```
Set theClass = theCategory.AddClass (theName)
```

Element	Description
theClass As Class	Returns the newly created Class object
theCategory As Category	Category to which the new class is being added
theName As String	Name of the class to be created

Category.AddClassDiagram Method

Description

This method adds a new use case diagram or class diagram to a package (REI Category object). If the AddClassDiagram method is used with a package (Category) in the "Use Case View", the new diagram represents a use case diagram. If the AddClassDiagram method is used with a package (Category) in the "Logical View", the new diagram represents a class diagram.

Syntax

```
Set objClassDiagram = objPackage.AddClassDiagram
      (strDiagramName)
```

Element	Description
objClassDiagram As ClassDiagram	Returns the use case diagram or class diagram being added to the package (Category).
objPackage As Category	Package (Category) to which the diagram is being added
strDiagramName As String	Name to be given to the new diagram

Category.AddDataModelDiagram Method

Description

This method adds a new data model diagram to a package (REI Category object).

Syntax

```
Set objClassDiagram = objPackage.AddDataModelDiagram
      (strDiagramName)
```

Element	Description
objClassDiagram As ClassDiagram	Returns the data model diagram being added to the package (Category)
objPackage As Category	Package (Category) to which the data model diagram is being added
strDiagramName As String	Name to be given to the new data model diagram

Category.AddScenarioDiagram Method

Description

This method adds a scenario diagram to a category.

Syntax

`Set` *`theScenarioDiagram`* = *`theCategory`*.**`AddScenarioDiagram`** (*`theName, theType`*)

Element	Description
theScenarioDiagram As Scenario Diagram	Returns the scenario diagram being added to the category
theCategory As Category	Category to which the scenario diagram is being added
theName As String	Name of the scenario diagram being added
theType As Integer	Type of scenario diagram being added 1 = Sequence Diagram 2 = Collaboration Diagram

Category.AddUseCase Method

Description

This method creates a new use case in a category and returns it in the specified *object.*

Syntax

`Set` *`theUseCase`* = *`theCategory`*.**`AddUseCase`** (*`theName`*)

Element	Description
theUseCase As Use Case	Returns the newly created use case object
theName As String	Name of the use case being created, expressed as a string
theCategory As Category	Category to which the use case is being added

Category.DeleteCategory Method

Description

This method deletes a category from a category.

Syntax

```
Isdeleted = theCategory.DeleteCategory (theCategory)
```

Element	Description
Isdeleted As Boolean	Returns a value of True when the category is deleted
theCategory As Category	Instance of the category from which the category is being deleted
theCategory As Category	Instance of the category being deleted

Category.DeleteCategoryDependency Method

Description

This method deletes a category dependency from a category.

Syntax

```
IsDeleted = theCategory.DeleteCategoryDependency
        (theDependency)
```

Element	Description
IsDeleted As Boolean	Returns a value of True when the dependency is successfully deleted
theCategory As Category	Category from which to delete the dependency
theDependency As CategoryDependency	Category dependency to delete

Category.DeleteClass Method

Description

This method deletes a class from a category.

Syntax

Isdeleted = *theCategory*.**DeleteClass** (*theClass*)

Element	Description
Isdeleted As Boolean	Returns a value of True when the class is deleted
theCategory As Category	Category from which the class is being deleted
theClass As Class	Instance of the class being deleted

Category.DeleteClassDiagram Method

Description

This method deletes a class diagram from a category.

Syntax

Isdeleted = *theCategory*.**DeleteClassDiagram**
  (*theClassDiagram*)

Element	Description
Isdeleted As Boolean	Returns a value of True when the class diagram is deleted
theCategory As Category	Instance of the category from which the class diagram is being deleted
theClassDiagram As ClassDiagram	Instance of the class diagram being deleted

Category.DeleteScenarioDiagram Method

Description

This method deletes a scenario diagram from a category.

Syntax

```
Deleted = theCategory.DeleteScenarioDiagram
      (theScenarioDiagram)
```

Element	Description
deleted As Boolean	Returns a value of True when the scenario diagram is deleted
theCategory As Category	Instance of the category from which the scenario diagram is being deleted
theScenarioDiagram As Scenario Diagram	Instance of the scenario diagram being deleted

Category.DeleteUseCase Method

Description

This method deletes a use case from a category.

Syntax

```
Isdeleted = theCategory.DeleteUseCase (theUseCase)
```

Element	Description
Isdeleted As Boolean	Returns a value of True when the use case is deleted
theCategory As Category	Instance of the category from which the use case is being deleted
theUseCase As Use Case	Instance of the use case being deleted

Category.GetAllCategories Method

Description

This method returns all categories belonging to the category collection.

Syntax

Set *theCategories* = *theCategory*.**GetAllCategories** ()

Element	Description
theCategories As CategoryCollection	The collection of categories retrieved from the category
theCategory As Category	Category from which categories are being retrieved

Category.GetAllClasses Method

Description

This method returns all the classes belonging to the category and all of the classes belonging to all child categories of the specified category. If you only want to know the classes belonging to the specified category, use the Classes property.

Syntax

Set *theClasses* = *theCategory*.**GetAllClasses** ()

Element	Description
theClasses As ClassCollection	The collection of classes retrieved from the category and all of its child categories
theCategory As Category	Category from which classes and classes of child categories are being retrieved

Category.GetAllUseCases Method

Description

This method returns all use cases belonging to the category.

Syntax

`Set` *`theUseCases`* `=` *`theCategory`*`.`**`GetAllUseCases`** `()`

Element	**Description**
theUseCases As UseCaseCollection	The collection of use cases retrieved from the category
theCategory As Category	Category from which use cases are being retrieved

Category.GetAssignedSubsystem Method

Description

This method retrieves the assigned subsystem from a category.

Syntax

`Set` *`theSubsystem`* `=` *`theCategory`*`.`**`GetAssignedSubsystem`** `()`

Element	**Description**
theSubsystem As Subsystem	Returns the subsystem assigned to the category. If there is no assigned subsystem, returns a value of Nothing.
theCategory As Category	Instance of the category being checked for an assigned subsystem.

Category.GetCategoryDependencies Method

Description

This method returns the collection of category dependencies belonging to the category.

Syntax

```
Set theCategoryDependencies =
      theCategory.GetCategoryDependencies ()
```

Element	Description
theCategoryDependencies As CategoryDependencyCollection	Returns the collection category dependencies belonging to the category
theCategory As Category	Category whose category dependencies are being retrieved

Category.HasAssignedSubsystem Method

Description

This method determines whether a category has an assigned subsystem.

Syntax

```
isAssigned = theCategory.HasAssignedSubsystem ()
```

Element	Description
isAssigned As Boolean	Returns a value of True if the category has an assigned subsystem
theCategory As Category	Instance of the category being checked for an assigned subsystem

Category.RelocateAssociation Method

Description

This method relocates or moves an association from its current package to the specified package, usually the package in which the association's client and supplier classes are located. You will discover that this method needs to be used when you attempt to work with an association, for example, on a class diagram, but have no access to its specification or any information because the association is in an unloaded package. You can then use **RelocateAssociation** to relocate the association to a more appropriate package so that you can then work with it.

Syntax

theCategory.**RelocateAssociation** (*theAssociation*)

Element	Description
theCategory As Category	Package to which you want to move the association
theAssociation As Association	Association being relocated

Category.RelocateCategory Method

Description

This subroutine relocates a category in a model.

Syntax

theCategory.**RelocateCategory** *theCategory*

Element	Description
theCategory As Category	Model that contains the category being relocated
theCategory As Category	Category being relocated

Category.RelocateClass Method

Description

This subroutine relocates a class in a category.

Syntax

`theCategory``.`**`RelocateClass`** *`theClass`*

Element	Description
theCategory As Category	Category that contains the class being relocated
theClass As Class	Class being relocated

Category.RelocateClassDiagram Method

Description

This subroutine relocates a class diagram in a category.

Syntax

`theCategory``.`**`RelocateClassDiagram`** *`theClassDiagram`*

Element	Description
theCategory As Category	Category that contains the class diagram being relocated
theClassDiagram As ClassDiagram	Class diagram being relocated

Category.RelocateScenarioDiagram Method

Description

This subroutine relocates a scenario diagram in a category.

Syntax

theCategory.**RelocateScenarioDiagram** *theScenarioDiagram*

Element	Description
theCategory As Category	Instance of the category that contains the scenario diagram being relocated
theScenarioDiagram As ScenarioDiagram	ScenarioDiagram being relocated

Category.SetAssignedSubsystem Method

Description

This subroutine assigns a subsystem to a category.

Syntax

theCategory.**SetAssignedSubsystem** *theSubsystem*

Element	Description
theCategory As Category	Instance of the category to which the subsystem is being assigned
theSubsystem As Subsystem	Instance of the subsystem assigned to the category

Category.TopLevel Method

Description

This method determines whether the specified object is the root category.

Syntax

IsTopLevel = *theCategory*.**TopLevel** ()

Element	Description
IsTopLevel As Boolean	Returns a value of True if the specified object is the root category
theCategory As Category	Category object being tested as root category

CategoryDependency Class

The CategoryDependency class allows you to define and manipulate dependency relationships between categories.

CategoryDependency Class Properties

The following table summarizes the CategoryDependency Class properties.

Table 19 CategoryDependency Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Class Properties	Inherits all Relation class properties

CategoryDependency Class Methods

The following table summarizes the CategoryDependency Class methods.

Table 20   CategoryDependency Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
Relation Methods	Inherits all Relation class methods
GetContextCategory	Retrieves the context (client) category belonging to the category dependency
GetSupplierCategory	Retrieves the supplier category belonging to the category dependency

CategoryDependency.GetContextCategory Method

Description

This method retrieves the context (client) category belonging to the given category dependency.

Syntax

```
Set theCategory = theCategoryDependency.GetContextCategory
        ()
```

Element	**Description**
theCategory As Category	Returns the context (client) category belonging to the category dependency
theCategoryDependency As CategoryDependency	Category dependency whose context category is being retrieved

CategoryDependency.GetSupplierCategory Method

Description

This method retrieves the supplier category belonging to the given category dependency.

Syntax

```
Set theCategory = theCategoryDependency.GetSupplierCategory
        ()
```

Element	Description
theCategory As Category	Returns the supplier category belonging to the category dependency
theCategoryDependency As CategoryDependency	Category dependency whose supplier category is being retrieved

Class Class

The Class class allows you to get and set the characteristics and relationships of specific classes in a model.

Some of the questions answered by class properties are:

- Is this an abstract class?
- Is this class a fundamental type?
- Is this class persistent?
- Can this class be concurrent with any other classes?
- What set of attributes and operations belong to this class?
- What relationships are defined between this class and other objects in the model?

Class methods allow you to get and set this information for the classes in the model.

Class Class Properties

The following table summarizes the Class Class properties.

Table 21 Class Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Abstract	Indicates whether the class is abstract
Attributes	Collection that contains the attributes belonging to the class
Cardinality	Defines the cardinality of the class
ClassKind	Defines the kind of class this is
Concurrency	Controls concurrency of the class with other objects
ExportControl	Controls class visibility
FundamentalType	Defines the class as a fundamental type
Operations	Collection that contains the operations belonging to the class
Parameters	Collection that contains the parameters belonging to the class
ParentCategory	Category that contains the class
ParentClass	Specifies the parent class of this class
Persistence	Defines the class as persistent
Space	Defines the space algorithm to use for the class
StateMachine	State machine that belongs to the class

Class.Abstract Property

Description

Indicates whether the class is an abstract class.

Syntax

`Class``.`**`Abstract`**

Property Type

Boolean

Class.Attributes Property

Description

Causes the class to inherit all of the properties of a specified attribute collection.

Note: *This property is read-only.*

Syntax

`Class``.`**`Attributes`**

Property Type

AttributeCollection

Class.Cardinality Property

Description

Sets the cardinality of the class.

Syntax

`Class``.`**`Cardinality`**

Property Type

String

Class.ClassKind Property

Description

The ClassKind property is a rich data type. The following table describes the valid forms of expressing the ClassKind rich data type.

Table 22 Class.ClassKind Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection. Valid values are: ■ NormalClass ■ ParameterizedClass ■ InstantiatedClass ■ Utility ■ ParameterizedUtility ■ InstantiatedUtilitly ■ Meta ■ AllClasses ■ NotAClassKind

Note: *This property is read-only.*

Syntax

`Class``.`**`ClassKind`**

Property Type

ClassKind

Class.Concurrency Property

Description

The Class Concurrency property is a rich data type that controls class concurrency. The following table describes the valid forms of expressing the Class Concurrency rich data type for the Class class.

Table 23 Class.Concurrency Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Sequential ■ Guarded ■ Active ■ Synchronous

Note: *This property is read-only.*

Syntax

Class.**Concurrency**

Property Type

Concurrency

Class.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the class object. The following table describes the valid forms of expressing the ExportControl rich data type for the Class class.

Table 24 Class.ExportControl Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible string values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControl property is read-only. Its Value and Name, however, are read/write.*

Syntax

theClass.**ExportControl.Name** = **"PrivateAccess"**

theClass.**ExportControl.Value** = **2**

theNameStr = *theClass*.**ExportControl.Name**

theValue = *theClass*.**ExportControl.Value**

Property Type

RichType or ClassExportControl

Class.FundamentalType Property

Description

Defines this class as a fundamental type.

Syntax

Class.**FundamentalType**

Property Type

Boolean

Class.Operations Property

Description

Causes the class to inherit all of the operations of a specified operation collection.

Note: *This property is read-only.*

Syntax

Set *theOperationCollection* = *theClass*.**Operations**

Property Type

OperationCollection

Class.Parameters Property

Description

This property defines the collection of valid parameters for the class.

Note: *This property is read-only. To add parameters to a class, see AddParameter. To delete parameters from a class, see DeleteParameter.*

Syntax

theParms = *theClass*.**Parameters**

Property Type

ParameterCollection

Class.ParentCategory Property

Description

Indicates the category that contains the class.

Note: *This property is read-only.*

Syntax

Class.**ParentCategory**

Property Type

Category

Class.ParentClass Property

Description

Specifies the parent class of this class.

Note: *This property is read-only.*

Syntax

`Class``.`**`ParentClass`**

Property Type

Class

Class.Persistence Property

Description

Indicates whether the class is persistent.

Syntax

`Class``.`**`Persistence`**

Property Type

Boolean

Class.Space Property

Description

Defines the space algorithm to use for the class.

Syntax

`Class``.`**`Space`**

Property Type

String

Class.StateMachine Property

Description

Specifies the state machine that belongs to the class. A state machine defines all of the state information, including states, transitions, and state diagrams, defined for a given class.

A class can have zero or one state machine.

Note: *This property is read-only.*

Syntax

Class.**StateMachine**

Property Type

StateMachine

Class Class Methods

The following table summarizes the Class class methods.

Table 25 Class Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddAssignedModule	Assigns a module (component) to the class
AddAssociation	Adds an Association to a class
AddAttribute	Adds an attribute to a class
AddClassDependency	Adds a class dependency to the class
AddHas	Adds a HasRelationship to a class
AddInheritRel	Adds an InheritRelationship between classes in a class
AddInstantiateRel	Adds an instantiated relation between two classes

Method	Description
AddNestedClass	Adds a nested class to a class
AddOperation	Adds an operation to a class
AddParameter	Adds a parameter to a class
AddRealizeRel	Adds a realize relation to a class
CreateStateMachine	Creates a state machine for a class
DeleteAssociation	Deletes an Association from a class
DeleteAttribute	Deletes an attribute from a class
DeleteClassDependency	Deletes a class dependency from the class
DeleteHas	Deletes a HasRelationship from a class
DeleteInheritRel	Deletes an InheritRelationship between classes in a class
DeleteInstantiateRel	Deletes an instantiated relation from a class
DeleteNestedClass	Deletes a nested class from a class
DeleteOperation	Deletes an operation from a class
DeleteParameter	Deletes a parameter from a class.
DeleteRealizeRel	Deletes a realize relationship from a class
DeleteStateMachine	Deletes the class's state machine from the model
GetAllNestedClasses	Recursively retrieves all subclasses of a class
GetAssignedLanguage	Retrieves the name of the programming language assigned to the class
GetAssignedModules	Retrieves the modules assigned to the class
GetAssociateRoles	Retrieves the associate roles belonging to the class
GetAssociations	Retrieves the associations belonging to the class
GetClassDependencies	Retrieves the collection of class dependencies belonging to the class

Method	Description
GetClients	Retrieves the collection of classes that are clients of the specified class
GetHasRelations	Retrieves the Has Relationships belonging to the class
GetInheritRelations	Retrieves the Inherit Relations belonging to the class
GetInstantiateRelations	Retrieves the instantiated relations belonging to the class
GetLinkAssociation	Retrieves the association for a link class
GetNestedClasses	Retrieves the nested classes belonging to the class
GetRealizeRelations	Retrieves the collection of realize relations belonging to the class
GetRoles	Retrieves the roles belonging to the class
GetSubClasses	Retrieves the subclasses belonging to the class
GetSuperclasses	Retrieves the superclasses belonging to the class
IsALinkClass	Defines a class as a Link class
IsNestedClass	Determines whether the class is a nested class
RemoveAssignedModule	Removes an assigned module from the class

Class.AddAssignedModule Method

Description

This subroutine assigns a module to the class.

Syntax

theClass.**AddAssignedModule** *theModule*

Element	Description
theClass As Class	Class to which the module is being assigned
theModule As Module	Module being assigned to the class

Class.AddAssociation Method

Description

This method adds an association to a class and returns it in the specified object.

Syntax

Set *theAssociation* = *theClass*.**AddAssociation**
      (*theSupplierRoleName, theSupplierName*)

Element	Description
theAssociation As Association	Returns the association being added to the class
theClass As Class	Class to which the association is being added
theSupplierRoleName As String	Name of the supplier role in the association
theSupplierName As String	Name of the class, use case, or actor to which to attach the association ***Note:*** *If this name is not unique, you must use the qualified name (for example, Logical View::package_name::supplier_name)*

Class.AddAttribute Method

Description

This method creates a new attribute and adds it to a class.

Syntax

```
Set theAttribute = theClass.AddAttribute (AttrName,
      AttrType, InitValue)
```

Element	Description
theAttribute As Attribute	Returns the attribute being added to the class
theClass As Class	Class to which the attribute is being added
AttrName As String	Name of the attribute being added to the class
AttrType As String	Type of attribute being added to the class
InitValue As String	Initial value of the attribute

Class.AddClassDependency Method

Description

This method creates a new class dependency and adds it to a class.

Syntax

```
Set theDependency = theClass.AddClassDependency
      (theSupplierName, theSupplierType)
```

Element	Description
theClassDependency As ClassDependency	Returns the class dependency being added to the class
theClass As Class	Class to which the class dependency is being added
theSupplierName As String	Name of the supplier class of the class dependency
theSupplierType As String	Type of supplier of the class dependency

Class.AddHas Method

Description

This method creates a new HasRelationship and adds it to a class.

Syntax

```
Set theHasRel = theClass.AddHas
      (theSupplierRoleName,theSupplierRoleType)
```

Element	Description
theHasRel As HasRelation	Returns the Has relationship being added to the class
theClass As Class	Name of the class to which the relationship is being added
theSupplierRoleName As String	Name of the supplier role in the relationship
theSupplierRoleType As String	Type of the supplier role in the relationship

Class.AddInheritRel Method

Description

This method creates a new InheritRelation and adds it to a class.

Syntax

```
Set theInheritRel = theClass.AddInheritRel
      (theName,theParentClassName)
```

Element	Description
theInheritRel As InheritRelation	Returns the Inherit relationship being added to the class
theClass As Class	Class to which the relationship is being added
theName As String	Name of the relationship being added to the class
theParentClassName As String	Name of the parent class from which the Class inherits its properties and methods

Class.AddInstantiateRel Method

Description

This method creates a new instantiated relation and adds it to a class.

Syntax

```
Set theInstantiateRel = theClass.AddInstantiateRel
      (theSupplierName)
```

Element	Description
theInstantiateRel As InstantiateRelation	Returns the Instantiated relation being added to the class
theClass As Class	Class to which the instantiated relation is being added
theSupplierName As String	Name of the parameterized class who is supplying the actual values for the parameters of the instantiated relation

Class.AddNestedClass Method

Description

This method creates a new nested class and adds it to a class.

Syntax

```
Set theNestedClass = theClass.AddNestedClass (theName)
```

Element	Description
theNestedClass As Class	Returns the nested class being added to the class
theClass As Class	Class to which the nested class is being added
theName As String	Name of the class being added to the class

Class.AddOperation Method

Description

This method creates a new operation and adds it to a class.

Syntax

```
Set theOperation = theClass.AddOperation (OperationName,
      OperationType)
```

Element	Description
theOperation As Operation	Returns the operation being added to the class
theClass As Class	Class to which the operation is being added
OperationName As String	Name of the operation being added to the class
OperationType As String	Type of operation being added to the class

Class.AddParameter Method

Description

This method creates a new parameter and adds it to a parameterized class. In Rational Rose, this method corresponds to adding a line of information to the Formal Arguments section of the Detail tab of the Class Specification for a class whose Type field on the General tab is set to ParameterizedClass. Before calling this method, made sure that the class is a ParameterizedClass by checking the class' ClassKind property. If you attempt to add a parameter to a non-parameterized class, you will get a run-time error.

Syntax

```
Set theParm = theClass.AddParameter (theParmName,
      theParmType, theParmDefault, theParmPos)
```

Element	Description
theParm As Parameter	Returns the parameter being added to the class.
theClass As Class	Parameterized class to which the parameter is being added.
theParmName As String	Name of the parameter to be added to the class. Make sure that this name contains no spaces.
theParmType As String	Type of parameter to be added to the class (e.g., "Boolean", "Integer", "myUserDefinedType").
theParmDefault As String	Default value for the parameter (e.g., "True", "0", "Start").
theParmPos As Integer	Position of the parameter in the list. Position does matter. Position numbering starts at 0. Therefore, if you want to add the parameter to the top or first line in the list, set this argument to 0. To add the parameter to the second line from the top in the list, set this argument to 1. To add the parameter to the nth line in the list, set this argument to n–1. ***Note:*** *You cannot have "empty" lines in the list. Positioning the parameter at a particular line in the list only works if the list is filled up to and/or past that point.*

Parameter Position Examples

If you do not have any parameters in the list and add a parameter to position 2 (line 3), Rational Rose positions the parameter as first in the list (not third with 2 empty lines above it).

```
Set theParm = theClass.AddParameter ("Parameter3", "String",
       "", 2)
```

List before:	List after:
line 1:	line 1: Parameter3
line 2:	line 2:
line 3:	line 3:

If you have 1 parameter in the list and add a parameter to position 2 (line 3), Rational Rose does not leave line 2 blank and add the parameter to line 3. Instead, it adds the parameter to line 2.

```
Set theParm = theClass.AddParameter ("Parameter3", "String",
       "", 2)
```

List before:	List after:
line 1: Parameter1	line 1: Parameter1
line 2:	line 2: Parameter3
line 3:	line 3:

If, however, you have 2 or more parameters and add a parameter to position 2 (line 3), Rational Rose adds the parameter to line 3 in the parameter list.

```
Set theParm = theClass.AddParameter ("Parameter3", "String",
       "", 2)
```

List before:	List after:
line 1: Parameter1	line 1: Parameter1
line 2: Parameter2	line 2: Parameter2
line 3:	line 3: Parameter3

Class.AddRealizeRel Method

Description

This method creates a new realize relation and adds it to a class.

Syntax

```
Set theRealizeRelation = theClass.AddRealizeRel
      (theRelationName, theInterfaceName)
```

Element	Description
theRealizeRelation As RealizeRelation	Returns the realize relation being added to the class
theClass As Class	Class to which the realize relation is being added
theRelationName As String	Name of the relation being added
theInterfaceName As String	Name of the interface with which to create the realize relation

Class.CreateStateMachine Method

Description

This subroutine creates a state machine for a class.

Note: *A class can have zero or one state machine. Multiple state machines are not allowed.*

Syntax

```
theClass.CreateStateMachine
```

Element	Description
theClass As Class	Class to which you are adding the state machine

Class.DeleteAssociation Method

Description

This method deletes an association from a class.

Syntax

isDeleted = *theClass*.**DeleteAssociation** (*theAssociation*)

Element	Description
isDeleted As Boolean	Returns a value of True when the association is deleted
theClass As Class	Class from which the association is being deleted
theAssociation As Association	Name of the association being deleted (The association must belong to the specified class.)

Class.DeleteAttribute Method

Description

This method deletes an attribute from a class.

Syntax

isDeleted = *theClass*.**DeleteAttribute** (*theAttribute*)

Element	Description
isDeleted As Boolean	Returns a value of True when the attribute is deleted
theClass As Class	Class from which the attribute is being deleted
theAttribute As Attribute	Attribute being deleted from the class

Class.DeleteClassDependency Method

Description

This method deletes a class dependency from a class.

Syntax

`isDeleted = theClass.DeleteClassDependency (theDependency)`

Element	Description
isDeleted As Boolean	Returns a value of True when the class dependency is deleted
theClass As Class	Class from which the class dependency is being deleted
theDependency As ClassDependency	Class dependency being deleted

Class.DeleteHas Method

Description

This method deletes a HasRelationship from a class.

Syntax

`isDeleted = theClass.DeleteHas (theHasRel)`

Element	Description
isDeleted As Boolean	Returns a value of True when the Has relationship is deleted from the class
theClass As Class	Class from which the relationship is being deleted
theHasRel As HasRelationship	HasRelationship being deleted from the class

Class.DeleteInheritRel Method

Description

This method deletes an InheritRelation from a class.

Syntax

isDeleted = *theClass*.**DeleteInheritRel** (*theInheritRel*)

Element	Description
isDeleted As Boolean	Returns a value of True when the InheritRelation is deleted from the class
theClass As Class	Class from which the relationship is being deleted
theInheritRel As InheritRelation	InheritRelation being deleted from the class

Class.DeleteInstantiateRel Method

Description

This method deletes an instantiated relation from a class.

Syntax

```
Set isDeleted = theClass.DeleteInstantiateRel
      (theInstantiateRel)
```

Element	Description
isDeleted As Boolean	Returns a value of True when the instantiate relation is deleted from the class
theClass As Class	Class from which the instantiated relation is being deleted
theInstantiateRel As InstantiateRelation	Instantiated relation being deleted from the class

Class.DeleteNestedClass Method

Description

This method deletes an association from a class.

Syntax

`isDeleted` `=` *`theClass`*`.`**`DeleteNestedClass`** `(`*`theNestedClass`*`)`

Element	Description
isDeleted As Boolean	Returns a value of True when the nested class is deleted
theClass As Class	Class from which the nested class is being deleted
theNestedClass As Class	Nested class being deleted

Class.DeleteOperation Method

Description

This method deletes an operation from a class.

Syntax

`isDeleted` `=` *`theClass`*`.`**`DeleteOperation`** `(`*`theOperation`*`)`

Element	Description
isDeleted As Boolean	Returns a value of True when the operation is deleted from the class
theClass As Class	Class from which the operation is being deleted
theOperation As Operation	Operation being deleted from the class

Class.DeleteParameter Method

Description

This method deletes a parameter from a parameterized class. In Rational Rose, this method corresponds to deleting a line of information from the Formal Arguments section of the Detail tab of the Class Specification for a class whose Type field on the General tab is set to ParameterizedClass. Before calling this method, make sure that the class is a ParameterizedClass by checking the class' ClassKind property. If you attempt to delete a parameter from a non-parameterized class, you will get a run-time error.

Syntax

```
Set isDeleted = theClass.DeleteParameter (theParameter)
```

Element	**Description**
isDeleted As Boolean	Returns a value of True when the parameter is deleted from the class
theClass As Class	Parameterized class from which the parameter is being deleted
theParameter As Parameter	Parameter being deleted from the parameterized class

Class.DeleteRealizeRel Method

Description

This method deletes a realize relation from a class.

Syntax

isDeleted = *theClass*.**DeleteRealizeRel** (*theRealizeRel*)

Element	Description
isDeleted As Boolean	Returns a value of True when the relation is deleted from the class
theClass As Class	Class from which the realize relation is being deleted
theRealizeRel As RealizeRelation	Realize relation being deleted

Class.DeleteStateMachine Method

Description

This subroutine deletes a class's state machine from the model.

Syntax

theClass.**DeleteStateMachine**

Element	Description
theClass As Class	Class whose state machine is being deleted

Class.GetAllNestedClasses Method

Description

This method retrieves all classes nested within the specified class and all of its nested classes.

For example, if **Class A** has 2 nested classes, **NClass1** and **NClass2**, and **NClass1** has a nested class, **NestedCls**, applying the GetAllNestedClasses method to **Class A** returns all 3 nested classes, **NClass1**, **NClass2**, and **NestedCls**, not just the first-level nested classes.

To retrieve only the first-level nested classes for the specified class, use the Class.GetNestedClasses method.

Syntax

```
Set theClassCollection = myClass.GetAllNestedClasses ()
```

Element	Description
theClassCollection As ClassCollection	Collection of all classes that are nested within the specified class and all of its nested classes
myClass As Class	Class from which to retrieve all nested classes

Class.GetAssignedLanguage Method

Description

This method returns the name of the programming language assigned to the class.

Syntax

```
theLanguage = theClass.GetAssignedLanguage ()
```

Element	Description
theLanguage As String	Returns the name of the programming language assigned to the class
theClass As Class	Class whose assigned language is being returned

Class.GetAssignedModules Method

Description

This method retrieves the collection of modules assigned to a class.

Syntax

`Set` *`theModules`* `=` *`theClass`*`.`**`GetAssignedModules`**`()`

Element	Description
theModules As ModuleCollection	Returns the collection of modules assigned to the class. If no modules are assigned, returns a value of **Nothing**.
theClass As Class	Instance of the class from which the assigned modules are being retrieved

Class.GetAssociateRoles Method

Description

This method retrieves the roles of the classes associated with the specified class and returns them in the specified object.

To retrieve roles associated with the specified class itself, use GetRoles.

Syntax

`Set` *`theAssocRoles`* `=` *`theClass`*`.`**`GetAssociateRoles`** `()`

Element	Description
theAssocRoles As RoleCollection	Returns the role collection of classes associated with the specified class
theClass As Class	Class from which the collection is being retrieved

Comparison of GetAssociateRoles and GetRoles

The sample Rational RoseScript code below illustrates the differences between GetAssociateRoles and GetRoles.

```
Sub Main
   Dim theClasses As ClassCollection
   Dim classA As Class
   Dim ClassB As Class
   Dim theAssocRolesA As RoleCollection
   Dim theAssocRolesB As RoleCollection
   Dim theRolesA As RoleCollection
   Dim theRolesB As RoleCollection
   Dim theRole As Role

   viewport.open
   Set theClasses = RoseApp.CurrentModel.GetAllClasses ()
   Set classA = theClasses.GetFirst("A")
   Set classB = theClasses.GetFirst ("B")
   Set theAssocRolesA = classA.GetAssociateRoles ()
   Set theRolesA = classA.GetRoles ()
   Set theAssocRolesB = classB.GetAssociateRoles ()
   Set theRolesB = classB.GetRoles ()

   Print "GetAssociateRoles for A:"
   Tot = theAssocRolesA.Count
   For i = 1 To Tot
      Set theRole = theAssocRolesA.GetAt (i)
      Print theRole.Name
   Next i
   Print

   Print "GetRoles for A:"
   Tot = theRolesA.Count
   For i = 1 To Tot
      Set theRole = theRolesA.GetAt (i)
      Print theRole.Name
   Next i
   Print

   Print "GetAssociateRoles for B:"
   Tot = theAssocRolesB.Count
   For i = 1 To Tot
      Set theRole = theAssocRolesB.GetAt (i)
      Print theRole.Name
```

```
    Next i
    Print

    Print "GetRoles for B:"
    Tot = theRolesB.Count
    For i = 1 To Tot
       Set theRole = theRolesB.GetAt (i)
       Print theRole.Name
    Next i
    Print

End Sub
```

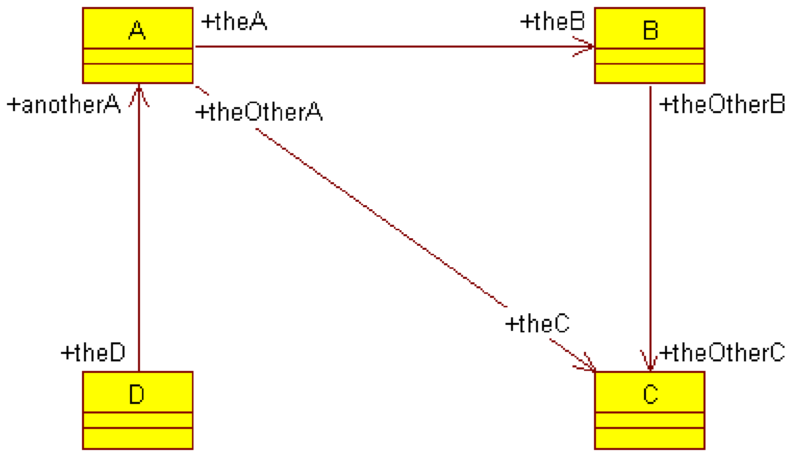

Figure 1   Example: Comparing GetAssociateRoles and GetRoles

Ouput

GetAssociateRoles for A:

theB

theD

theC

GetRoles for A:

theA

anotherA

theOtherA

GetAssociateRoles for B:

theA

theOtherC

GetRoles for B:

theB

theOtherB

Class.GetAssociations Method

Description

This method retrieves an association collection from a class and returns it in the specified object.

Syntax

Set *theAssociationCollection* = *theClass*.**GetAssociations** ()

Element	Description
theAssociationCollection As AssociationCollection	Returns the association collection from the class
theClass As Class	Class from which the collection is being retrieved

Class.GetClassDependencies Method

Description

This method retrieves the class dependencies belonging to the class.

Syntax

`Set` *`theClassDependencies`* `=` *`theClass`*`.`**`GetClassDependencies`** `()`

Element	**Description**
theClassDependencies As ClassDependencyCollection	Returns the class dependency collection belonging to the class
theClass As Class	Class from which the dependencies are being retrieved

Class.GetClients Method

Description

This method retrieves the collection of classes that are clients of the specified class. Prior to the exposure of this method, there was no way to traverse directly from a supplier class to the client class or classes. GetClients makes it easier for you to determine who the client classes are for a particular supplier class, which is very important in working with inheritance. This method is also useful when you want a supplier class to provide items to its client classes.

Syntax

```
Set theClassClients = theClass.GetClients (theRelationKind,
      theRelationType)
```

Element	Description
theClassClients As ClassCollection	Returns the collection of classes that are clients of the class
theClass As Class	Supplier class from which the client classes are being retrieved
theRelationKind As ClientRelKind Enum or Integer	Kind of relationship between the specified class and its clients (e.g., Friend)
theRelationType As ClientRelType Enum or Integer	Type of relationship between the specified class and its clients (e.g., Instantiation, Association, Dependency)

Class.GetHasRelations Method

Description

This method retrieves an HasRelationship collection from a class and returns it in the specified object.

Syntax

`Set` *`theHasRelations`* `=` *`theClass`*`.`**`GetHasRelations`** `()`

Element	**Description**
theHasRelations As HasRelationshipCollection	Returns theHasRelationship collection from the class
theClass As Class	Class from which the collection is being retrieved

Class.GetInheritRelations Method

Description

This method retrieves an inherit relation collection from a class and returns it in the specified object.

Syntax

`Set` *`theInheritRelations`* `=` *`theClass`*`.`**`GetInheritRelations`** `()`

Element	**Description**
theInheritRelations As InheritRelationshipCollection	Returns the InheritRelationship collection from the class
theClass As Class	Class from which the collection is being retrieved

Class.GetInstantiateRelations Method

Description

This method retrieves the collection of all instantiated relations from a class and returns it in the specified object. GetInstantiateRelations provides a quick way to get at all the instantiated relations belonging to a class.

Syntax

```
Set theInstantiateRelations =
      theClass.GetInstantiateRelations ()
```

Element	Description
theInstantiateRelations As InstantiateRelationCollection	Returns the Instantiated Relation collection from the class
theClass As Class	Class from which the instantiated relation collection is being retrieved

Class.GetLinkAssociation Method

Description

This method returns the corresponding association if the specified object is a link class.

Syntax

```
theAssociation = theClass.GetLinkAssociation ()
```

Element	Description
theAssociation As Association	Returns the link association for a link class
theClass As Class	Class whose link association is being retrieved

Class.GetNestedClasses Method

Description

This method retrieves the nested class collection from a class and returns it in the specified object.

Syntax

```
Set theNestedClasses = theClass.GetNestedClasses ()
```

Element	Description
theNestedClasses As ClassCollection	Returns the nested class collection from the class
theClass As Class	Class from which the collection is being retrieved

Class.GetRealizeRelations Method

Description

This method retrieves the collection of realize relations belonging to the class.

Syntax

```
Set theRealizeRelsColl = theClass.GetRealizeRelations ()
```

Element	Description
theRealizeRelsColl As RealizeRelationCollection	Returns the collection of realize relations belonging to the class
theClass As Class	Class from which the collection is being retrieved

Class.GetRoles Method

Description

This method retrieves the roles of the specified class and returns the role collection in the specified object.

To retrieve roles of the classes associated with the specified class, use GetAssociateRoles

Syntax

Set *theRoles* = *theClass*.**GetRoles** ()

Element	Description
theRoles As RoleCollection	Returns the collection of roles from the class
theClass As Class	Class from which the collection is being retrieved

Comparison of GetAssociateRoles and Get Roles

See discussion of same topic under the *Class.GetAssociateRoles Method* section in this reference guide.

Class.GetSubClasses Method

Description

This method retrieves the subclasses belonging to the class.

Syntax

Set *theSubclasses* = *theClass*.**GetSubclasses** ()

Element	Description
theSubclasses As ClassCollection	Returns the collection of classes belonging to the class
theClass As Class	Class from which the collection is being retrieved

Class.GetSuperclasses Method

Description

This method retrieves the parent superclasses of the class. Note that this method is not recursive. This method does not retrieve the grandparents, or any other ancestors, of the class. To retrieve all ancestors, write a procedure that calls this method for each parent class until all ancestors are retrieved. The REI does not provide a method to automatically retrieve all ancestor classes for a class.

Syntax

`Set` *`theParentClasses`* `=` *`theClass`*`.`**`GetSuperclasses`** `()`

Element	Description
theParentClasses As ClassCollection	Returns the collection of parent superclasses of the class
theClass As Class	Class from which the collection is being retrieved

Class.IsALinkClass Method

Description

This method determines whether a class is the link in a link attribute.

Syntax

`isLink` `=` *`theClass`*`.`**`isALinkClass`** `()`

Element	Description
isLink As Boolean	Returns a value of True if the specified class is the link in a link attribute
theClass As Class	The instance of the class being tested as a link class

Class.IsNestedClass Method

Description

This method that determines whether a class is nested.

Syntax

isNested = *theClass*.**IsNestedClass** ()

Element	Description
isNested As Boolean	Returns a value of True if the specified class is nested
theClass As Class	The instance of the class being checked for nesting

Class.RemoveAssignedModule Method

Description

This subroutine removes a module assignment from the class.

Syntax

theClass.**RemoveAssignedModule** *theModule*

Element	Description
theClass As Class	Class from which the module assignment is being removed
theModule As Module	Module whose class assignment is being removed

ClassDependency Class

The ClassDependency class exposes properties and methods that:

- Determine the characteristics of dependencies between classes
- Allow you to retrieve class dependencies

ClassDependency Class Properties

The following table summarizes the ClassDependency Class properties.

Table 26 ClassDependency Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Properties	Inherits all Relation class properties
ClassRelation Properties	Inherits all ClassRelation class properties
ClientCardinality	Client cardinality of the ClassDependency
ExportControl	Controls visibility of the ClassDependency
InvolvesFriendship	Indicates whether the ClassDependency involves friendship
SupplierCardinality	Supplier cardinality of the ClassDependency

ClassDependency.ClientCardinality Property

Description

Specifies the number of clients allowable for the ClassDependency.

Syntax

`ClassDependency``.`**`ClientCardinality`**

Property Type

String

ClassDependency.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the ClassDependency object. The following table describes the valid forms of expressing the ExportControl rich data type for the ClassDependency class.

Table 27 ClassDependency.Export Control Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible string values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControl property is read-only. Its Value and Name, however, are read/write.*

Syntax

theClassDependency.**ExportControl.Name** = **"PrivateAccess"**

theClassDependency.**ExportControl.Value** = **2**

theNameStr = *theClassDependency*.**ExportControl.Name**

theValue = *theClassDependency*.**ExportControl.Value**

Property Type

RichType or UsesRelationExportControl

ClassDependency.InvolvesFriendship Property

Description

Indicates whether the ClassDependency involves friendship.

Syntax

`ClassDependency``.`**`InvolvesFriendship`**

Property Type

Boolean

ClassDependency.SupplierCardinality Property

Description

Specifies the number of suppliers allowable for the ClassDependency.

Syntax

`ClassDependency``.`**`SupplierCardinality`**

Property Type

String

ClassDependency Class Methods

The following table summarizes the ClassDependency Class methods.

Table 28   ClassDependency Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
Relation Methods	Inherits all Relation methods
ClassRelation Methods	Inherits all ClassRelation methods

ClassDiagram Class

The class diagram class allows you to add, retrieve and delete classes and categories to and from a class diagram. The properties and methods of the ClassDiagram class apply specifically to class diagrams. In addition, this class inherits all diagram class properties and methods.

ClassDiagram Class Properties

The following table summarizes the ClassDiagram Class properties.

Table 29 ClassDiagram Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
Diagram Properties	Inherits all Diagram class properties
ParentCategory	Category that contains the class diagram

ClassDiagram.ParentCategory Property

Description

Specifies the category that contains the class diagram.

Note: *This property is read-only.*

Syntax

```
ClassDiagram.ParentCategory
```

Property Type

Category

ClassDiagram Class Methods

The following table summarizes the ClassDiagram Class methods.

Table 30 ClassDiagram Class Methods Summary

Method	**Description**
AddAssociation	Adds an association to the diagram
AddCategory	Adds a category to the diagram
AddClass	Adds a class icon to the diagram
AddUseCase	Adds a use case to the diagram
GetAssociations	Retrieves the collection that contains the associations belonging to the class diagram
GetCategories	Retrieves the collection that contains the categories belonging to the class diagram
GetClasses	Retrieves the collection that contains the classes belonging to the class diagram
GetClassView	Returns a class view from a class diagram
GetSelectedCategories	Retrieves the categories that are currently selected in the class diagram
GetSelectedClasses	Retrieves the classes that are currently selected in the class diagram
GetUseCases	Retrieves the use cases belonging to the diagram
IsDataModelingDiagram	Indicates whether the class diagram is a data model diagram
IsUseCaseDiagram	Determines whether the class diagram is a use case diagram
RemoveAssociation	Removes an association from a class diagram
RemoveCategory	Removes a category from a class diagram
RemoveClass	Removes a class from a class diagram
RemoveUseCase	Removes a use case from the diagram

ClassDiagram.AddAssociation Method

Description

This method adds an association icon to a class diagram.

Syntax

isAdded = *theObject*.**AddAssociation** (*theAssociation*)

Element	Description
isAdded As Boolean	Returns a value of True when the association icon is added to the diagram
theObject As ClassDiagram	Diagram to which the association icon is being added
theAssociation As Association	Association whose icon is being added to this class diagram

ClassDiagram.AddCategory Method

Description

This method adds a category icon to a class diagram.

Syntax

isAdded = *theObject*.**AddCategory** (*theCategory*)

Element	Description
isAdded As Boolean	Returns a value of True when the category icon is added to the diagram
theObject As ClassDiagram	Diagram to which the category icon is being added
theCategory As Category	Category whose icon is being added to the diagram

ClassDiagram.AddClass Method

Description

This method adds a class icon to a class diagram.

Syntax

isAdded = *theObject*.**AddClass** (*theClass*)

Element	Description
isAdded As Boolean	Returns a value of True when the class icon is added to the diagram
theObject As ClassDiagram	Diagram to which the class icon is being added
theClass As Class	Class whose icon is being added to this class diagram

ClassDiagram.AddUseCase Method

Description

This method adds a use case icon to a class diagram.

Syntax

isAdded = *theObject*.**AddUseCase** (*theUseCase*)

Element	Description
isAdded As Boolean	Returns a value of True when the use case icon is added to the diagram
theObject As ClassDiagram	Diagram to which the use case icon is being added
theUseCase As UseCase	Use case whose icon is being added to the diagram

ClassDiagram.GetAssociations Method

Description

This method retrieves a collection of associations from a class diagram.

Syntax

```
Set theAssociations = theObject.GetAssociations ()
```

Element	Description
theAssociations As AssociationCollection	Returns the collection of associations from the class diagram
theObject As ClassDiagram	Class diagram from which to retrieve the associations

ClassDiagram.GetCategories Method

Description

This method retrieves a collection of categories from a class diagram.

Syntax

```
Set theCategories = theObject.GetCategories ()
```

Element	Description
theCategories As CategoryCollection	Returns the collection of categories from the class diagram
theObject As ClassDiagram	Class diagram from which to retrieve the categories

ClassDiagram.GetClasses Method

Description

This method retrieves a collection of classes from a class diagram.

Syntax

Set *theClasses* = *theObject*.**GetClasses** ()

Element	Description
theClasses As ClassCollection	Returns the collection of classes from the class diagram
theObject As ClassDiagram	Class diagram from which to retrieve the classes

ClassDiagram.GetClassView Method

Description

This method retrieves a class view from a class diagram. If the view does not yet exist, the method creates the view.

Syntax

Set *theClassView* = *theObject*.**GetClassView** (*theClass*)

Element	Description
theClassView As ClassView	Returns a class view from a class diagram
theObject As ClassDiagram	Class diagram from which to retrieve the class view
theClass As Class	Class whose view is being retrieved

ClassDiagram.GetObject Method

Description

This method retrieves the OLE object associated with a specified class diagram object.

Note: *This method is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

Set *theOLEObject* = *theObject*.**GetObject** ()

Element	Description
theOLEObject As Object	Returns the OLE automation interface object associated with the specified object
theObject As ClassDiagram	Instance of the object whose interface object is being returned

ClassDiagram.GetSelectedCategories Method

Description

This method retrieves the collection of currently selected categories from a class diagram.

Syntax

Set *theCategories* = *theObject*.**GetSelectedCategories** ()

Element	Description
theCategories As CategoryCollection	Returns the collection of currently selected categories from the class diagram
theObject As ClassDiagram	Class diagram from which to retrieve the categories

ClassDiagram.GetSelectedClasses Method

Description

This method retrieves the collection of currently selected classes from a class diagram.

Syntax

Set *theClasses* = *theObject*.**GetSelectedClasses** ()

Element	Description
theClasses As ClassCollection	Returns the collection of currently selected classes from the classes diagram
theObject As ClassDiagram	Class diagram from which to retrieve the classes

ClassDiagram.GetUseCases Method

Description

This method retrieves a collection of use cases from a class diagram.

Syntax

Set *theUseCases* = *theObject*.**GetUseCases** ()

Element	Description
theUseCases As UseCaseCollection	Returns the collection of use cases from the class diagram
theObject As ClassDiagram	Class diagram from which to retrieve the use cases

ClassDiagram.IsDataModelingDiagram Method

Description

This method indicates whether a class diagram is a data model diagram.

Syntax

`*isDMDiagram* = *objClassDiagram*.**IsDataModelingDiagram** ( )`

Element	Description
isDMDiagram As Boolean	Returns a value of True if the specified class diagram is a data model diagram
objClassDiagram As ClassDiagram	Class diagram being tested as a data model diagram

ClassDiagram.IsUseCaseDiagram Method

Description

This method that determines whether a class diagram is a use case diagram.

Syntax

`*IsUseCase* = *theObject*.**IsUseCaseDiagram** ()`

Element	Description
IsUseCase As Boolean	Returns a value of True if the specified class diagram is a use case diagram
theObject As ClassDiagram	The instance of the class diagram being tested as a use case diagram

ClassDiagram.RemoveAssociation Method

Description

This method removes an association icon from a class diagram.

Syntax

Removed = **theObject**.*RemoveAssociation* (*theAssociation*)

Element	Description
Removed As Boolean	Returns a value of True when the association icon is removed from the diagram
theObject As ClassDiagram	Diagram from which the association icon is being removed
theAssociation As Association	Association whose icon is being removed from the diagram

ClassDiagram.RemoveCategory Method

Description

This method removes a category icon from a class diagram.

Syntax

Removed = *theObject*.**RemoveCategory** (*theCategory*)

Element	Description
Removed As Boolean	Returns a value of True when the category icon is removed from the diagram
theObject As ClassDiagram	Diagram from which the category icon is being removed
theCategory As Category	Category whose icon is being removed from the diagram

ClassDiagram.RemoveClass Method

Description

This method removes a class icon from a class diagram.

Syntax

Removed = *theObject*.**RemoveClass** (*theClass*)

Element	Description
Removed As Boolean	Returns a value of True when the class icon is removed from the diagram
theObject As ClassDiagram	Diagram from which the class icon is being removed
theClass As Class	Class whose icon is being removed from the diagram

ClassDiagram.RemoveUseCase Method

Description

This method removes a use case icon from a class diagram.

Syntax

Removed = *theObject*.**RemoveUseCase** (*theUseCase*)

Element	Description
Removed As Boolean	Returns a value of True when the use case icon is removed from the diagram
theObject As ClassDiagram	Diagram from which the use case icon is being removed
theUseCase As UseCase	Use case whose icon is being removed from the diagram

ClassRelation Class

The ClassRelation class inherits from the Relation class and is the parent class of the HasRelationship, ClassDependency, and InheritRelation classes.

ClassRelation Class Properties

The following table summarizes the ClassDiagram Class properties.

Table 31 ClassRelation Class Properties Summary

Property	Description
RoseItem properties	Inherits all RoseItem Properties
Element properties	Inherits all Element Properties
Relation properties	Inherits all Relation properties

ClassRelation Class Methods

The following table summarizes the ClassRelation Class methods.

Table 32 ClassRelation Class Methods Summary

Method	Description
RoseObject methods	Inherits all RoseObject methods
RoseItem methods	Inherits all RoseItem methods
Element methods	Inherits all Element methods
Relation methods	Inherits all Relation methods
GetContextClass	Retrieves the Class relation’s context (client) class
GetSupplierClass	Retrieves the Class relation’s supplier class

ClassRelation.GetContextClass Method

Description

This method retrieves the Class relation’s context (client) class.

Syntax

Set *theClass* = *theClassRelation*.**GetContextClass** ()

Element	**Description**
theClass As Class	Returns the realize relation’s context (client) class
theClassRelation As ClassRelation	ClassRelation whose context class is being retrieved

ClassRelation.GetSupplierClass Method

Description

This method retrieves the Class relation’s supplier class.

Syntax

Set *theClass* = *theClassRelation*.**GetSupplierClass** ()

Element	**Description**
theClass As Class	Returns the realize relation’s supplier class
theClassRelation As ClassRelation	ClassRelation whose supplier class is being retrieved

ClassView Class

A class is a set of objects that share a common structure and a common behavior. The class view is the visual representation of a class, and is what appears on a diagram in the model.

The class view class inherits the RoseItem properties and methods that determine the size and placement of the class view on a diagram.

Additional ClassView properties determine which information, such as operations, operation signature, and attributes are visible in the view.

ClassView Class Properties

The following table summarizes the ClassView Class properties.

Table 33 ClassView Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItemView Properties	Inherits all RoseItemView properties
AutomaticResize	Indicates whether the class view will be automatically resized when displayed in the view port
ShowOperationsSignature	Indicates whether the class's operations signature will be shown when the class view is displayed in the view port
ShowAllAttributes	Indicates whether the class's attributes will be visible when the class view is displayed in the view port
ShowAllOperations	Indicates whether the class's operations will be visible when the class view is displayed in the view port

Property	Description
SuppressAttributes	Indicates whether to suppress the class's attributes when the class view is displayed in the view port
SuppressOperations	Indicates whether to suppress the class's operations when the class view is displayed in the view port
Visibility	Sets or indicates whether access adornments appear

ClassView.AutomaticResize Property

Description

Indicates whether the class view will be automatically resized when displayed in the view port. Corresponds to the Automatic Resize option in the Rational Rose shortcut menu.

Syntax

ClassView.**AutomaticResize**

Property Type

Boolean

ClassView.ShowAllAttributes Property

Description

Indicates whether the class's attributes will be visible when the class view is displayed in the view port.

Syntax

ClassView.**ShowAllAttributes**

Property Type

Boolean

ClassView.ShowAllOperations Property

Description

Indicates whether the class's operations will be visible when the class view is displayed in the view port. Corresponds to the Show All Operations option in the Rational Rose shortcut menu.

Syntax

ClassView.**ShowAllOperations**

Property Type

Boolean

ClassView.ShowOperationsSignature Property

Description

Indicates whether the class's operations signature will be shown when the class view is displayed in the view port. Corresponds to the Show Operations Signature option in the Rational Rose shortcut menu.

Syntax

ClassView.**ShowOperationsSignature**

Property Type

Boolean

ClassView.SuppressAttributes Property

Description

Indicates whether to suppress the class's attributes when the class view is displayed in the view port. Corresponds to the Suppress Attributes option in the Rational Rose shortcut menu.

Syntax

`ClassView`.**`SuppressAttributes`**

Property Type

Boolean

ClassView.SuppressOperations Property

Description

Indicates whether to suppress the class's operations when the class view is displayed in the view port. Corresponds to the Suppress Operations option in the Rational Rose shortcut menu.

Syntax

`ClassView`.**`SuppressOperations`**

Property Type

Boolean

ClassView.Visibility Property

Description

This property sets or indicates whether the access adornments of the attributes and operations in a class view appear on a diagram.

Syntax

```
isVisible = objClassView.Visibility
objClassView.Visibility = TRUE
```

Property Type

Boolean

ClassView Class Methods

The following table summarizes the ClassView Class methods.

Table 34 ClassView Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItemView Methods	Inherits all RoseItemView methods
GetDisplayedAttributes	Returns the collection of attributes displayed on the class view
GetDisplayedOperations	Returns the collection of operations displayed on the class view

ClassView. GetDisplayedAttributes Method

Description

This method retrieves the collection of attributes displayed on the specified class view. To retrieve the collection of *all* attributes for the underlying class, use the Class.Attributes property.

Syntax

```
Set objAttributeCollection =
        objClassView.GetDisplayedAttributes ( )
```

Element	Description
objAttributeCollection As AttributeCollection	Returns the collection of attributes displayed on the class view
objClassView As ClassView	Class view from which the collection of displayed attributes is being retrieved

See also

ClassView.GetDisplayedOperations method

ClassView.GetDisplayedOperations Method

Description

This method retrieves the collection of operations displayed on the specified class view. To retrieve the collection of *all* operations for the underlying class, use the Class.Operations property.

Syntax

```
Set objOperationCollection =
     objClassView.GetDisplayedOperations ( )
```

Element	Description
objOperationCollection As OperationCollection	Returns the collection of operations displayed on the class view
objClassView As ClassView	Class view from which the collection of displayed operations is being retrieved

See also

ClassView.GetDisplayedAttributes method

ClientRelKind Enumeration

ClientRelKind is an enumeration that defines values corresponding to the kinds of relationships between Class objects. ClientRelKind can be used in the GetClients Class method.

The following table describes the valid values for the ClientRelKind enumeration.

Table 35 ClientRelKind Enumeration Valid Values

Value	**Integer Value**	**Description**
rsAnyKind	0	Use to evaluate every kind of relationship. For example, use **rsAnyKind** in the GetClients Class method to find client Class objects with any kind of relationship to a specified Class object.
rsFriend	1	Use to evaluate only friend relationships. For example, Use **rsFriend** in the GetClients Class method to only return client Class objects with a friend relationship to a specified Class object.

Note: *Rational Rose Automation users may use the value (e.g.,* ***rsAnyKind****) in their methods. Rational Rose Script users must use the integer value (e.g., 0) in their methods.*

ClientRelType Enumeration

ClientRelType is an enumeration that defines values corresponding to the types of relationships between Class objects. ClientRelType can be used in the GetClients Class method.

The following table describes the valid values for the ClientRelType enumeration.

Table 36 ClientRelType Enumeration Valid Values

Value	Integer Value	Description
rsTypeAny	0	Use to evaluate every type of relationship, including associations. For example, use **rsTypeAny** in the GetClients Class method to find all the client Class objects with any type of relationship (e.g., Has, Inherits) to the specified Class object.
rsTypeAssociation	4	Use to evaluate only Associations. For example, use **rsTypeAssociation** in the GetClients Class method to find all the client Class objects with an Association relationship to the specified Class object.
rsTypeDependency	5	Use to evaluate only Dependency relationships. For example, use **rsTypeDependency** in the GetClients Class method to find all the client Class objects with a Dependency relationship to the specified Class object.
rsTypeHas	1	Use to evaluate only Has relationships and associations. For example, use **rsTypeHas** in the GetClients Class method to find all the client Class objects with a Has relationship or Has association to the specified Class object.

Value	Integer Value	Description
rsTypeInherits	3	Use to evaluate only Inherits relationships. For example, use **rsTypeInherits** in the GetClients Class method to find all the client Class objects with an Inherits relationship to the specified Class object.
rsTypeInstantiation	2	Use to evaluate only Instantiation relationships. For example, use **rsTypeInstantiation** in the GetClients Class method to find all the client Class objects with an Instantiation relationship to the specified Class object.
rsTypeRealizes	6	Use to evaluate only Realizes relationships. For example, use **rsTypeRealizes** in the GetClients Class method to find all the client Class objects with a Realizes relationship to the specified Class object.

Note: *Rational Rose Automation users may use the value (e.g., **rsTypeAny**) in their methods. Rational Rose Script users must use the integer value (e.g., 0) in their methods.*

Collection Classes

For most elements of a Rational Rose model there is a corresponding collection. So, for example, for every class there is a class collection; for every category there is a category collection; for every property, there is a property collection, and so on.

Rational Rose extensibility provides properties and methods that allow you to access a particular element in any given collection.

Collection Class Properties

Count is the only property that applies to collections. It allows you to increment a counter from one element to the next as you iterate through the collection.

Table 37 Collection Class Properties Summary

Property	Description
Count	Position of an object within a collection

Collection.Count Property

Description

Often used when iterating through a collection, this property indicates the position of an object within a collection.

Note: *This property is read-only.*

Syntax

`theCollection`.**`Count`**

Property Type

Integer

Examples

```
theClassCollection.Count
theCategoryCollection.Count
theDeviceCollection.Count
```

Methods for All Collections

The following table summarizes the collection methods that allow you to locate and retrieve the elements in any collection. While all of these properties and methods are the same, they act upon different types of objects. For example, the ClassCollection.GetAt method retrieves a class object, the CategoryCollection.GetAt method retrieves a category object, and so on.

The following table summarizes the Collection Class methods that are applicable to all collections.

Table 38 Collection Class Methods Summary (for all collections)

Method	Description
Exists	Indicates whether an object exists in a given collection
FindFirst	Retrieves the index (position) of the first instance of an object in a given collection
FindNext	Retrieves the index (position) of the next instance of an object in a given collection
GetWithUniqueID	Retrieves the instance of an object in a given collection, given the object's unique ID ***Note:*** *Objects that do not have a uniqueID (for example, ExternalDocument and Property objects) cannot be retrieved using this method.*
GetAt	Retrieves a specified instance of an object in a given collection
GetFirst	Retrieves the first instance of an object from a given collection
GetObject	Returns the OLE interface object associated with the given collection
IndexOf	Finds the index (position) of an object in a given collection

Collection.Exists Method

Description

This method checks for the existence of an object in a collection.

Syntax

`Exists` `=` *`theCollection`*`.`**`Exists`** `(`*`theObject`*`)`

Element	Description
Exists As Boolean	Returns a value of True if the object exists in the collection
theCollection As Collection	The collection being checked
theObject As Object	Instance of the object whose existence is being checked

Collection.FindFirst Method

Description

This method returns the index (position) of the first instance of the named object from a collection.

Note: *To retrieve the object itself, use the GetAt method and specify the index returned by this method.*

Syntax

`Set` *`theIndex`* `=` *`theCollection`*`.`**`FindFirst`** `(`*`theName`*`)`

Element	Description
theIndex As Integer	Returns the index of the first instance of the named object in the collection Returns a value of 0 if the named object is not found
theCollection As Collection	Collection from which the index is being retrieved
theName As String	Name of the object whose index is being retrieved

Collection.FindNext Method

Description

When iterating through a collection, this method retrieves the index (position) of the next instance of the named object, given the index of the current instance.

Note: *To retrieve the object itself, use the GetAt method and specify the index returned by this method.*

Syntax

`NextIndex` = *`theCollection`*.**`FindNext`** (*`CurrentIndex, theName`*)

Element	**Description**
NextIndex As Integer	Returns the index of the next instance of an object from the collection Returns a value of 0 if the named object is not found
theCollection As Collection	Collection from which the next index is being retrieved
CurrentIndex As Integer	Index of the current object instance in the collection
theName As String	Name of the object whose index is being retrieved

Collection.GetAt Method

Description

This method retrieves a particular object from a collection, given the object's position in the collection.

Syntax

`Set` *theObject* = *theCollection*.**GetAt** (*theIndex*)

Note: *To get the index of the object, use the IndexOf, FindFirst or FindNext method.*

Element	Description
theObject As Object	Returns an object from the collection
theCollection As Collection	Collection from which to retrieve the object
theIndex As Integer	Index (position) of the object in the collection

Collection.GetFirst Method

Description

This method retrieves the first instance of the named object from a collection.

Syntax

`Set` *theObject* = *theCollection*.**GetFirst** (*theName*)

Element	Description
theObject As Object	Returns the first instance of the named object from the collection
theCollection As Collection	Collection from which to retrieve the object
theName As String	Name of the object to retrieve

Collection.GetObject Method

Description

This method retrieves the OLE object associated with a specified collection.

Note: *This method is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

```
Set theOLEObject = theCollection.GetObject ()
```

Element	Description
theOLEObject As Object	Returns the OLE automation interface object associated with the specified object
theObject As Collection	Instance of the object whose interface object is being returned

Collection.GetWithUniqueID Method

Description

This method retrieves an object from a collection, given the object's unique ID. This is simpler than iterating through the collection to find a named or indexed object.

Every element in a model has a unique ID. You cannot set this ID, but you can retrieve it.

Syntax

```
Set theObject = theCollection.GetWithUniqueID (theUniqueID)
```

Element	Description
theObject As Object	Returns the object whose unique ID you specify
theCollection As Collection	Collection from which to retrieve the object
theUniqueID As String	UniqueID of the object to retrieve

Collection.IndexOf Method

Description

This method retrieves the index (position) of an instance of an object in a collection.

Syntax

```
Set theIndex = theCollection.IndexOf (theObject)
```

Element	**Description**
theIndex As Integer	Returns the index (position) of the given object Returns a value of 0 if the class is not found
theCollection As Collection	Collection from which the index is being retrieved
theObject As Object	Instance of the object whose index is being retrieved

Methods for User-Defined Collections

The following table describes the four additional collection methods, which allow you to add and remove objects from a collection. However, these methods are only valid for user-defined collections and cannot be used with Rational Rose Model collections:

The following table summarizes the Collection Class methods that are applicable to user-defined collections only.

Table 39 User-Defined Collection Class Methods Summary

Method	**Description**
Add	Adds an object to the object collection
AddCollection	Adds a collection to an object collection
Remove	Removes a collection from an object collection
RemoveAll	Removes the entire contents of a collection

Collection.Add Method

Description

This subroutine adds an object to a collection.

Syntax

theCollection.**Add** *theObject*

Element	Description
theCollection As Collection	Collection to which the object is being added
theObject As Object	Object being added to the collection

Collection.AddCollection Method

Description

This subroutine adds a collection of objects to a collection.

Note: *The objects are added as individual objects, not as a collection. For this reason, should you need to remove one or more of these objects from the destination collection, you can simply use the Remove or RemoveAll method.*

Syntax

theCollection.**AddCollection** *theObjectCollection*

Element	Description
theCollection As Collection	Collection to which the collection of objects is being added
theObjectCollection As Collection	Collection whose objects are being added

Collection.Remove Method

Description

This subroutine removes an object from a collection.

Syntax

`theCollection`.**`Remove`** *`theObject`*

Element	**Description**
theCollection As Collection	Collection from which the class is being removed
theObject As Object	Object being removed from the collection

Collection.RemoveAll Method

Description

This subroutine removes all objects from a collection.

Syntax

`theCollection`.**`RemoveAll`**

Element	**Description**
theCollection As Collection	Collection from which all objects are being removed

ComponentView Class

You use components (also called modules) to capture a model's physical implementation information, such as program files and subsystems. The component view is the visual representation of a component and is what appears on a diagram in the model.

The component view class inherits the RoseItemView properties and methods that determine the size and placement of the component view. It also allows you to retrieve the component object itself from the component view.

ComponentView Class Properties

The following table summarizes the ComponentView Class properties.

Table 40 ComponentView Class Properties Summary

Property	**Description**
Element properties	Inherits all Element class properties
RoseItemView properties	Inherits all RoseItemView class properties

ComponentView Class Methods

The following table summarizes the ComponentView Class methods.

Table 41 ComponentView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element	Inherits all Element class methods
RoseItemView	Inherits all RoseItemView class methods
GetComponent	Retrieves the module object (component) represented by the component view

ComponentView.GetComponent Method

Description

This method retrieves the component (module) represented by the component view.

Syntax

Set *theComponent* = *theComponentView*.**GetComponent** ()

Element	**Description**
theComponent As Module	Returns the component represented by the component view. Note that the REI return class is currently called Module, not Component.
theComponentView As ComponentView	Instance of the component view whose corresponding component (module) is being retrieved.

ConnectionRelation Class Overview

The ConnectionRelation class is an abstract class that exposes Rose's connection functionality in the extensibility interface. With the properties and methods of the ConnectionRelation class, you can:

- Retrieve information about connections, such as name, application, model, documentation, stereotypes, external documents, and supplier name
- Create and retrieve tool and property settings for connections
- Retrieve the connection's unique internal Rose identification number and qualified name
- Add and delete external documents
- Open specification sheets for connections
- Retrieve and set the connection's characteristics
- Determine whether the connection has a supplier or client
- Determine if the connection's supplier is a Device or Processor
- Retrieve the connection's supplier or client

The ConnectionRelation class corresponds to connections in the Deployment Diagram of the Rose user interface.

ConnectionRelation Class Properties

The following table describes the ConnectionRelation Class properties.

Table 42 ConnectionRelation Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Properties	Inherits all Relation class properties
Characteristics	Text describing the physical characteristics of the connection
SupplierIsDevice	Indicates whether the supplier of the connection relation is a device

ConnectionRelation. Characteristics Property

Description

Use this property to specify text providing a physical description of the connection.

Examples:

RS232 cable

satellite-to-ground communication

Syntax

```
strCharacteristics = objConnectionRelation.Characteristics
objConnectionRelation.Characteristics = "RS232 cable"
```

Property Type

String

ConnectionRelation. SupplierIsDevice Property

Description

This property indicates whether the supplier of the connection is a device or process. If the supplier of the connection is a device, this property is True. If the supplier of the connection is a processor, this property is False.

Note: *This property is read-only.*

Syntax

```
blnSupplierIsDevice = objConnectionRelation.SupplierIsDevice
```

Property Type

Boolean

ConnectionRelation Class Methods

The following table describes the ConnectionRelation Class methods.

Table 43 ConnectionRelation Class Methods Summary

Method	**Description**
RoseObject methods	Inherits RoseObject methods
Element methods	Inherits Element methods
RoseItem methods	Inherits RoseItem methods
Relation methods	Inherits Relation methods

ContextMenuItem Class

The ContextMenuItem class allows you to define text and actions for the shortcut menus.

ContextMenuItem Class Properties

The following table summarizes the ContextMenuItem Class properties.

Table 44 ContextMenuItem Class Properties Summary

Property	**Description**
Caption	Specifies the text to be displayed to the user in the shortcut menu
InternalName	Specifies the internal name to be used by the add-in to identify the shortcut menu item
MenuID	Specifies the Rational Rose menu identification for the shortcut menu item
MenuState	Specifies the condition of the shortcut menu item (e.g., disabled, enabled, checked, unchecked)

ContextMenuItem.Caption Property

Description

Specifies the text to be displayed to the user in the shortcut menu. Type an ampersand (&) in front of the letter that you want to designate as the access key. This letter is underlined when Rational Rose displays the caption on the shortcut menu.

Syntax

```
theContextMenuItem.Caption = “My &Caption Name”

theCaption = theContextMenuItem.Caption
```

Property Type

String

ContextMenuItem.InternalName Property

Description

Specifies the internal name to be used by the Add-In to identify the shortcut menu item. This string is NOT localized. It is used to indicate which menu item is selected by the user. It is up to the Add-In to map this internal name to the requested action.

Syntax

theContextMenuItem.**InternalName** = **"internalName1"**

theName$ = *theContextMenuItem*.**InternalName**

Property Type

String

ContextMenuItem.MenuID Property

Description

Specifies the Rational Rose menu identification for the shortcut menu item. This property is useful for debugging purposes. It also provides another way to get at a particular shortcut menu option.

Note: *This is a read-only property. Rational Rose sets The MenuID when AddContextMenuItem is called.*

Syntax

theID = *theContextMenuItem*.**MenuID**

Property Type

Integer

ContextMenuItem.MenuState Property

Description

Specifies the condition of the shortcut menu item (e.g., disabled, enabled, checked, unchecked).

Syntax

theContextMenuItem.**MenuState** = **DISABLED**

theContextMenuItem.**MenuState** = **0**

theState = *theContextMenuItem*.**MenuState**

Property Type

MenuState Enum or Integer

ContextMenuItem Class Methods

The following table summarizes the ContextMenuItem Class methods.

Table 45 ContextMenuItem Class Methods Summary

Method	**Description**
RoseObject methods	Inherits all RoseObject class methods. Please note that the most useful of these methods for the ContextMenuItem Class is the IdentifyClass method.

ContextMenuItemType Enumeration

ContextMenuItemType is an enumeration that defines values corresponding to the types of items to which shortcut menu options can be applied. ContextMenuItemType can be used in the AddContextMenuItem and GetContextMenuItems AddIn methods.

The following table describes the valid values for the ContextMenuItemType enumeration.

Table 46 ContextMenuItemType Enumeration Valid Values

Value	**Integer Value**	**Description**
rsActivity	13	Use for shortcut menu options that only apply to activities
rsAttribute	5	Use for shortcut menu options that only apply to attributes
rsClass	4	Use for shortcut menu options that only apply to classes
rsComponent	7	Use for shortcut menu options that only apply to components
rsDecision	17	Use for shortcut menu options that only apply to decisions
rsDefault	0	Use for shortcut menu options that only apply to a multiple selection of different types of items
rsDeploymentUnit	11	Use for shortcut menu options that only apply to deployment units
rsDiagram	1	Use for shortcut menu options that only apply to diagrams
rsExternalDoc	12	Use for shortcut menu options that only apply to external documents
rsModel	10	Use for shortcut menu options that only apply to models
rsOperation	6	Use for shortcut menu options that only apply to operations

Value	Integer Value	Description
rsPackage	2	Use for shortcut menu options that only apply to packages
rsProperties	9	Use for shortcut menu options that only apply to model properties
rsRole	8	Use for shortcut menu options that only apply to roles
rsState	14	Use for shortcut menu options that only apply to states
rsSwimlane	18	Use for shortcut menu options that only apply to swimlanes
rsSynchronization	16	Use for shortcut menu options that only apply to synchronizations
rsTransition	15	Use for shortcut menu options that only apply to transitions
rsUseCase	3	Use for shortcut menu options that only apply to use cases

Note: *Rational Rose Automation users may use the value (e.g., **rsAttribute**) in their methods. Rational Rose Script users must use the integer value (e.g., 5) in their methods.*

ControllableUnit Class

The ControllableUnit class is a an abstract class that exposes Rose's controllable unit functionality in the extensibility interface.

For example, you can:

- Load and unload units
- Determine whether a unit is modifiable or has been modified
- Determine whether a unit is controlled
- Get the file name associated with a unit
- Save a unit to a file

ControllableUnit Class Properties

The following table summarizes the ControllableUnit Class properties.

Table 47 ControllableUnit Class Properties Summary

Property	**Description**
Element Class Properties	Inherits all Element class methods
RoseItem Class Properties	Inherits all RoseItem class properties

ControllableUnit Class Methods

The following table summarizes the ControllableUnit Class methods.

Table 48 ControllableUnit Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Class Methods	Inherits all Element class methods
RoseItem Class Methods	Inherits all RoseItem class methods
Control	Associates a controllable unit with a file that can be passed to a configuration management application
GetAllSubUnitItems	Recursively retrieves all subunits of the specified controllable unit
GetFileName	Retrieves the name of the file that contains the controllable unit
GetSubUnitItems	Retrieves the immediate subunits of the specified controllable unit
IsControlled	Indicates whether a controllable unit is controlled
IsLoaded	Indicates whether a controllable unit is loaded in the current model
IsLocked	Indicates whether a controllable unit is locked by another process
IsModifiable	Indicates whether a controllable unit can be modified

Method	Description
IsModified	Indicates whether a controllable unit has been modified
Load	Loads a controllable unit into the current model
Lock	Attempts to lock a controllable unit
Modifiable	Allows a controllable unit to be modified
NeedsRefreshing	Indicates whether a controllable unit needs to be updated
Refresh	Updates a loaded controllable unit
SaveAs	Saves a controllable unit under a new name
Save	Saves a controllable unit
Unload	Unloads a controllable unit from the current model
Unlock	Attempts to unlock a controllable unit
Uncontrol	Uncontrols a controllable unit

ControllableUnit.Control Method

Description

This method associates a controllable unit with a filename, so that it can be passed to a configuration management application.

Syntax

`IsControlled` = *`theControllableUnit`*.**`Control`** (*`thePath`*)

Element	Description
IsControlled As Boolean	Returns a value of True when the unit is successfully controlled
theControllableUnit As ControllableUnit	Name of the controllable unit to place under control
thePath As String	Fully qualified path and file name that contain the unit

ControllableUnit.GetAllSubUnitItems Method

Description

This method recursively retrieves all subunits of the specified controllable unit. It retrieves the immediate subunits of the specified controllable unit plus all subunits of the immediate subunits and so on. To retrieve only the immediate subunits of the specified controllable unit, use the GetSubUnitItems method.

Syntax

`Set` *`theSubUnits`* = *`theControllableUnit`*.**`GetAllSubUnitItems`** ()

Element	Description
theSubUnits As ControllableUnitCollection	Recursively returns the collection of controllable units that are subunits of the specified controllable unit
theControllableUnit As ControllableUnit	Controllable unit whose subunits are being retrieved

ControllableUnit.GetFileName Method

Description

This method retrieves the name of the file that contains the controllable unit.

Syntax

```
theFileName = theControllableUnit.GetFileName ()
```

Element	Description
theFileName As String	Returns the name of the file that contains the controllable unit
theControllableUnit As ControllableUnit	Controllable unit whose file name is being retrieved

ControllableUnit.GetSubUnitItems Method

Description

This method retrieves the immediate subunits of the specified controllable unit. This method is not recursive. If the subunits of the specified controllable unit have subunits, GetSubUnitItems does not include those subunits in the returned collection. To recursively retrieves all subunits of the specified controllable unit, use the GetAllSubUnitItems method.

Syntax

```
Set theSubUnits = theControllableUnit.GetSubUnitItems ()
```

Element	Description
theSubUnits As ControllableUnitCollection	Returns the collection of controllable units that are immediate subunits of the specified controllable unit
theControllableUnit As ControllableUnit	Controllable unit whose immediate subunits are being retrieved

ControllableUnit.IsControlled Method

Description

This method checks whether a given controllable unit has an associated file.

Syntax

`Controlled` = *`theControllableUnit`*.**`IsControlled`** ()

Element	Description
Controlled As Boolean	Returns a value of True if the given controllable unit has an associated file
theControllableUnit As ControllableUnit	Controllable unit being checked

ControllableUnit.IsLoaded Method

Description

This method checks whether a given controllable unit is loaded in the current model.

Syntax

`Loaded` = *`theControllableUnit`*.**`IsLoaded`** ()

Element	Description
Loaded As Boolean	Returns a value of True if the given controllable unit is loaded in the current model
theControllableUnit As ControllableUnit	Controllable unit being checked

ControllableUnit.IsLocked Method

Description

This method checks whether a given controllable unit is locked by another process. Use the Lock method to lock a controllable unit. Use the Unlock method to unlock a controllable unit.

Syntax

isCntUnitLocked = *theControllableUnit*.**IsLocked** ()

Element	Description
isCntUnitLocked As Boolean	Returns a value of True if the controllable unit is locked and is therefore in use by another process and cannot be modified Returns a value of False if the controllable unit is locked by the calling process which can, therefore, modify the controllable unit
theControllableUnit As ControllableUnit	Controllable unit being checked

ControllableUnit.IsModifiable Method

Description

This method checks whether a given controllable unit is flagged as modifiable.

Syntax

Modifiable = *theControllableUnit*.**IsModifiable** ()

Element	Description
Modifiable As Boolean	Returns a value of True if the unit is flagged as modifiable
theControllableUnit As ControllableUnit	Controllable unit being checked

ControllableUnit.IsModified Method

Description

This method checks whether a given controllable unit has been modified since the last time it was checked out of source control.

Syntax

`Modified` `=` *`theControllableUnit`*`.`**`IsModified`** `()`

Element	**Description**
Modified As Boolean	Returns a value of True if the unit has been modified since the last time it was checked out of source control
theControllableUnit As ControllableUnit	Controllable unit being checked

ControllableUnit.Load Method

Description

This method loads a controllable unit in the current model.

Syntax

`IsLoaded` `=` *`theControllableUnit`*`.`**`Load`** `()`

Element	**Description**
IsLoaded As Boolean	Returns a value of True when the controllable unit is loaded in the current model
theControllableUnit As ControllableUnit	Controllable unit being loaded

ControllableUnit.Lock Method

Description

This method attempts to lock a controllable unit. For the lock to be successful, the controllable unit must not be locked by another process. To determine whether the controllable unit is locked by another process, use IsLocked. Use the Unlock method to unlock a controllable unit.

Syntax

theControllableUnit.**Lock** ()

Element	Description
theControllableUnit As ControllableUnit	Controllable unit being locked

ControllableUnit.Modifiable Method

Description

This method sets a controllable unit as modifiable or not modifiable:

- If you pass a parameter of True on this method, the controllable unit will be modifiable.
- If you pass a parameter of False on this method, the controllable unit will not be modifiable.

Syntax

UnitSet = *theControllableUnit*.**Modifiable** (*Modifiable*)

Element	Description
UnitSet As Boolean	Returns a value of True when the controllable unit's modifiable status has been successfully set
theControllableUnit As ControllableUnit	Controllable unit being set to modifiable or not modifiable
Modifiable As Boolean	If True, controllable unit is modifiable; if False, controllable unit is not modifiable

ControllableUnit.NeedsRefreshing Method

Description

This method checks whether the specified controllable unit needs to be updated. That is, this method checks whether the controllable unit has changed since it was loaded. After determining that the controllable unit needs to be updated, use the Refresh method to update the controllable unit.

Syntax

`isDirty` `=` *`theControllableUnit`*`.`**`NeedsRefreshing`** `()`

Element	Description
isDirty As Boolean	Returns a value of True when the controllable unit has changed and needs to be updated
theControllableUnit As ControllableUnit	Controllable unit being checked

ControllableUnit.Refresh Method

Description

This method updates a loaded controllable unit. Use Refresh to update a controllable unit that has changed since it was loaded. To determine whether a loaded controllable unit needs updating, use the NeedsRefreshing method.

Syntax

`theControllableUnit``.`**`Refresh`** `()`

Element	Description
theControllableUnit As ControllableUnit	Loaded Controllable unit to be updated

ControllableUnit.Save Method

Description

This method saves a controllable unit.

Syntax

IsSaved = *theControllableUnit*.**Save** ()

Element	**Description**
IsSaved As Boolean	Returns a value of True when the controllable unit is successfully saved
theControllableUnit As ControllableUnit	Controllable unit being saved

ControllableUnit.SaveAs Method

Description

This method saves a controllable unit to a different file.

Syntax

IsSaved = *theControllableUnit*.**SaveAs** (*Path*)

Element	**Description**
IsSaved As Boolean	Returns a value of True when the controllable unit is successfully saved
theControllableUnit As ControllableUnit	Controllable unit being saved
Path As String	Fully qualified path and file name in which to save the controllable unit

ControllableUnit.Uncontrol Method

Description

This method uncontrols a controllable unit.

Syntax

`IsUncontrolled` `=` *`theControllableUnit`*`.`**`Uncontrol`** `()`

Element	**Description**
IsUncontrolled As Boolean	Returns a value of True when the controllable unit is uncontrolled
theControllableUnit As ControllableUnit	Controllable unit being uncontrolled

ControllableUnit.Unload Method

Description

This method unloads a controllable unit from the current model.

Syntax

`IsUnloaded` `=` *`theControllableUnit`*`.`**`Unload`** `()`

Element	**Description**
IsUnloaded As Boolean	Returns a value of True when the controllable unit is unloaded from the current model
theControllableUnit As ControllableUnit	Controllable unit being unloaded

ControllableUnit.Unlock Method

Description

This method attempts to unlock a controllable unit. For the unlock to be successful, the calling process must own the lock on the controllable unit. To determine whether the controllable unit is locked by another process, use IsLocked. Use the Lock method to lock a controllable unit.

Syntax

theControllableUnit.**Unlock** ()

Element	Description
theControllableUnit As ControllableUnit	Controllable unit being unlocked

Decision Class

The Decision class is an abstract class that exposes Rose's decision functionality in the extensibility interface. With the properties and methods of the Decision class, you can:

- Retrieve information about the decision, such as name, application, model, documentation, stereotypes, and external documents
- Retrieve objects associated with decisions such as parent activities, parent states, parent state machines, and swimlanes
- Create and retrieve tool and property settings for decisions
- Add and delete external documents
- Open specification sheets for decisions
- Add and delete transitions

The Decision class corresponds to decisions in the Rose user interface.

Decision Class Properties

The following table describes the Decision Class properties.

Table 49 Decision Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
StateVertex Properties	Inherits all StateVertex class properties

Decision Class Methods

The following table describes the Decision Class methods.

Table 50 Decision Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
StateVertex Methods	Inherits all StateVertex class methods

DecisionView Class

The DecisionView class is an abstract class that exposes Rose's decision view functionality in the extensibility interface. With the properties and methods of the DecisionView class, you can:

- Retrieve information about the decision represented by the decision view, including the decision object
- Retrieve objects associated with the decision view such as the diagram it is on, any parent or child views, and line vertices
- Retrieve physical information about the decision view such as position, height, width, fill color, line color, font
- Create and retrieve tool and property settings for decision views

The DecisionView class corresponds to decisions on diagrams in the Rose user interface.

DecisionView Class Properties

The following table describes the DecisionView Class properties.

Table 51 DecisionView Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItemView Properties	Inherits all RoseItemView class properties

DecisionView Class Methods

The following table describes the DecisionView Class methods.

Table 52 DecisionView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItemView Methods	Inherits all RoseItemView class methods
GetDecision	Returns the decision object represented by the decision view object

DecisionView.GetDecision Method

Description

This method retrieves the decision represented by the decision view.

Syntax

```
Set theDecision = myDecisionView.GetDecision ()
```

Element	**Description**
theDecision As Decision	Returns the decision object represented by the decision view object
myDecisionView As DecisionView	Decision view object from which to retrieve the decision object

DefaultModelProperties Class

The DefaultModelProperties Class is a container for the default model properties that belong to a model. There is one and only one DefaultModelProperties object per model.

Note: *If you use PropertyCollection methods to retrieve model properties, the collection can include both default and non-default model properties.*

DefaultModelProperties Class Properties

The following table summarizes the DefaultModelProperties Class properties.

Table 53 DefaultModelProperties Class Properties Summary

Property	**Description**
Element Class Properties	Inherits all Element class properties
RoseItem Class Properties	Inherits all RoseItem class properties

DefaultModelProperties Class Methods

The following table summarizes the DefaultModelProperties Class methods.

Table 54 DefaultModelProperties Class Methods Summary

Method	Description
AddDefaultProperty	Adds a default property to a property set
CloneDefaultPropertySet	Clones a default property set to use as a base for creating a new property set
CreateDefaultPropertySet	Creates a new default property set from scratch
DeleteDefaultProperty	Deletes a default property from a default property set
DeleteDefaultPropertySet	Deletes a default property from a model
FindDefaultProperty	Finds a specified default property given a model, class, and tool name
GetDefaultPropertySet	Retrieves the default property set for a given tool and class
GetDefaultSetNames	Retrieves the names of the default property sets for a given class and tool name
GetToolNamesForClass	Retrieves the tool names associated with a given class
IsToolVisible	Determines whether the tab with a given tool's default model properties is visible in specifications
SetToolVisibility	Sets the tool's visibility; that is, whether the property tab for the given tool will appear in specifications

DefaultModelProperties.AddDefaultProperty Method

Description

This method adds a default property to a model:

- The class name, tool name and set name determine where the property is added.
- The property name, property type, and property value define the property itself.

Syntax

```
IsAdded = theProperties.AddDefaultProperty (theClassName,
      theToolName, theSetName, thePropName, thePropType,
      theValue)
```

Element	Description
IsAdded As Boolean	Returns a value of True when the default property is successfully added
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the class to which the default property applies; corresponds to the Type field in the property specification editor of the Rational Rose user interface Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element.
theToolName As String	Name of the tool to which the default property applies; If the tool does not exist, it will be created
theSetName As String	Name of the property set to which the default property applies
thePropName As String	Name of the default property
thePropType As String	PropertyType of the default property
theValue As String	Value of the default property

DefaultModelProperties.CloneDefaultPropertySet Method

Description

This method creates a new default property set by cloning an existing property set.

Syntax

```
IsCloned = theProperties.CloneDefaultPropertySet
      (theClassName, theToolName, theExistingSetName,
      theNewSetName)
```

Element	Description
IsCloned As Boolean	Returns a value of True when the default property set is successfully cloned
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the extensibility class to which the new default property set applies Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element
theToolName As String	Name of the tool to which the new default property set applies
theExistingSetName As String	Name of the existing default property set being cloned
theNewSetName As String	Name of the new default property set created from the clone

DefaultModelProperties.CreateDefaultPropertySet Method

Description

This method creates a new default property set without using an existing property set as a base.

Syntax

IsCreated = *theProperties*.**CreateDefaultPropertySet** (*theClassName, theToolName, theNewSetName*)

Element	Description
IsCreated As Boolean	Returns a value of True when the default property set is successfully created
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the extensibility class to which the new default property set applies Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element
theToolName As String	Name of the tool to which the new default property set applies
theNewSetName As String	Name of the newly created default property set

DefaultModelProperties.DeleteDefaultProperty Method

Description

This method deletes a default property from a model. This method only deletes the property that belongs to the given class, tool, and set. If a different combination of class, tool, and set contains a default property with the same property name, that default property will remain intact and will not be deleted.

Syntax

```
IsDeleted = theProperties.DeleteDefaultProperty
      (theClassName, theToolName, theSetName,
      thePropName)
```

Element	**Description**
IsDeleted As Boolean	Returns a value of True when the default property is successfully deleted
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the extensibility class to which the default property applies Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element
theToolName As String	Name of the tool to which the default property applies
theSetName As String	Name of the property set to which the default property applies
thePropName As String	Name of the default property to delete

DefaultModelProperties.DeleteDefaultPropertySet Method

Description

This method deletes a default property set from a model.

Syntax

```
IsDeleted = theProperties.DeleteDefaultPropertySet
      (theClassName, theToolName, theSetName)
```

Element	Description
IsDeleted As Boolean	Returns a value of True when the default property set is successfully deleted
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the extensibility class to which the deleted default property set applies Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element
theToolName As String	Name of the tool to which the deleted default property set applies
theSetName As String	Name of the default property set to delete

DefaultModelProperties.FindDefaultProperty Method

Description

This method finds a specific default model property, given the name of the class, tool, and property set that contain it.

Syntax

```
thePropertey = theProperties.FindDefaultProperty
      (theClassName, theToolName, theSetName,
      thePropName)
```

Element	Description
theProperty As Property	Returns the default model property, if found Returns an empty value if the property does not exist
theProperties As DefaultModelProperties	Contains the properties belonging to the model
theClassName As String	Name of the extensibility class to search Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element
theToolName As String	Name of the tool to search
theSetName As String	Name of the default property set to search
thePropName As String	Name of the default property to find

DefaultModelProperties.GetDefaultPropertySet Method

Note: *This method replaces the Element class method,* ***GetPropertiesFromSet*** *used in Rational Rose version 4.0*

Description

This method retrieves the set of default model properties that belongs to a given extensibility class, tool, and set.

Syntax

```
Set thePropColl = theProperties.GetDefaultPropertySet
      (theClassName, theToolName, theSetName)
```

Element	Description
thePropColl As PropertyCollection	Returns the set of default model properties that belongs to the specified extensibility class, tool, and set
theProperties As DefaultModelProperties	Contains the properties belonging to the model
theClassName As String	Name of the extensibility class to which the retrieved default property set belongs Use the Element.GetPropertyClassName method to retrieve the valid string to pass as *theClassName* for a model element
theToolName As String	Name of the tool to which the retrieved default property set belongs
theSetName As String	Name of the set from which the property collection is being retrieved

DefaultModelProperties.GetDefaultSetNames Method

Description

This method retrieves the names of the default property sets that contain the model's default properties.

Syntax

```
theSetNames = theProperties.GetDefaultSetNames
      (theClassName, theToolName)
```

Element	Description
theSetNames As StringCollection	Returns a StringCollection containing the valid default property set names for the given extensibility class and tool
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the extensibility class for which you are retrieving valid default property set names Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element
theToolName As String	Name of the tool for which you are retrieving valid default property set names

DefaultModelProperties.GetToolNamesForClass Method

Description

This method retrieves the names of the tools associated with the given properties and class name.

Syntax

```
Set theToolNames = theProperties.GetToolNamesForClass
         (theClassName)
```

Element	Description
theToolNames As StringCollection	Returns a StringCollection containing the valid tool names for the given extensibility class
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theClassName As String	Name of the extensibility class for which you are retrieving valid tool names Use the Element.GetPropertyClassName method to retrieve the valid string to pass as theClassName for a model element

DefaultModelProperties.IsToolVisible Method

Description

This method determines whether the property tab for the given tool will appear in the property specification.

Syntax

`IsVisible` = *`theProperties`*.**`IsToolVisible`** (*`theToolName`*)

Element	Description
IsVisible As Boolean	Returns a value of True if the default model properties' tool is visible
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theToolName As String	Name of the tool to which the default properties belong

DefaultModelProperties.SetToolVisibility Method

Description

This subroutine sets the tool's visibility; that is, whether the property tab for the given tool will appear in the property specification.

Syntax

`theProperties`.**`SetToolVisibility`** *`theToolName`*, *`Visibility`*

Element	Description
theProperties As DefaultModelProperties	Contains the default properties belonging to the model
theToolName As String	Name of the tool whose visibility is being set
Visibility As Boolean	Set to True to make the tool visible; set to False to make the tool invisible

DependencyRelation Class

The DependencyRelation class is an abstract class that exposes Rose's dependency relationship functionality in the extensibility interface. With the properties and methods of the DependencyRelation class, you can:

- Retrieve information about dependency relationships such as name, documentation, and stereotype
- Retrieve information about objects associated with dependency relationships such as supplier name
- Retrieve objects associated with dependency relationships such as external documents, state machine owner, Rose application, model, supplier, and client
- Determine if the dependency relationship has a client and supplier
- Open specification sheets for dependency relationships
- Add and delete external documents
- Create and retrieve tool and property settings for dependency relationships

The DependencyRelation class corresponds to dependency relationships in the Rose user interface.

DependencyRelation Class Properties

The following table summarizes the DependencyRelation class properties.

Table 55 DependencyRelation Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Properties	Inherits all Relation class properties

DependencyRelation Class Methods

The following table summarizes the DependencyRelation class methods.

Table 56 DependencyRelation Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
Relation Methods	Inherits all Relation class methods

DeploymentDiagram Class

A deployment diagram is a visual representation of devices and processors. The deployment diagram class exposes properties and methods that allow you to add, retrieve and delete devices and processors in a deployment diagram.

DeploymentDiagram Class Properties

The following table summarizes the DeploymentDiagram Class properties.

Table 57 DeploymentDiagram Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
Diagram Properties	Inherits all Diagram Class properties

DeploymentDiagram Class Methods

The following table summarizes the DeploymentDiagram Class methods.

Table 58 DeploymentDiagram Class Methods Summary

Property	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
Diagram Methods	Inherits all Diagram Class methods
AddDevice	Adds a device to a deployment diagram
AddProcessor	Adds a processor to a deployment diagram
GetDevices	Retrieves the devices belonging to a deployment diagram
GetProcessors	Retrieves the processor belonging to a deployment diagram
RemoveDevice	Removes a device from a deployment diagram
RemoveProcessor	Removes a processor from a deployment diagram

DeploymentDiagram.AddDevice Method

Description

This method adds a device icon to a deployment diagram.

Syntax

Set *theView* = *theObject*.**AddDevice** (*theDevice, XPosition, YPosition*)

Element	Description
theView As RoseItemView	Returns the device icon being added to the diagram
theObject As DeploymentDiagram	Diagram to which the icon is being added
theDevice As Device	Device whose icon is being added to the diagram
Xposition As Integer	X axis coordinate of the icon in the diagram
YPosition As Integer	Y axis coordinate of the icon in the diagram

DeploymentDiagram.AddProcessor Method

Description

This method adds a processor icon to a deployment diagram.

Syntax

```
Set theView = theObject.AddProcessor (theProcessor,
      XPosition, YPosition)
```

Element	Description
theView As RoseItemView	Returns the processor icon being added to the diagram
theObject As DeploymentDiagram	Diagram to which the icon is being added
theProcessor As Processor	Processor whose icon is being added to the diagram
XPosition As Integer	X axis coordinate of the icon in the diagram
YPosition As Integer	Y axis coordinate of the icon in the diagram

DeploymentDiagram.GetDevices Method

Description

This method retrieves the collection of devices belonging to the deployment diagram.

Syntax

```
Set theDevices = theObject.GetDevices ()
```

Element	Description
theDevices As DeviceCollection	Returns the collection of devices belonging to the deployment diagram
theObject As DeploymentDiagram	Deployment diagram from which to retrieve the devices

DeploymentDiagram.GetProcessors Method

Description

This method retrieves the collection of processors belonging to the deployment diagram.

Syntax

Set *theProcessors* = *theObject*.**GetProcessors** ()

Element	Description
theProcessors As ProcessorCollection	Returns the collection of processors belonging to the deployment diagram
theObject As DeploymentDiagram	Deployment diagram from which to retrieve the processors

DeploymentDiagram.RemoveDevice Method

Description

This method removes a device icon from a deployment diagram.

Syntax

Removed = *theObject*.**RemoveDevice** (*theDevice*)

Element	Description
Removed As Boolean	Returns a value of True when the device icon is removed
theObject As DeploymentDiagram	Diagram from which the icon is being removed
theDevice As Device	Device whose icon is being removed from the diagram

DeploymentDiagram.RemoveProcessor Method

Description

This method removes a processor icon from a deployment diagram.

Syntax

`Removed = theObject.RemoveProcessor (theProcessor)`

Element	Description
Removed As Boolean	Returns a value of True when the processor icon is removed
theObject As DeploymentDiagram	Diagram from which the icon is being removed
theProcessor As Processor	Processor whose icon is being removed from the diagram

DeploymentUnit Class

The DeploymentUnit class exposes Rational Rose's deployment diagram functionality in the extensibility interface. DeploymentUnit allows you to use the same methods and properties as ControllableUnit. Use the DeploymentUnit Class to work with the Deployment Diagram.

With the DeploymentUnit class, you can:

- Load and unload the deployment diagram
- Determine whether the deployment diagram is modifiable or has been modified
- Determine whether the deployment diagram is controlled
- Get the file name associated with the deployment diagram
- Save the deployment diagram to a file

Check the lists of properties and methods for complete information.

DeploymentUnit Class Properties

The following table summarizes the DeploymentUnit Class properties.

Table 59 DeploymentUnit Class Properties Summary

Property	**Description**
Element Properties	Inherits Element properties
RoseItem Properties	Inherits RoseItem properties

DeploymentUnit Class Methods

The following table summarizes the DeploymentUnit Class methods.

Table 60 DeploymentUnit Class Methods Summary

Method	**Description**
Element Methods	Inherits Element methods
RoseItem Methods	Inherits RoseItem methods
RoseObject Methods	Inherits RoseObject methods
ControllableUnit Methods	Inherits ControllableUnit methods

Device Class

A device is hardware that is not capable of executing a program (a printer, for example). The device class exposes properties and methods that allow you to define and manipulate the characteristics of devices.

Device Class Properties

The following table summarizes the Device Class properties.

Table 61 Device Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Characteristics	Collection that contains the characteristics belonging to the device
Connections	Returns the collection of connections for the device

Device.Characteristics Property

Description

Specifies the characteristics of the device.

Syntax

Device.**Characteristics**

Property Type

String

Device. Connections Property

Description

This property returns the collection of connections for the specified device.

Note: *This property is read-only.*

Syntax

```
Set objConnectionRelationCollection = objDevice.Connections
```

Property Type

ConnectionRelationCollection

Device Class Methods

The following table summarizes the Device Class methods.

Table 62 Device Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddDeviceConnection	Adds a device connection to the device
AddProcessorConnection	Adds a processor connection to the device
GetConnectedDevices	Retrieves the collection of devices that are connected to the device
GetConnectedProcessors	Retrieves the collection of processors that are connected to the device
RemoveDeviceConnection	Removes a device connection from the device
RemoveProcessorConnection	Removes a processor connection from the device

Device.AddDeviceConnection Method

Description

Creates a new device connection and adds it to the device.

Syntax

`Connected` `=` *`theDevice`*`.`**`AddDeviceConnection`** `(`*`theDevice`*`)`

Element	**Description**
Connected As Boolean	Returns a value of True when the device is connected
theDevice As Device	Device to which the connection is being added
theDevice As Device	Device connection being added

Device.AddProcessorConnection Method

Description

Creates a new processor connection and adds it to the device.

Syntax

`Connected` `=` *`theDevice`*`.`**`AddProcessorConnection`** `(`*`theProcessor`*`)`

Element	**Description**
Connected As Boolean	Returns a value of True when the processor is connected
theDevice As Device	Device to which the connection is being added
theProcessor As Processor	Processor connection being added

Device.GetConnectedDevices Method

Description

Retrieves the collection of devices that are connected to the device.

Syntax

Set *theDevices* = *theDevice*.**GetConnectedDevices** ()

Element	Description
theDevices As DeviceCollection	Returns the collection of devices belonging to the device
theDevice As Device	Device whose connected devices are being retrieved

Device.GetConnectedProcessors Method

Description

This method retrieves the collection of processors that are connected to this device.

Syntax

Set *theProcessors* = *theDevice*.**GetConnectedProcessors** ()

Element	Description
theProcessors As ProcessorCollection	Returns the collection of processors that are connected to the specified processor
theDevice As Device	Device whose connected processors are being retrieved

Device.RemoveDeviceConnection Method

Description

Removes a device connection from the device.

Syntax

```
Removed = theDevice.RemoveDeviceConnection (theDevice)
```

Element	Description
Removed As Boolean	Returns a value of True when the device connection is removed
theDevice As Device	Device from which the connection is being removed
theDevice As Device	Device connection being removed

Device.RemoveProcessorConnection Method

Description

Removes a processor connection from the device.

Syntax

```
Removed = theDevice.RemoveProcessorConnection
       (theProcessor)
```

Element	Description
Removed As Boolean	Returns a value of True when the processor connection is removed
theDevice As Device	Device from which the connection is being removed
theProcessor As Processor	Processor connection being removed

Diagram Class

The Diagram class exposes a set of properties and methods, which all other diagram classes (for example, class diagrams, scenario diagrams, etc.) inherit. These properties and methods determine the size and placement of a diagram on the Rational Rose user's computer screen.

Diagram Class Properties

The following table summarizes the Diagram Class properties.

Table 63 Diagram Class Properties Summary

Property	**Description**
Element	Inherits all Element class properties
Documentation	Specifies the documentation belonging to the diagram
ExternalDocuments	Specifies the collection of external documents belonging to the diagram
Items	Collection of items belonging to the diagram
ItemViews	Collection of item views belonging to the diagram
Visible	Determines diagram visibility
ZoomFactor	Specifies the zoom factor of the diagram

Diagram.Documentation Property

Description

Specifies the documentation belonging to the Diagram.

Syntax

```
Diagram.Documentation
```

Property Type

String

Diagram.ExternalDocuments Property

Description

This property specifies the collection of external documents belonging to a particular diagram. Rational Rose lists ExternalDocuments for a diagram in the browser. Since external documents are controllable, you can use this property, for example, to get all the external documents belonging to a diagram and send them to a configuration management system.

Note: *This property is read-only.*

Syntax

myDiagramExtDocs = *theDiagram*.**ExternalDocuments**

Property Type

ExternalDocumentCollection

Diagram.Items Property

Description

Specifies the collection of items belonging to the diagram.

Note: *This property is read-only.*

Syntax

Diagram.**Items**

Property Type

ItemCollection

Diagram.ItemViews Property

Description

Specifies the collection of item views belonging to the diagram.

Note: *This property is read-only.*

Syntax

Diagram.**ItemViews**

Property Type

ItemViewCollection

Diagram.Visible Property

Description

Indicates whether the diagram is visible on the computer screen.

Syntax

Diagram.**Visible**

Property Type

Boolean

Diagram.ZoomFactor Property

Description

This property specifies the percentage zoom factor at which to display the diagram. Set this to a number between 20 and 100, inclusive, where 20 corresponds to 20% of actual size and 100 corresponds to 100% of actual size (or actual size).

Syntax

x = *theDiagram*.**ZoomFactor**

theDiagram.**ZoomFactor** = **100**

Property Type

Integer

Diagram Class Methods

The following table summarizes the Diagram Class methods.

Table 64 Diagram Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
Activate	Makes a diagram the currently active diagram in the application
AddExternalDocument	Adds an external document object to a diagram
AddNoteView	Adds a note view to a diagram
AddRelationView	Adds a relation view object to a diagram
DeleteExternalDocument	Deletes an external document object from a diagram
Exists	Determines whether a specified diagram object exists

Method	Description
GetNoteViews	Retrieves the note views belonging to a diagram
GetParentContext	Returns the model element that contains the diagram
GetSelectedItems	Retrieves the currently selected items from a diagram
GetViewFrom	Retrieves a view from a diagram
Invalidate	Redraws a diagram
IsActive	Indicates whether the diagram is the currently active diagram in the application
Layout	Draws a diagram
RemoveItemView	Removes a Rose item view
RemoveNoteView	Removes a note view from a diagram
Render	Creates a Windows metafile and renders the diagram to the file
RenderEnhanced	Creates an enhanced Windows metafile and renders the diagram to the file
RenderEnhancedToClipBoard	Renders the diagram in enhanced metafile format and stores it in the Clipboard
RenderToClipboard	Renders the diagram in metafile format and stores it in the Clipboard
Update	Updates a diagram

Diagram.Activate Method

Description

This subroutine makes the specified diagram the active diagram in Rational Rose. The active diagram is the window in Rational Rose which currently has the focus.

Syntax

theDiagram.**Activate**

Element	Description
theDiagram As Diagram	Diagram to activate

Diagram.AddExternalDocument Method

Description

This method adds an external document object to a diagram. In Rational Rose, this method corresponds to right-clicking on a diagram in the browser and selecting **New** from the shortcut menu.

Note: *This method does not check whether the added file or URL is valid. You must check this yourself.*

Syntax

```
Set theExternalDoc = theObject.AddExternalDocument
      (theExternalDocName, theExternalDocType)
```

Element	Description
theExternalDoc as ExternalDocument	Returns the external document object added to the specified diagram
theObject As Diagram	Diagram to which the external document object is being added
theExternalDocName As String	Contains the name of the external document object. For a file, this must include the entire path name (e.g., “C:\My Documents\My File Name”). For a URL, this must include the entire address (e.g., “http://myCompany.com/myHomePage/mySubPage/”)
theExternalDocType As Integer	Indicates the external document type: 1 = File 2 = URL

Diagram.AddNoteView Method

Description

This method adds a note view object to a diagram.

Syntax

```
Set theNoteView = theDiagram.AddNoteView (theNoteText,
      theNoteViewType)
```

Element	Description
theNoteView as NoteView	Returns the note view object added to the diagram
theDiagram As Diagram	Diagram to which the note view object is being added
theNoteText As String	Contains the text of the note view object
theNoteViewType As Integer	Indicates whether the note is free floating or enclosed in a box: 1 = Free floating text label 2 = Note with box

Diagram.AddRelationView Method

Description

This method adds a relation view object to a diagram.

Syntax

```
isAdded = theDiagram.AddRelationView (theRelation)
```

Element	Description
isAdded As Boolean	Returns a value of True if a view of the relation is successfully added to the diagram
theDiagram As Diagram	Diagram to which to add the view of the relation
theRelation As Relation	Relation to add to the diagram

Diagram.DeleteExternalDocument Method

Description

This method deletes an external document object from a diagram. In Rational Rose, this method corresponds to right-clicking on the external document in the browser and selecting **Delete** from the shortcut menu.

Syntax

```
Set isDeleted = theObject.DeleteExternalDocument
       (theExternalDocument)
```

Element	Description
isDeleted As Boolean	Returns the value of True if the external document object is successfully deleted
theObject As Diagram	Diagram from which the external document object is being deleted
theExternalDoc as ExternalDocument	External document object to be deleted from the diagram

Diagram.Exists Method

Description

This method determines whether a specified diagram object exists.

Syntax

```
Exists = theDiagram.Exists (theRoseItem)
```

Element	Description
Exists As Boolean	Returns the value of True if the diagram object exists
theDiagram As Diagram	Instance of the diagram whose existence is being checked
theRoseItem As RoseItem	Instance of the Rational Rose item that corresponds to the diagram object

Diagram.GetNoteViews Method

Description

This method returns the collection of note views belonging to a diagram.

Syntax

Set *theNoteViews* = *theDiagram*.**GetNoteViews** ()

Element	Description
theNoteViews As NoteViewCollection	Returns the collection of note views belonging to the diagram
theDiagram As Diagram	Instance of the diagram whose note view objects are being retrieved

Diagram. GetParentContext Method

Description

This method returns the model element that contains (provides the context for) the specified diagram. For example, for an overview diagram in the Logical View Package of the model, GetParentContext returns the category object named “Logical View”.

Syntax

Set *objRoseItem* **=** *objDiagram*.**GetParentContext ( )**

Element	Description
objRoseItem As RoseItem	Model element that contains the specified diagram
objDiagram As Diagram	Diagram whose containing model element is being retrieved

Diagram.GetSelectedItems Method

Description

This method returns all currently selected items in a diagram.

Syntax

`Set` *`theItemCollection`* = *`theDiagram`*.**`GetSelectedItems`** `()`

Element	Description
theItemCollection As ItemCollection	Returns the Rational Rose item view (view object) that represents the specified Rational Rose item
theDiagram As Diagram	Instance of the diagram whose selected items are being retrieved

Diagram.GetViewFrom Method

Description

This method retrieves the Rational Rose item view that represents the specified Rational Rose item.

Syntax

`Set` *`theView`* = *`theDiagram`*.**`GetViewFrom`** `(`*`theRoseItem`*`)`

Element	Description
theView As RoseItemView	Returns the Rational Rose item view (view object) that represents the specified Rational Rose item
theDiagram As Diagram	Instance of the diagram that contains the view object
theRoseItem As RoseItem	Instance of the Rational Rose item whose view item is being returned

Diagram.Invalidate Method

Description

This subroutine invalidates a Rational Rose diagram; that is, it causes the diagram to be redrawn.

Syntax

`theDiagram.Invalidate`

Element	Description
theDiagram As Diagram	Diagram being redrawn

Diagram.IsActive Method

Description

This method indicates whether the diagram is the currently active diagram in the application.

Syntax

`IsActive = theDiagram.IsActive ()`

Element	Description
IsActive As Boolean	Returns a value of True if the diagram is currently active in Rational Rose; otherwise it returns a value of False
theDiagram As Diagram	Diagram being checked as current diagram

Diagram.Layout Method

Description

This method arranges a Rose diagram according to the same layout algorithm used by the **Format > Layout Diagram** feature in the Rose user interface.

Syntax

```
objDiagram.Layout
```

Element	Description
objDiagram As Diagram	Diagram being arranged according to the Rose layout algorithm

Diagram.RemoveItemView Method

Description

This method removes an item view object from a diagram.

Syntax

```
IsRemoved = theDiagram.RemoveItemView (theRoseItemView)
```

Element	Description
IsRemoved As Boolean	Returns a value of True if the Rose item view object is successfully removed
theDiagram As Diagram	Diagram from which the Rose item view object is being removed
theRoseItemView as RoseItemView	Rose item view object to be removed from the diagram

Diagram.RemoveNoteView Method

Description

This method removes a note view object from a diagram.

Syntax

```
Set IsRemoved = theDiagram.RemoveNoteView (theNoteView)
```

Element	Description
IsRemoved As Boolean	Returns a value of True when the note view object is successfully removed
theDiagram As Diagram	Diagram from which the note view object is being removed
theNoteView as NoteView	Note view object to be removed from the diagram

Diagram.Render Method

Description

This subroutine renders a Rational Rose diagram to a Windows metafile, allowing the diagram to be opened and edited in any application that works with Windows metafiles.

Syntax

```
theDiagram.Render theFileName
```

Element	Description
theDiagram As Diagram	Diagram to render
theFileName As String	Name of the Windows metafile in which to save the diagram

Diagram.RenderEnhanced Method

Description

This subroutine renders a Rational Rose diagram to an enhanced Windows metafile, allowing the diagram to be opened and edited in any application that works with Windows metafiles.

Syntax

theDiagram.**RenderEnhanced** *theFileName*

Element	**Description**
theDiagram As Diagram	Diagram to render
theFileName As String	Name of the enhanced Windows metafile in which to save the diagram

Diagram.RenderEnhancedToClipBoard Method

Description

This subroutine renders a Rational Rose diagram to the Clipboard, preserving its Enhanced metafile formatting information. As with any Clipboard object, it can then be pasted into other windows or compatible applications.

Syntax

theDiagram.**RenderEnhancedToClipboard**

Element	**Description**
theDiagram As Diagram	Diagram to render

Diagram.RenderToClipboard Method

Description

This subroutine renders a Rational Rose diagram to the Clipboard in Windows metafile format. As with any Clipboard object, it can then be pasted into other windows or compatible applications.

Syntax

theDiagram.**RenderToClipboard**

Element	Description
theDiagram As Diagram	Diagram to render

Diagram.Update Method

Description

This subroutine updates a Rational Rose diagram.

Syntax

theDiagram.**Update**

Element	Description
theDiagram As Diagram	Diagram being updated

Element Class

The Element class provides the interface to model properties.

Every object in a Rational Rose model (including the model itself) is an element. And every element in a Rational Rose model has a name and a unique ID. Following this logic, you can use Element Class methods to obtain the ID for any item in the current model, and from there get or set its properties and property sets.

The unique element ID also provides the most direct means of accessing an item from a collection. While you can still use GetFirst and GetNext methods to iterate through a collection, you can also use the GetwithUniqueID method to obtain the item right away, without searching through the collection.

Element Class Properties

The following table summarizes the Element Class properties.

Table 65 Element Class Properties Summary

Property	**Description**
Application	Application to which the element belongs
Model	Model to which the element belongs
Name	Name of the model element

Element.Application Property

Description

Application to which the element belongs.

Note: *This property is read-only.*

Syntax

Element.**Application**

Property Type

Application

Element.Model Property

Description

Model to which the element belongs.

Note: *This property is read-only.*

Syntax

`*Element*.**Model**`

Property Type

Model

Element.Name Property

Description

Name of the model element.

Syntax

`*Element*.**Name**`

Property Type

String

Element Class Methods

The following table summarizes the Element Class methods.

Table 66 Element Class Methods Summary

Method	**Description**
RoseObject	Inherits all RoseObject methods
CreateProperty	Creates a new property for a given model element
FindDefaultProperty	Retrieves the default property for a given element property
FindProperty	Retrieves a property of the given element

Method	Description
GetAllProperties	Retrieves the collection of properties for the element
GetCurrentPropertySetName	Retrieves the current property set for the given model element
GetDefaultPropertyValue	Retrieves the default value for a property of the given model element
GetDefaultSetNames	Retrieves the names of the default property sets belonging to the element
GetIconIndex	Retrieves the index of the specified element's browser icon
GetPropertyClassName	Retrieves a string value (that is, a class name) required by the model object when working with default property sets
GetPropertyValue	Retrieves the current value for a property of the given model element
GetQualifiedName	Retrieves the qualified name of a model element
GetToolNames	Retrieves the names of the tools associated with the element
GetToolProperties	Retrieves the collection of properties for the given element and tool
GetUniqueId	Retrieves the unique ID associated with the given model element
InheritProperty	Removes the overridden value from an element's property so that the default value is used
IsDefaultProperty	Indicates whether the current value of a property is set to its default value
IsOverriddenProperty	Indicates whether the current value of a property is an overridden value

Method	**Description**
OverrideProperty	Overrides the default value of a property
RenderIconToClipboard	Places the element's browser icon on the Clipboard
SetCurrentPropertySetName	Specifies a given property set as the current property set for the element

Element.CreateProperty Method

Description

This method creates a new property for a given model element and tool.

Syntax

```
IsCreated = theElement.CreateProperty (theToolName,
      thePropName, theValue, theType)
```

Element	**Description**
IsCreated As Boolean	Returns a value of True when the property is created for the element
theElement As Element	Element for which the property is being created
theToolName As String	Name of the tool to which the property applies
thePropName As String	Name of the property being created
theValue As String	Default value of the new property
theType As String	Property type of the property Valid values are: ■ String ■ Integer ■ Float ■ Char ■ Boolean ■ Enumeration ***Note:*** *Other values may be valid if user-defined enumerated types exist.*

Element.FindDefaultProperty Method

Description

This method returns the default property, regardless of whether it has been overridden. To retrieve the overridden property, use FindProperty.

Syntax

```
Set theProperty = theElement.FindDefaultProperty
      (theToolName, thePropName)
```

Element	**Description**
theProperty As Property	Returns the default property given its name and associated tool name
theElement As Element	Model element whose default property is being returned
theToolName As String	Name of the tool to which the default property applies
thePropName As String	Name of the default property being retrieved

Element.FindProperty Method

Description

This method returns the overridden property, if one exists. Otherwise, it returns the default property. To retrieve the default property, even if overridden, use FindDefaultProperty.

Syntax

```
Set theProperty = theElement.FindProperty (theToolName,
        thePropName)
```

Element	Description
theProperty As Property	Returns the overridden or default property given its name and its associated tool name
theElement As Element	Model element whose overridden or default property is being returned
theToolName As String	Name of the tool to which the overridden or default property applies
thePropName As String	Name of the overridden or default property being retrieved

Element.GetAllProperties Method

Description

This method returns the collection of properties belonging to the specified element.

Syntax

```
Set theProperties = theElement.GetAllProperties ()
```

Element	Description
theProperties As PropertyCollection	Returns the collection of properties belonging to the specified element
theElement As Element	Model element whose properties are being returned

Element.GetCurrentPropertySetName Method

Description

This method returns the name of the currently active property set given the element and a tool name.

Syntax

```
theName = theElement.GetCurrentPropertySetName
      (theToolName)
```

Element	Description
theName As String	Returns the name of the currently active property set
theElement As Element	Element to which the property set belongs
theToolName As String	Name of the tool to which the property set belongs

Element.GetDefaultPropertyValue Method

Description

This method retrieves the default property value given a tool name and property name.

Syntax

```
theValue = theElement.FindDefaultProperty (theToolName,
      thePropName)
```

Element	Description
theValue As String	Returns the default property value for the specified tool name and property name
theElement As Element	Element for which the default property value is being retrieved
theToolName As String	Name of the tool to which the property applies
thePropName As String	Name of the property being retrieved

Element.GetDefaultSetNames Method

Description

This method retrieves the names of the default property sets defined for the specified element and tool.

Syntax

```
Set theStringCollection = theElement.GetDefaultSetNames
       (theToolName)
```

Element	Description
theStringCollection As StringCollection	Returns the names of the default property sets defined for the given element and tool name
theElement As Element	Element whose default set names are being retrieved
theToolName As String	Name of the tool whose default set names are being retrieved

Element. GetIconIndex Method

Description

This method retrieves the index of the bitmap, in Rose's predefined set of browser icons, for the specified element.

Syntax

```
intIndex = objElement.GetIconIndex ( )
```

Element	Description
intIndex As Integer	Index of the specified element's browser icon
objElement As Element	Element whose icon index is being retrieved

Element.GetPropertyClassName Method

Description

This method retrieves the class name of a given element.

Syntax

```
theClassName = theElement.GetPropertyClassName ()
```

Element	Description
theClassName As String	Returns the class name for the given element
theElement as Element	Element whose class name is being retrieved

Element.GetPropertyValue Method

Description

This method retrieves the current value of a property of an element, given a property and tool name.

Syntax

```
theValue = theElement.GetPropertyValue (theToolName,
       thePropName)
```

Element	Description
theValue As String	Returns the current value for the given tool and property
theElement As Element	Element for which the property value is being retrieved
theToolName As String	Name of the tool for which a property value is being retrieved
thePropName As String	Name of the property whose value is being retrieved

Element.GetQualifiedName Method

Description

This method retrieves the qualified name of a model element.

The qualified name includes the names of the packages to which the element belongs. This allows the name to resolve to a specific class, since Rational Rose allows multiple classes of the same name to exist in a model, as long as they are in different packages.

For example:

- The qualified name of the SubsystemView Class is: Logical View::Physical Classes::SubsystemView
- The qualified name of the PathMap Class is: Logical View::Application Classes::PathMap

Syntax

`Set` `theName = theElement.`**`GetQualifiedName`** `()`

Element	**Description**
theName As String	Returns the qualified name of the element
theElement As Element	Element whose qualified name is being returned

Element.GetToolNames Method

Description

This method retrieves the names of the tools defined for the specified element.

Syntax

```
Set theStringCollection = theElement.GetToolNames
```

Element	Description
theStringCollection As StringCollection	Returns the names of the tools for the given element.
theElement As Element	Element whose tool names are being retrieved

Element.GetToolProperties Method

Description

This method retrieves the properties for the given element and tool name.

Syntax

```
Set thePropertyCollection = theElement.GetToolProperties
      (theToolName)
```

Element	Description
thePropertyCollection As PropertyCollection	Returns the collection of properties defined for the specified tool name and element
theElement As Element	Element whose tool properties are being retrieved
theToolName As String	Name of the tool whose properties are being retrieved

Element.GetUniqueId Method

Description

This method retrieves the unique ID for a model element. Each element in a model has a unique ID, which is set internally. You cannot set this value, but you can retrieve it.

Syntax

```
Set theUniqueID = theElement.GetUniqueID ()
```

Element	Description
theUniqueId As String	Returns the string value of the element's unique ID
theElement As Element	Element whose ID is being returned

Element.InheritProperty Method

Description

This method removes the overridden value from an element's property so that the default value is used. If there is no default value, then a call to the GetPropertyValue method on the inherited property returns an empty string.

Syntax

```
IsInherited = theElement.InheritProperty (theToolName,
        thePropName)
```

Element	Description
IsInherited as Boolean	Returns a value of True when the property is returned to its inherited (default) value
theElement As Element	Element to which the property belongs
theToolName As String	Name of the tool to which the property applies
thePropName As String	Name of the property whose value is being inherited

Element.IsDefaultProperty Method

Description

This method indicates whether the current value of a property is set to its default value.

Syntax

```
IsDefault = theElement.IsDefaultProperty (theToolName,
      thePropName)
```

Element	Description
IsDefault As Boolean	Returns a value of True if the current value of the property is set to its default value
theElement As Element	The model element whose property value is being checked
theToolName As String	Tool name to which the property applies
thePropName As String	Name of the property whose default status is being checked

Element.IsOverriddenProperty Method

Description

This method indicates whether the default value of a property is currently overridden by a different value.

Syntax

```
IsOverridden = theElement.IsOverriddenProperty
      (theToolName, thePropName)
```

Element	Description
IsOverridden As Boolean	Returns a value of True if the default value of a property is currently overridden
theElement As Element	The model element whose property value is being checked
theToolName As String	Tool name to which the property applies
thePropName As String	Name of the property whose overridden status is being checked

Element.OverrideProperty Method

Description

This method overrides the default value of an element's property. If the given property does not exist in the default set, a new string type property is created for this element only.

Syntax

IsOverridden = *theElement*.**OverrideProperty** (*theToolName*, *thePropName*, *theValue*)

Element	Description
IsOverrridden As Boolean	Returns a value of True when the property value is successfully overridden
theElement as Element	Element to which the property applies
theToolName As String	Name of the tool to which the property applies
thePropName As String	Name of the property whose default value is being overridden
theValue As String	Value being set in place of the default value

Element. RenderIconToClipboard Method

Description

This method renders the browser icon of the specified element to the Clipboard.

Syntax

blnRendered = *objElement*.**RenderIconToClipboard** ( )

Element	Description
blnRendered As Boolean	Returns the outcome of the rendering ■ If the icon is successfully rendered to the clipboard, this value is True ■ If the icon is not successfully rendered to the clipboard, this value is False
objElement as Element	Element whose browser icon is being rendered to the Clipboard

Element.SetCurrentPropertySetName Method

Description

This method specifies a given property set as the current property set for the element.

Syntax

```
IsCurrentSet = theElement.SetCurrentPropertySetName
      (theToolName, theSetName)
```

Element	Description
IsCurrentSet As Boolean	Returns a value of True when the given property set is set to the current property set for the element
theElement As Element	Element whose current property set is being set
theToolname As String	Name of the tool to which the property set applies
theSetName As String	Name of the property set to become the current set

Event Class

An event is an occurrence that causes a state transition. Use Event class methods and properties to define and control events that affect states and state transitions of objects in a model.

Event Class Properties

The following table summarizes the Event Class properties.

Table 67 Event Class Properties Summary

Property	Description
Element Properties	Inherits all Element Class properties
Arguments	Conditions affecting the event
GuardCondition	Defines a condition which, when true, causes the event to occur. As long as the condition remains false, the event will not occur.
Name	Name of the event

Event.Arguments Property

Description

Specifies the arguments for the event.

Syntax

`Event.Arguments`

Property Type

String

Event.GuardCondition Property

Description

Defines a condition which, when true, causes the event to occur. As long as the condition remains false, the event will not occur.

Syntax

`Event.GuardCondition`

Property Type

String

Event.Name Property

Description

Specifies the name of the event.

Syntax

Event.**Name**

Property Type

String

Event Class Methods

The following table summarizes the Event Class methods.

Table 68 Event Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
GetAction	Retrieves the action that corresponds to the event

Event.GetAction Method

Description

This method retrieves the action that corresponds to the specified event.

Syntax

Set *theAction* = *theEvent*.**GetAction** ()

Element	Description
theAction As Action	Returns the action that corresponds to the event
theEvent As Event	Event whose corresponding action is being retrieved

ExternalDocument Class

The ExternalDocument class exposes properties and methods that allow you to create external documents (reports) from within the Rational Rose environment. For example, you can start Word for Windows and output information from a Rational Rose model into a Word document.

ExternalDocument Class Properties

The following table summarizes the ExternalDocument Class properties.

Table 69 ExternalDocument Class Properties Summary

Property	**Description**
ParentCategory	Specifies the application to use to open the document
ParentDiagram	If attached to a diagram, specifies the diagram to which the external document is attached
ParentItem	If attached to a RoseItem, specifies the RoseItem to which the external document is attached
Path	Specifies the path to the document ***Note:*** *Path property and URL property are mutually exclusive. Do not specify both properties.*
URL	Specifies an internet document's URL ***Note:*** *Path property and URL property are mutually exclusive. Do not specify both properties.*

ExternalDocument.ParentCategory Property

Description

Specifies the category that contains the external document.

Note: *This property is read-only.*

Syntax

```
ExternalDocument.ParentCategory
```

Property Type

Category

ExternalDocument. ParentDiagram Property

Description

If attached to a diagram, this property specifies the diagram to which the external document is attached. If the external document is not attached to a diagram, this property returns **Nothing**. If this property returns **Nothing**, the specified ExternalDocument is attached to a RoseItem. To retrieve the RoseItem, use the ExternalDocument.ParentItem property.

Note: *This property is read-only.*

Syntax

```
Set objDiagram = objExternalDocument.ParentDiagram
```

Property Type

Diagram

ExternalDocument. ParentItem Property

Description

If attached to a RoseItem, this property specifies the RoseItem to which the external document is attached. If the external document is not attached to a RoseItem, this property returns **Nothing**. If this property returns **Nothing**, the specified ExternalDocument is attached to a Diagram. To retrieve the Diagram, use the ExternalDocument.ParentDiagram property.

Note: *This property is read-only.*

Syntax

```
Set objRoseItem = objExternalDocument.ParentItem
```

Property Type

RoseItem

ExternalDocument.Path Property

Description

Specifies the path to the external document.

Note: *An external document is created with a type parameter of either Path or URL. When accessing an external document, you must specify the correct property (Path or URL) or a runtime error will occur. For example, you cannot access an external document whose type is Path by specifying a URL.*

Syntax

```
ExternalDocument.Path
```

Property Type

String

ExternalDocument.URL Property

Description

Specifies the Universal Resource Locator (URL) of an internet document.

Note: *An external document is created with a type parameter of either Path or URL. When accessing an external document, you must specify the correct property (Path or URL), or a runtime error will occur. For example, you cannot access an external document whose type is URL by specifying a Path.*

Syntax

`ExternalDocument``.`**`URL`**

Property Type

String

ExternalDocument Class Methods

The following table summarizes the ExternalDocument Class methods.

Table 70 ExternalDocument Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
GetIconIndex	Retrieves the index of the specified external document's browser icon
IsURL	Checks whether the document has a URL
Open	Opens an external document
RenderIconToClipboard	Places the external document's browser icon on the Clipboard

ExternalDocument. GetIconIndex Method

Description

This method retrieves the index of the bitmap, in Rose's predefined set of browser icons, for the specified external document.

Syntax

intIndex **=** *objExternalDocument*.**GetIconIndex ( )**

Element	Description
intIndex As Integer	Index of the specified external document's browser icon
objExternalDocument As ExternalDocument	External document whose icon index is being retrieved

ExternalDocument.IsURL Method

Description

Checks whether the document is an internet document and therefore has a universal resource locator (URL).

Syntax

IsURL = *theObject*.**IsURL** ()

Element	Description
IsURL As Boolean	Returns a value of true if the object has a URL
theObject As ExternalDocument	Contains the document being checked

ExternalDocument.Open Method

Description

Opens an external document based on a specified application path.

If you do not specify an application path, the Rational Rose application attempts to locate and launch the application based on the external document's type (file extension).

For example, if the ExternalDocument is linked to a file with the .txt extension, and you have associated .txt files with the Notepad application, Rational Rose attempts to locate and start Notepad and opens the .txt file that contains the external document.

Syntax

IsOpen = *theObject*.**Open** (*AppPath*)

Element	Description
IsOpen As Boolean	Returns a value of true when the specified document is successfully opened
theObject As ExternalDocument	Document being opened
AppPath As String	Path to the application executable being used to open the document. **Note:** *You can specify any appropriate application to open the document. For example, you can use Word or WordPad to open a .doc file.*

ExternalDocument. RenderIconToClipboard Method

Description

This method renders the browser icon of the specified external document to the Clipboard.

Syntax

```
blnRendered = objExternalDocument.RenderIconToClipboard ( )
```

Element	Description
blnRendered As Boolean	Returns the outcome of the rendering ■ If the icon is successfully rendered to the clipboard, this value is True ■ If the icon is not successfully rendered to the clipboard, this value is False
objExternalDocument As ExternalDocument	External document whose browser icon is being rendered to the Clipboard

HasRelationship Class

The Has Relationship indicates a containment or aggregation relationship between classes. The HasRelationship class exposes properties and methods that:

- Determine the characteristics of Has Relationships in a model (for example, client and supplier cardinality and whether the relationship is static)
- Allow you to retrieve Has Relationships

HasRelationship Class Properties

The following table summarizes the HasRelationship Class properties.

Table 71 HasRelationship Class Properties Summary

Property	Description
RoseItem Properties	Inherits all RoseItem properties
ClientCardinality	Client cardinality of the HasRelationship
Containment	Defines class containment for a HasRelationship
ExportControl	Controls HasRelationship visibility
Static	Determines whether the relationship is static
SupplierCardinality	Supplier cardinality of the HasRelationship

HasRelationship.ClientCardinality Property

Description

Indicates the number of instances of the HasRelationship that are allowed.

Syntax

HasRelationship.**ClientCardinality**

Property Type

String

HasRelationship.Containment Property

Description

The Containment property is a rich data type. The following table describes the valid forms of expressing the Containment rich data type.

Table 72 HasRelationship.Containment Rich Data Types

Rich Data Type	**Description**
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Unspecified ■ ByValue ■ ByReference

Note: *This property is read-only.*

Syntax

`Object`.**`Containment`**

Property Type

Containment

HasRelationship.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the HasRelationship object. The following table describes the valid forms of expressing the ExportControl rich data type for the HasRelationship class.

Table 73 HasRelationship.Export Control Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible string values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControl property is read-only. Its Value and Name, however, are read/write.*

Syntax

```
theHasRelationship.ExportControl.Name = "PrivateAccess"

theHasRelationship.ExportControl.Value = 2

theNameStr = theHasRelationship.ExportControl.Name

theValue = theHasRelationship.ExportControl.Value
```

Property Type

RichType or HasRelationshipExportControl

HasRelationship.Static Property

Description

Indicates whether the HasRelationship is static.

Syntax

`Object``.`**`Static`**

Property Type

Boolean

HasRelationship.SupplierCardinality Property

Description

Indicates the number of suppliers allowable for the Has Relationship.

Syntax

`HasRelationship``.`**`SupplierCardinality`**

Property Type

String

HasRelationship Class Methods

The following table summarizes the HasRelationship Class methods.

Table 74 Has Relationship Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods

InheritRelation Class

The Inherit Relation indicates a hierarchical relationship between classes in which one class shares the structure and/or behavior of another class. The InheritRelation class exposes properties and methods that:

- Determine the characteristics of Inherit Relations between classes
- Allow you to retrieve Inherit Relations

InheritRelation Class Properties

The following table summarizes the InheritRelation Class properties.

Table 75 InheritRelation Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Properties	Inherits all Relation class properties
ClassRelation Class Properties	Inherits all ClassRelation class properties
ExportControl	Controls InheritRelation visibility
FriendshipRequired	Indicates whether friendship is required by the InheritRelation
Virtual	Indicates whether the relation is virtual

InheritRelation.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the InheritRelation object. The following table describes the valid forms of expressing the ExportControl rich data type for the InheritRelation class.

Table 76 InheritRelation.Export Control Rich Data Types

Rich Data Type	**Description**
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible string values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControl property is read-only. Its Value and Name, however, are read/write.*

Syntax

theInheritRelation.**ExportControl.Name** = **"PrivateAccess"**

theInheritRelation.**ExportControl.Value** = **2**

theNameStr = *theInheritRelation*.**ExportControl.Name**

theValue = *theInheritRelation*.**ExportControl.Value**

Property Type

RichType or InheritRelationship.ExportControl

InheritRelation.FriendshipRequired Property

Description

Indicates whether the Inherit Relation requires friendship. Friendship can be required between a supplier and a client in the relationship.

Syntax

InheritRelation.**FriendshipRequired**

Property Type

Boolean

InheritRelation.Virtual Property

Description

Indicates whether the Inherit Relation is virtual.

Syntax

InheritRelation.**Virtual**

Property Type

Boolean

InheritRelation Class Methods

The following table summarizes the InheritRelation Class methods.

Table 77 InheritRelation Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
Relation Methods	Inherits all Relation class methods
ClassRelation Methods	Inherits all ClassRelation class methods

InstanceView Class

The instance view class inherits the RoseItemView properties and methods that determine the size and placement of an instance view. It also allows you to retrieve the instance object itself from the instance view.

InstanceView Class Properties

The following table summarizes the InstanceView Class properties.

Table 78 InstanceView Class Properties Summary

Property	**Description**
Element properties	Inherits all Element Class properties
RoseItemView properties	Inherits all RoseItemView Class properties

InstanceView Class Methods

The following table summarizes the InstanceView Class methods.

Table 79 InstanceView Class Methods Summary

Method	**Description**
Element properties	Inherits all Element methods
RoseItemView properties	Inherits all RoseItemView methods
GetInstance	Returns the instance object represented by the instance view

InstanceView.GetInstance Method

Description

This method retrieves the instance represented by the instance view.

Syntax

Set *theInstance* = *theInstanceView*.**GetObject** ()

Element	Description
theInstance As ObjectInstance	Returns the object instance represented by the instanceview
theInstanceView As InstanceView	InstanceView whose corresponding instance is being retrieved

InstantiateRelation Class

The InstantiateRelation class allows you to get the instantiate relationships that a class has. This is especially useful in code and documentation generation.

InstantiateRelation Class Properties

The following table summarizes the InstantiateRelation Class properties.

Table 80 InstantiateRelation Class Properties Summary

Property	Description
Element properties	Inherits Element Properties
RoseItem properties	Inherits RoseItem Properties
Relation properties	Inherits Relation properties

InstantiateRelation Class Methods

The following table summarizes the InstantiateRelation Class methods.

Table 81 InstantiateRelation Class Methods Summary

Method	**Description**
Element methods	Inherits Element methods
RoseItem methods	Inherits RoseItem methods
RoseObject methods	Inherits RoseObject methods
Relation methods	Inherits Relation methods
ClassRelation Methods	Inherits ClassRelation methods

LineVertex Class

The LineVertex class defines objects that are the points where one line segment of an association or relation view ends and the next line segment begins. To work with a collection of LineVertex objects, use the RoseItemView property, LineVertices.

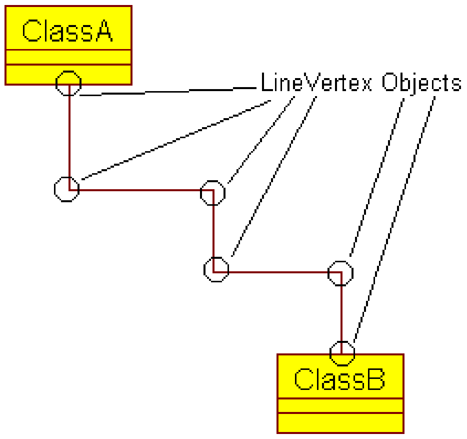

Figure 2 LineVertex Objects

LineVertex Class Properties

There are no LineVertex class properties.

LineVertex Class Methods

The following table summarizes the LineVertex Class methods.

Table 82 LineVertex Class Methods Summary

Method	**Description**
RoseObject Class Methods	Inherits all RoseObject class methods
GetXPosition	Returns the X coordinate for a vertex
GetYPosition	Returns the Y coordinate for a vertex

LineVertex.GetXPosition Method

Description

This method returns the X coordinate of the point where an association or relation view line segment begins or ends.

Syntax

theXPos = *theLineVertex*.**GetXPosition** ()

Element	**Description**
theXPos As Integer	X coordinate of the specified LineVertex in Logical Units relative to the upper left hand corner of the image rendered by the Diagram RenderToClipboard method
theLineVertex As LineVertex	LineVertex whose X coordinate is being retrieved

LineVertex.GetYPosition Method

Description

This method returns the Y coordinate of the point where an association or relation view line segment begins or ends.

Syntax

theYPos = *theLineVertex*.**GetYPosition** ()

Element	**Description**
theYPos As Integer	Y coordinate of the specified LineVertex in Logical Units relative to the upper left hand corner of the image rendered by the Diagram RenderToClipboard method
theLineVertex As LineVertex	LineVertex whose Y coordinate is being retrieved

LineVertex Sample Rational Rose Script

The following sample Rational Rose Script gets the LineVertex collection for a selected association in a particular class diagram. The script then iterates through the collection and prints the x and y coordinates of each LineVertex.

```
Sub Main
   Dim theCat As Category
   Dim theDiags As ClassDiagramCollection
   Dim theDiagram As ClassDiagram
   Dim theItemViews As ItemViewCollection
   Dim theView As RoseItemView
   Dim theVertices As LineVertexCollection
   Dim theVert As LineVertex

   'Get the class diagram named "Main".
   Set theCat = RoseApp.CurrentModel.RootCategory
   Set theDiags = theCat.ClassDiagrams
   Set theDiagram = theDiags.GetFirst ("Main")

   'Iterate through all ItemViews. For each selected ItemView,
   'get the line vertices, iterate through them and print out
   'their xand y coordinates.
   Set theItemViews = theDiagram.ItemViews
   Total = theItemViews.Count
   For i = 1 To Total
      Set theView = theItemViews.GetAt (i)
      isSelected = theView.IsSelected ()
      If isSelected Then
         Set theVertices = theView.LineVertices
         TotVerts = theVertices.Count
         For j = 1 To TotVerts
            Print "Line Vertex ";j;"  ";
            Set theVert = theVertices.GetAt (j)
            x = theVert.GetXPosition()
            y = theVert.GetYPosition ()
            Print "(";x;", ";y;")"
         Next j
      End If
   Next i
End Sub
```

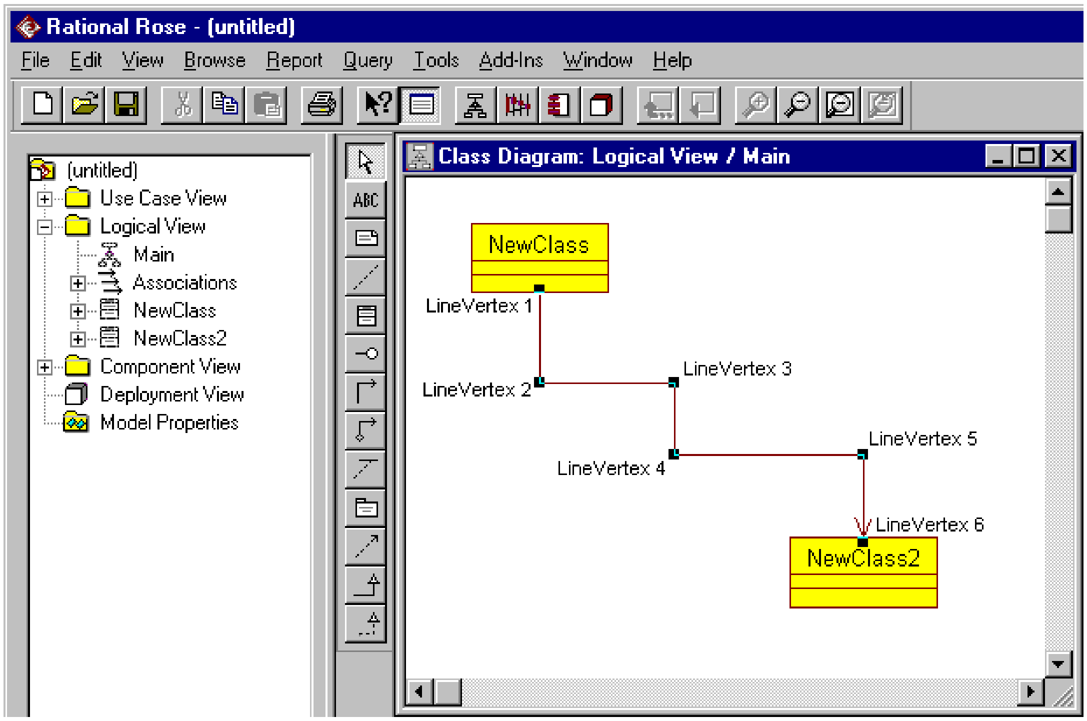

Figure 3 Example: Retrieving Line Vertices

Output

```
Line Vertex  1   ( 238 ,  207 )
Line Vertex  2   ( 238 ,  365 )
Line Vertex  3   ( 478 ,  365 )
Line Vertex  4   ( 478 ,  493 )
Line Vertex  5   ( 816 ,  493 )
Line Vertex  6   ( 816 ,  640 )
```

Link Class

Objects interact through their links to other objects. A link is an instance of an association, in the same way that an object is an instance of a class.

Link properties and methods allow you to define links between objects and determine the nature of the objects' associations.

Link Class Properties

The following table summarizes the Link Class properties.

Table 83 Link Class Properties Summary

Property	**Description**
Element properties	Inherits all Element properties
RoseItem properties	Inherits all RoseItem properties
LinkRole1	Defines an object instance as a Role1 link
LinkRole2	Defines an object instance as a Role2 link
LinkRole1Shared	Defines an object instance as a Role1 link with shared visibility
LinkRole2Shared	Defines an object instance as a Role2 link with shared visibility
LinkRole1Visibility	Defines an object instance as a Role1 link and determines its visibility type
LinkRole2Visibility	Defines an object instance as a Role2 link and determines its visibility type

Link.LinkRole1 Property

Description

Defines an object instance as a Role1 link.

Note: *This property is read-only.*

Syntax

`Link`.**`LinkRole1`**

Property Type

ObjectInstance

Link.LinkRole2 Property

Description

Defines an object instance as a Role2 link.

Note: *This property is read-only.*

Syntax

`Link`.**`LinkRole2`**

Property Type

ObjectInstance

Link.LinkRole1Shared Property

Description

Defines an object instance as a Role1 link with shared visibility.

Syntax

`Link`.**`LinkRole1Shared`**

Property Type

Boolean

Link.LinkRole2Shared Property

Description

Defines an object instance as a Role2 link with shared visibility.

Syntax

`Link`.**`LinkRole2Shared`**

Property Type

Boolean

Link.LinkRole1Visibility Property

Description

Defines an object instance as a Role1 link with a specified visibility type. Visibility type is a rich data type. The following table describes the valid forms of expressing the Role1 visibility for the Link class.

Table 84 Link.LinkRole1Visibility Rich Data Types

Rich Data Type	Description
Type As LinkVisibility	Valid types are: ■ Unspecified Not a specified type ■ Field Indicates that the client object operates on one of its own data members ■ Parameters Indicates that the supplier object is visible to the client object because it is a parameter for one of the client's operations ■ Local Indicates that the supplier object is local to an operation of the client object ■ Global Indicates that the supplier is global to the client object

Note: *This property is read-only.*

Syntax

`Link``.`**`LinkRole1Visibility`**

Property Type

LinkVisibility

Link.LinkRole2Visibility Property

Description

Defines an object instance as a Role2 link with a specified visibility type. Visibility type is a rich data type. The following table describes the valid forms of expressing the Role2 visibility for the Link class.

Table 85 Link.LinkRose2Visibility Rich Data Types

Rich Data Type	Description
Type As LinkVisibility	Valid types are: ■ Unspecified Not a specified type ■ Field Indicates that the client object operates on one of its own data members ■ Parameters Indicates that the supplier object is visible to the client object because it is a parameter for one of the client's operations. ■ Local Indicates that the supplier object is local to an operation of the client object ■ Global Indicates that the supplier is global to the client object

Note: *This property is read-only.*

Syntax

`Link``.`**`LinkRole2Visibility`**

Property Type

LinkVisibility

Link Class Methods

The following table summarizes the Link Class methods.

Table 86   Link Class Methods Summary

Method	**Description**
Element methods	Inherits all Element class methods
RoseItem methods	Inherits all RoseItem class methods
AddMessageTo	Adds a message to the link
AssignAssociation	Assigns an association to the link
DeleteMessage	Deletes a message from the link
GetAssociation	Retrieves the association that corresponds to the link
GetMessages	Retrieves the messages carried by the link
UnAssignAssociation	Removes an association assignment from a link

Link.AddMessageTo Method

Description

Adds a message to the specified link.

Syntax

Set *theMessage* = *theLink*.**AddMessageTo** (*theName, toInstance, theSequenceNumber*)

Element	Description
theMessage As Message	Returns the message added to the link If the object receiving the message is not an end of this link, the method returns a Nothing object
theLink As Link	Link to which the message is being added
theName As String	Name of the message being added
toInstance As ObjectInstance	Object instance to receive the message
theSequenceNumber As Integer	Position of the message relative to other messages in the diagram For example, if theSequence = 3, the message will be the third message in the diagram

Link.AssignAssociation Method

Description

Assigns an association to a link.

Syntax

`IsAssigned` = *`theLink`*.**`AssignAssociation`** (*`theAssoc`*)

Element	**Description**
IsAssigned As Boolean	Returns a value of True when the association is successfully assigned
theLink As Link	Link to which the association is being assigned
theAssoc As Association	Association being assigned to the link

Link.DeleteMessage Method

Description

Deletes a message from the specified link.

Syntax

`IsDeleted` = *`theLink`*.**`DeleteMessage`** (*`theMessage`*)

Element	**Description**
IsDeleted As Boolean	Returns a value of True when the message is successfully deleted
theLink As Link	Link from which to delete the message
theMessage As Message	Message to delete from the link

Link.GetAssociation Method

Description

This method retrieves the association that corresponds to the link.

Syntax

Set *theAssociation* = *theLink*.**GetAssociation** ()

Element	Description
theAssociation As Association	Returns the association that corresponds to the link
theLink As Link	Link from which to retrieve the association

Link.GetMessages Method

Description

Retrieves the collection of messages belonging to the specified link.

Syntax

Set *theMessages*= *theObject*.**GetMessages** ()

Element	Description
theMessages As MessageCollection	Returns the messages belonging to the link
theObject As Link	Link whose message collection is being retrieved

Link.UnAssignAssociation Method

Description

Removes an association assignment from a link.

Syntax

`IsUnAssigned` `=` *`theLink`*`.`**`UnAssignAssociation`** `()`

Element	**Description**
IsUnAssigned As Boolean	Returns a value of True when the association assignment is successfully removed
theLink As Link	Link from which the association assignment is being removed

MenuState Enumeration

MenuState is an enumeration that defines values corresponding to the status of shortcut menu options. MenuState can be used in the MenuState ContextMenuItem property.

The following table describes the valid values for the MenuState enumeration.

Table 87 MenuState Enumeration Valid Values

Value	**Integer Value**	**Description**
rsDisabled	0	Use to disable a shortcut menu option For example, use **rsDisabled** in the MenuState ContextMenuItem property. If the menu option was checked it remains checked, though disabled. If the menu option was unchecked, it remains unchecked, though disabled.
rsDisabledAndChecked	2	Use to disable and place a check mark in front of a shortcut menu option For example, use **rsDisabledAndChecked** in the MenuState ContextMenuItem property. If the menu option was checked it remains checked, though disabled. If the menu option was unchecked, it is checked and disabled.
rsDisabledAndUnchecked	3	Use to disable and remove, if necessary, the check mark in front of a shortcut menu option For example, use **rsDisabledAndUnchecked** in the MenuState ContextMenuItem property. If the menu option was checked it is unchecked and disabled. If the menu option was unchecked, it remains unchecked.

Value	Integer Value	Description
rsEnabled	1	Use to enable a shortcut menu option For example, use **rsEnabled** in the MenuState ContextMenuItem property. If the menu option was checked it remains checked. If the menu option was unchecked, it remains unchecked.
rsEnabledAndChecked	4	Use to enable and place a check mark in front of a shortcut menu option For example, use **rsEnabledAndChecked** in the MenuStateContextMenuItem property. If the menu option was checked it remains checked. If the menu option was unchecked, it is checked.
rsEnabledAndUnchecked		Use to enable and remove, if necessary, the check mark in front of a shortcut menu option For example, use **rsEnabledAndUnchecked** in the MenuStateContextMenuItem property. If the menu option was checked, it is unchecked. If the menu option was unchecked, it remains unchecked.

Note: *Rational Rose Automation users may use the value (e.g.,* ***rsDisabled****) in their methods. Rational Rose Script users must use the integer value (e.g., 0) in their methods.*

Message Class

Messages define the interaction between objects. The message class inherits all of the RoseItem properties and methods. In addition message class methods allow you to retrieve message sender and receiver, along with other message-specific information.

Message Class Properties

The following table summarizes the Message Class properties.

Table 88 Message Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Frequency	Specifies the frequency of the message; that is whether the message is sent once or periodically
Synchronization	Specifies the concurrency semantics for the message

Message.Frequency Property

Description

Specifies whether the message is to be sent one time only or sent periodically.

Syntax

Message.**Frequency**

Property Type

Integer

Values

0 = Send one time only

1 = Send periodically

Message.Synchronization Property

Description

Specifies the concurrency semantics for the message.

Syntax

`Message`.**`Synchronization`**

Property Type

Integer

Values

0 = Simple

1 = Synchronous

2 = Balking

3 = Timeout

4 = Asynchronous

Message Class Methods

The following table summarizes the Message Class methods.

Table 89 Message Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
GetLink	Retrieves the link associated with the message
GetOperation	Retrieves the operation associated with the message
GetReceiverObject	Retrieves the object that received the message
GetSenderObject	Retrieves the object that sent the message
GetSequenceInformation	Returns the message's sequence number
IsMessageToSelf	Indicates whether this is a message to self
IsOperation	Indicates whether the message is an operation

Message.GetLink Method

Description

Retrieves the link associated with the message.

Syntax

```
Set theLink = theMessage.GetLink ()
```

Element	**Description**
theLink As Link	Retrieves the link associated with the message
theMessage As Message	Message whose link is being retrieved

Message.GetOperation Method

Description

Retrieves the operation associated with the message.

Syntax

Set *theOperation* = *theMessage*.**GetOperation** ()

Element	**Description**
theOperation As Operation	Retrieves the operation associated with the message
theMessage As Message	Message whose operation is being retrieved

Message.GetReceiverObject Method

Description

This method returns the receiver object belonging to the message.

Syntax

Set *theObject* = *theMessage*.**GetReceiverObject** ()

Element	**Description**
theObject As ObjectInstance	Retrieves the receiver object belonging to the message
theMessage As Message	Message whose receiver object is being retrieved

Message.GetSenderObject Method

Description

This method returns the sender object belonging to the message.

Syntax

Set *theObject* = *theMessage*.**GetSenderObject** ()

Element	Description
theObject As ObjectInstance	Retrieves the sender object belonging to the message
theMessage As Message	Message whose sender object is being retrieved

Message. GetSequenceInformation Method

Description

This method returns a string containing the message's sequence number.

Top level numbering examples:

1
2
3

Hierarchical numbering examples:

1.
1.1.
1.2.1.

Syntax

```
strSequenceNumber = objMessage.GetSequenceInformation ( )
```

Element	Description
strSequenceNumber As String	Message's sequence number
objMessage As Message	Message whose sequence number is being retrieved

Message.IsMessageToSelf Method

Description

Indicates whether the message is a message to itself.

Syntax

IsMsg = *theObject*.**IsMessageToSelf** ()

Element	Description
IsMsg As Boolean	Returns a value of True if the message is a message to itself
theObject As Message	Message being checked

Message.IsOperation Method

Description

Indicates whether the message is an operation.

Syntax

IsOp = *theObject*.**IsOperation** ()

Element	Description
IsOp As Boolean	Returns a value of True if the message is an operation
theObject As Message	Message being checked

Model Class

Once you use the application class methods to set the current model, the model class provides properties and methods that allow you to work with the objects in that model.

For example, you can:

- Add objects (classes, categories, relationships, processors, devices, diagrams, etc.) to the model
- Retrieve objects from the model
- Delete objects from the model

Model Class Properties

The following table summarizes the Property Class properties.

Table 90 Model Class Properties Summary

Property	Description
Element	Inherits all element class properties
RoseItem	Inherits all RoseItem class properties
DefaultLanguage	Any valid string to be assigned as the default language to classes and components
DefaultProperties	Default properties belonging to the model
DeploymentDiagram	Deployment diagram associated with the model
DeploymentUnit	Returns the ControllableUnit form of the deployment diagram
Notation	Notation used by the model
RootCategory	Category named <Top Level> in Rose. RootCategory corresponds to the model's logical view. This value can be retrieved, but not set.
RootSubsystem	Subsystem named <Top Level> in Rose. RootSubsystem corresponds to the model's component view. This value can be retrieved, but not set.
RootUseCaseCategory	Root category to which the use cases belong. RootUseCaseCategory corresponds to the model's UseCase view. This value can be retrieved, but not set.
UseCases	Contains the use cases belonging to the model

Model.DefaultLanguage Property

Description

This property is any valid string to be assigned as the default language to all subsequently created classes and components until the default language is set to something else.

Example

If you set *`myModel`*.**`DefaultLanguage`** = "SomeText", then "SomeText" is used as the default language for the next classes and components that are created. If you set *`myModel`*.**`DefaultLanguage`** = "C++", then "C++" is used as the default language for the next classes and components that are created.

Syntax

`theLanguage$` = *`myModel`*.**`DefaultLanguage`**

`myModel`.**`DefaultLanguage`** = **`"Analysis"`**

Property Type

String

Model.DefaultProperties Property

Description

Collection of default properties belonging to the model.

Note: *This property is read-only.*

Syntax

`Model`.**`DefaultProperties`**

Property Type

DefaultModelProperties

Model.DeploymentDiagram Property

Description

Specifies a deployment diagram belonging to the model.

Note: *This property is read-only.*

Syntax

`Model.DeploymentDiagram`

Property Type

DeploymentDiagram

Model.DeploymentUnit Property

Description

This property corresponds to the controllable unit form of the Deployment Diagram. This allows you to control, uncontrol, load, or unload the Deployment Diagram.

Note: *This property is read-only.*

Syntax

`myModel.DeploymentUnit`

Property Type

DeploymentUnit

Model.Notation Property

Description

This property specifies the Notation used by the model (e.g., Booch).

Syntax

`x = myModel.Notation`

`myModel.Notation = 0`

`myModel.Notation = BoochNotation`

Property Type

NotationTypes Enum or Integer

Model.RootCategory Property

Description

Category named <Top Level> in Rose. RootCategory corresponds to the model's logical view. This value can be retrieved, but not set.

Note: *This property is read-only.*

Syntax

`Model.RootCategory`

Property Type

Category

Model.RootSubsystem Property

Description

Subsystem named <Top Level> in Rose. RootSubsystem corresponds to the model's component view. This value can be retrieved, but not set.

Note: *This property is read-only.*

Syntax

`Model`.**`RootSubsystem`**

Property Type

Subsystem

Model.RootUseCaseCategory Property

Description

Root category to which the use cases belong. RootUseCaseCategory corresponds to the model's UseCase view. This value can be retrieved, but not set.

Note: *This property is read-only.*

Syntax

`Model`.**`RootUseCaseCategory`**

Property Type

Category

Model.UseCases Property

Description

Specifies the collection that contains the use cases that belong to the model.

Note: *This property is read-only.*

Syntax

`Model`.**`UseCases`**

Property Type

UseCaseCollection

Model Class Methods

The following table summarizes the Model Class methods.

Table 91 Model Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element Class Methods
RoseItem Methods	Inherits all RoseItem Class Methods
ControllableUnit Methods	Inherits all ControllableUnit Class Methods
Package Class Methods	Inherits all Package Class methods
AddDevice	Adds a device to a model
AddProcessor	Adds a processor to the model
DeleteDevice	Deletes a device from a model
DeleteProcessor	Deletes a processor from the model
FindCategories	Finds a collection of categories belonging to the model
FindCategoryWithID	Finds a specific category belonging to the model

Method	Description
FindClasses	Finds a collection of classes belonging to the model
FindClassWithID	Finds a specific class belonging to the model
FindDiagramWithID	Finds a specific diagram using the diagram's unique ID
FindItems	Finds a collection of items belonging to the model
FindItemWithID	Finds a specific item belonging to the model
GetActiveDiagram	Retrieves the model's currently active diagram
GetAllCategories	Retrieves all categories belonging to the model
GetAllClasses	Retrieves all classes belonging to all categories in the model
GetAllDevices	Retrieves all devices belonging to the model
GetAllModules	Retrieves all modules belonging to the model
GetAllProcessors	Retrieves all processors belonging to the model
GetAllSubsystems	Retrieves all subsystems belonging to the model
GetAllUseCases	Retrieves all use cases belonging to the model
GetObject	Retrieves the OLE object associated with a specified model object. ***Note:*** *This method is only valid for Rose Script; it does not exist in Rose Automation*

Method	Description
GetSelectedCategories	Retrieves all currently selected categories from the model
GetSelectedClasses	Retrieves all currently selected classes from the model
GetSelectedDiagrams	Retrieves all diagrams selected in the browser
GetSelectedExternalDocuments	Retrieves all currently selected external documents from the model
GetSelectedItems	Returns a collection of all items selected in the current model
GetSelectedModules	Retrieves all currently selected modules from the model
GetSelectedSubsystems	Retrieves all currently selected subsystems from the model
GetSelectedUseCases	Retrieves all currently selected use cases from the model
Import	Imports a specified package (Category) or subsystem into a model
LoadControlledUnits	Loads a collection of controlled units
ResolveReferences	Fixes unresolved references

Model.AddDevice Method

Description

This method creates a new device and adds it to a model.

Syntax

Set *theDevice* = *theObject*.**AddDevice** (*theName*)

Element	**Description**
theDevice As Device	Returns the newly created device
theObject As Model	Instance of the model to which the device is being added
theName As String	Name of the device being added to the model

Model.AddProcessor Method

Description

This method creates a new processor and adds it to a model.

Syntax

Set *theProcessor* = *theObject*.**AddProcessor** (*theName*)

Element	**Description**
theProcessor As Processor	Returns the processor being added to the model
theObject As Model	Instance of the Processor being added to the model
theName As String	Name of the Processor being added to the model

Model.DeleteDevice Method

Description

This method deletes a device from a model.

Syntax

`Deleted = theObject.DeleteDevice (theDevice)`

Element	Description
Deleted As Boolean	Returns a value of True when the device is deleted
theObject As Model	Instance of the model from which the device is being deleted
theDevice As Device	Instance of the device being deleted

Model.DeleteProcessor Method

Description

This method deletes a processor from a model.

Syntax

`Deleted = theObject.DeleteProcessor (theProcessor)`

Element	Description
Deleted As Boolean	Returns a value of True when the processor is deleted from the model
theObject As Model	Instance of the model from which the processor is being deleted
theProcessor As Processor	Instance of the processor being deleted

Model.FindCategories Method

Description

This method returns a collection of categories belonging to the model.

Syntax

```
Set theCategoryCollection = theObject.FindCategories
       (theCategoryName)
```

Element	Description
theCategoryCollection As CategoryCollection	Returns a collection of categories that match the given category name
theObject As Model	Model that contains the categories
theCategoryName As String	Name of the category for which to search in the model

Model.FindCategoryWithID Method

Description

This method returns a specific category given the category's unique ID.

Syntax

```
Set theCategory = theObject.FindCategoryWithID (theUniqueID)
```

Element	Description
theCategory As Category	Returns the category that corresponds to the given UniqueID
theObject As Model	Model that contains the category
theUniqueID As String	UniqueID of the category for which to search

Model.FindClasses Method

Description

This method returns a collection of classes belonging to the model.

Syntax

```
Set theClassCollection = theObject.FindClasses
       (theClassName)
```

Element	Description
theClassCollection As ClassCollection	Returns a collection of classes that match the given class name
theObject As Model	Model that contains the classes
theClassName As String	Name of the class for which to search in the model

Model.FindClassWithID Method

Description

This method returns a specific class given the class's unique ID.

Syntax

```
Set theClass = theObject.FindClassWithID (theUniqueID)
```

Element	Description
theClass As Class	Returns the Class that corresponds to the given UniqueID
theObject As Model	Model that contains the Class
theUniqueID As String	UniqueID of the Class for which to search

Model. FindDiagramWithID Method

Description

This method retrieves the diagram given its unique internal Rose identification.

Syntax

```
Set objDiagram = objModel.FindDiagramWithID (strUniqueID)
```

Element	Description
objDiagram As Diagram	Returns the diagram that corresponds to the given unique ID
objModel As Model	Model containing the diagram
strUniqueID As String	Unique ID of the diagram to retrieve

Model.FindItems Method

Description

This method returns a collection of items belonging to the model.

Syntax

```
Set theItemCollection = theObject.FindItems (theItemName)
```

Element	Description
theItemCollection As ItemCollection	Returns a collection of items that match the given item name
theObject As Model	Model that contains the items
theItemName As String	Name of the item for which to search in the model

Model.FindItemWithID Method

Description

This method returns a specific item given the item's unique ID.

Syntax

`Set` *`theItem`* `=` *`theObject`*`.`**`FindItemWithID`** `(`*`theUniqueID`*`)`

Element	**Description**
theItem As Item	Returns the item that corresponds to the given UniqueID
theObject As Model	Model that contains the item
theUniqueID As String	UniqueID of the item for which to search

Model.GetActiveDiagram Method

Description

This method returns the currently active diagram from the current model. The active diagram is the window in Rational Rose which currently has the focus.

Syntax

`Set` *`theDiagram`* `=` *`theModel`*`.`**`GetActiveDiagram`** `()`

Element	**Description**
theDiagram As Diagram	Returns the currently active Rational Rose diagram from the model. Returns nothing if a window that is not a diagram, such as a script window or the browser, has the focus.
theModel As Model	Instance of the model from which the diagram is being retrieved

Model.GetAllCategories Method

Description

This method returns all categories belonging to the model.

Syntax

Set *theCategories* = *theObject*.**GetAllCategories** ()

Element	**Description**
theCategories As CategoryCollection	Returns the collection of categories retrieved from the model
theObject As Model	Instance of the model from which categories are being retrieved

Model.GetAllClasses Method

Description

This method returns all classes belonging to all categories in the model.

Syntax

Set *theClasses* = *theObject*.**GetAllClasses** ()

Element	**Description**
theClasses *As Class Collection*	Returns the collection of classes retrieved from the model
theObject As Model	Instance of the model from which classes are being retrieved

Model.GetAllDevices Method

Description

This method returns all devices belonging to the model.

Syntax

```
Set theDevices = theObject.GetAllDevices ()
```

Element	Description
theDevices As DeviceCollection	Returns the collection of devices retrieved from the model
theObject AsModel	Instance of the model from which devices are being retrieved

Model.GetAllModules Method

Description

This method returns all modules belonging to the model.

Syntax

```
Set theModules = theObject.GetAllModules ()
```

Element	Description
theModules As ModuleCollection	Returns the collection of modules retrieved from the model
theObject As Model	Instance of the model from which modules are being retrieved

Model.GetAllProcessors Method

Description

This method returns all processors belonging to the model.

Syntax

```
Set theProcessors = theObject.GetAllProcessors ()
```

Element	Description
theProcessors As ProcessorCollection	Returns the collection of processors retrieved from the model
theObject As Model	Instance of the model from which processors are being retrieved

Model.GetAllSubsystems Method

Description

This method returns all subsystems belonging to the model.

Syntax

```
Set theSubsystems = theObject.GetAllSubsystems ()
```

Element	Description
theSubsystems As SubsystemCollection	Returns the collection of subsystems retrieved from the model
theObject As Model	Instance of the model from which subsystems are being retrieved

Model.GetAllUseCases Method

Description

This method returns all use cases belonging to the model.

Syntax

```
Set theUseCases = theObject.GetAllUseCases ()
```

Element	Description
theUseCases As UseCaseCollection	Returns the collection of use cases retrieved from the model
theObject AsModel	Instance of the model from which use cases are being retrieved

Model.GetObject Method

Description

This method retrieves the OLE object associated with a specified model object.

Note: *This method is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

```
Set theOLEObject= theObject.GetObject ()
```

Element	Description
theOLEObject As Object	Returns the OLE automation interface object associated with the specified object
theObject As Model	Instance of the object whose OLE automation interface object is being returned

Model.GetSelectedCategories Method

Description

This method returns all categories selected in the current model.

Syntax

Set *theCategories* = *theObject*.**GetSelectedCategories** ()

Element	Description
theCategories As CategoryCollection	Returns the collection of categories currently selected in the model
theObject As Model	Instance of the model from which categories are being retrieved

Model.GetSelectedClasses Method

Description

This method returns all classes selected in the current model.

Syntax

Set *theClasses* = *theObject*.**GetSelectedClasses** ()

Element	Description
theClasses As ClassCollection	Returns the collection of classes currently selected in the model
theObject As Model	Instance of the model from which classes are being retrieved

Model.GetSelectedDiagrams Method

Description

This method returns all diagrams selected in the browser.

Syntax

```
Set colDiagrams = objModel.GetSelectedDiagrams ()
```

Element	Description
colDiagrams As DiagramCollection	Collection of diagrams currently selected in the browser
objModel As Model	Model from which diagrams are being retrieved

Model.GetSelectedExternalDocuments Method

Description

This method retrieves all external documents selected in the current model.

Syntax

```
Set theExternalDocs = theModel.GetSelectedExternalDocuments
    ()
```

Element	Description
theExternalDocs As ExternalDocumentCollection	Returns the collection of external documents currently selected in the model
theModel As Model	Instance of the model from which external documents are being retrieved

Model.GetSelectedItems Method

Description

This method returns the collection of all RoseItems selected in the current model. These items may be classes, components, packages, etc. GetSelectedItems returns all selected RoseItem objects regardless of whether they are selected in the browser or the currently active diagram. This method gives you the flexibility to work with different types of selected items (e.g., packages and classes) at the same time. This is instead of having to separate different types of items and then work with each type (e.g., GetSelectedClasses and work with the classes, then GetSelectedCategories and work with the packages).

Syntax

`Set` *`theItemCollection`* = *`myModel`*.**`GetSelectedItems`** ( )

Element	**Description**
theItemCollection As ItemCollection	Returns the collection of RoseItems currently selected in the model. Please note that the only items returned are those that inherit from RoseItem. For example, External Documents do not inherit from RoseItem and, therefore, are not returned by this method.
myModel As Model	Instance of the model from which items are being retrieved.

Model.GetSelectedModules Method

Description

This method returns all modules selected in the current model.

Syntax

Set *theModules* = *theObject*.**GetSelectedModules** ()

Element	Description
theModules As ModuleCollection	Contains the collection of modules currently selected in the model
theObject As Model	Instance of the model from which modules are being retrieved

Model.GetSelectedSubsystems Method

Description

This method returns all subsystems selected in the current model.

Syntax

Set *theSubsystems* = *theObject*.**GetSelectedSubsystems** ()

Element	Description
theSubsystems As SubsystemCollection	Returns the collection of subsystems currently selected in the model
theObject As Model	Instance of the model from which subsystems are being retrieved

Model.GetSelectedUseCases Method

Description

This method returns all use cases selected in the current model.

Syntax

Set *theUseCases* = *theObject*.**GetSelectedUseCases** ()

Element	Description
theUseCases As UseCaseCollection	Returns the collection of use cases currently selected in the model
theObject As Model	Instance of the model from which use cases are being retrieved

Model.Import Method

Description

This method imports packages (REI Category class) and subsystems into the current model.

Syntax

blnImported = *objModel*.**Import** (*strName*)

Element	Description
blnImported As Boolean	Returns TRUE if the package or subsystem is successfully imported into the model
objModel As Model	Model into which packages or subsystems are being imported
strName As String	Name of the package or subsystem to be imported

Model.LoadControlledUnits Method

Description

This method loads specified unloaded controlled units into a model.

Syntax

blnLoaded = *objModel*.**LoadControlledUnits** (*colUnits*)

Element	Description
blnLoaded As Boolean	Returns TRUE if the controlled units are successfully loaded into the model
objModel As Model	Model into which controlled units are being loaded
colUnits As ControllableUnitCollection	Collection of controlled units to be loaded

Model.ResolveReferences Method

Description

This method fixes unresolved references in the current model provided all the necessary model elements are loaded in the model. This method iterates through all the items in the model and resolves any previously unresolved associations and relations.

Syntax

theModel.**ResolveReferences** ()

Element	Description
theModel As Model	Instance of the model for which references are being resolved

Module Class

A module is a unit of code that serves as a building block for the physical structure of a system. The module class exposes properties and methods that allow you to define and manipulate the characteristics of modules.

Module Class Properties

The following table summarizes the Module Class properties.

Table 92 Module Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
AssignedLanguage	Specifies the programming language assigned to the module
Declarations	Collection that contains the declarations belonging to the module
OtherPart	Defines the module as another part of a subsystem
ParentSubSystem	Subsystem that contains the module
Part	Defines the module as part of a subsystem
Path	Defines the path in which the module resides
Type	Defines the module type

Module.Declarations Property

Description

Specifies the declarations collection belonging to the module.

Syntax

`Module``.`**`Declarations`**

Property Type

String

Module.OtherPart Property

Description

Specifies the other part of a module kind that has two parts. May be set to *Nothing* if the other part is not defined.

Note: *This property is read-only.*

Syntax

`Module``.`**`OtherPart`**

Property Type

Module

Module.ParentSubSystem Property

Description

Identifies the subsystem object that contains the module; is always set to a valid object (is never set to *Nothing).*

Note: *This property is read-only.*

Syntax

`Module`.**`ParentSubsystem`**

Property Type

Subsystem

Module.Part Property

Description

The Part property is a rich data type. The following table describes the valid forms of expressing the Module Part rich data type.

Table 93 Module.Part Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Specification ■ Body ■ Generic ■ Main

Note: *This property is read-only.*

Syntax

`Module``.`**`Part`**

Property Type

ModulePart

Module.Path Property

Description

If set during code generation, this property identifies the file path of the module. You can set this value to an appropriate directory path of your choice.

Note: *If the add-in performing code generation does not set this property,* ***Path*** *will not identify the file path of the module.*

Syntax

thePath = *theModule*.**Path**

theModule.**Path** = **“C:\MyModules”**

Property Type

String

Module.AssignedLanguage Property

Description

Specifies the programming language assigned to the module.

Syntax

Module.**AssignedLanguage**

Property Type

String

Module.Type Property

Description

The Type property is a rich data type. The following table describes the valid forms of expressing the Module Type rich data type.

Table 94 Module.Type Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ SubType ■ PackageType ■ TaskType

Note: *This property is read-only.*

Syntax

`Module``.`**`Type`**

Property Type

ModuleType

Module Class Methods

The following table summarizes the Module Class methods.

Table 95 Module Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddRealizeRel	Adds a realize relation to a module
AddVisibilityRelationship	Creates a new visibility relationship and adds it to a module
DeleteRealizeRel	Deletes a realize relation from a module
DeleteVisibilityRelationship	Deletes a visibility relationship from a module
GetAllDependencies	Retrieves all modules that affect the module
GetAssignedClasses	Retrieves the collection of classes assigned to the module
GetDependencies	Retrieves the collection of modules included in the module
GetRealizeRelations	Retrieves the collection of realize relations belonging to a module
GetSubsystemDependencies	Retrieves the collection of visibility relationships between a module and a subsystem

Module.AddRealizeRel Method

Description

This method adds a realize relation to a module.

Syntax

```
Set theRealizeRelation = theModule.AddRealizeRel
      (theRelationName, theInterfaceName)
```

Element	Description
theRealizeRelation As RealizeRelation	Returns the collection of visibility relationships between the module and the specified subsystem
theModule As Module	Module to which the Realize relation is being added
theRelationName As String	Name of the relation being added
theInterfaceName As String	Name of the interface being added

Module.AddVisibilityRelationship Method

Description

This method creates a new module visibility relationship and adds it to a module.

Syntax

```
Set theRelationship = theObject.AddVisibilityRelationship
       (theModule)
```

Element	Description
theRelationship As ModuleVisibilityRelationship	Creates a new module visibility relationship and returns it in the specified object
theObject As Module	Instance of the module to which the relationship is being added
theModule As Module	Instance of the module being added as a visibility relationship

Module.DeleteRealizeRel Method

Description

This method deletes a realize relation from a module.

Syntax

```
IsDeleted = theModule.DeleteRealizeRel (theRealizeRel)
```

Element	Description
IsDeleted As Boolean	Returns a value of True when the relation is deleted
theModule As Module	Module from which the Realize relation is being deleted
theRealizeRel As RealizeRelation	Realize relation being deleted

Module.DeleteVisibilityRelationship Method

Description

This method removes a module visibility relationship from a module.

Syntax

```
Deleted = theObject.DeleteVisibilityRelationship
      (theRelationship)
```

Element	Description
Deleted As Boolean	Returns a value of True when the relationship is deleted
theObject As Module	Instance of the module from which the relationship is being removed
theRelationship As ModuleVisibilityRelationship	Instance of the module visibility relationship being deleted

Module.GetAllDependencies Method

Description

This method retrieves the collection of module visibility relationships that affect a module.

Syntax

```
Set theDependencies = theObject.GetAllDependencies ()
```

Element	Description
theDependencies As ModuleVisibilityRelationshipCollection	Returns the collection of all modules belonging to the specified module, as well as all modules belonging to those modules
theObject As Module	Instance of the module from which the dependencies are being retrieved

Module.GetAssignedClasses Method

Description

This method retrieves the collection of classes assigned to a module

Syntax

`Set` *`theClasses`* `=` *`theObject`*`.`**`GetAssignedClasses`** `()`

Element	**Description**
theClasses As ClassCollection	Returns the collection of classes assigned to a module
theObject As Module	Instance of the module from which the classes are being retrieved

Module.GetDependencies Method

Description

This method retrieves the collection of module visibility relationships from a module

Syntax

`Set` *`theDependencies`* `=` *`theObject`*`.`**`GetDependencies`** `()`

Element	**Description**
theDependencies As ModuleVisibilityRelationshipCollection	Returns the collection of module visibility relationships belonging to a module
theObject As Module	Instance of the module from which the relationships are being retrieved

Module.GetRealizeRelations Method

Description

This method retrieves the collection of realize relationships belonging to a module

Syntax

```
Set theRelations = theModule.GetRealizeRelations ()
```

Element	Description
theRelations As RealizeRelationCollection	Returns the collection of realize relations belonging to a module
theModule As Module	Instance of the module from which the realize relationships are being retrieved

Module.GetSubsystemDependencies Method

Description

This method retrieves the collection of visibility relationships between a module and a subsystem.

Syntax

```
Set theDependencies = theObject.GetSubsystemDependencies
        (theSubsystem)
```

Element	Description
theDependencies As ModuleVisibility RelationshipCollection	Returns the collection of visibility relationships between the module and the specified subsystem
theObject As Module	Instance of the module whose relationships are being checked
theSubsystem As Subsystem	Instance of the subsystem whose relationships are being checked

ModuleDiagram Class

A module diagram maps the allocation classes and objects to modules. The module diagram class exposes properties and methods that allow you to add, retrieve and delete classes and objects in a module diagram.

ModuleDiagram Class Properties

The following table summarizes the ModuleDiagram Class properties.

Table 96 ModuleDiagram Class Properties Summary

Property	Description
Element Properties	Inherits all Element Class properties
Diagram Properties	Inherits all Diagram Class properties
ComponentViews	Specifies the collection of component views on a component diagram
ParentSubsystem	Subsystem that contains the module diagram
SubsystemViews	Specifies the collection of subsystem views on a component diagram

ModuleDiagram.ComponentViews Property

Description

This property identifies all the component views on a component diagram.

Note: *This property is read-only.*

Syntax

```
Set colComponentViews =
      objComponentDiagram.ComponentViews
```

Property Type

ComponentViewCollection

ModuleDiagram.ParentSubsystem Property

Description

Identifies the subsystem object that contains the module.

Is always set to a valid object (is never set to *Nothing*).

Note: *This property is read-only.*

Syntax

```
ModuleDiagram.ParentSubsystem
```

Property Type

Subsystem

ModuleDiagram.SubsystemViews Property

Description

This property identifies all the subsystem views on a component diagram.

Note: *This property is read-only.*

Syntax

```
Set colSubsystemViews =
      objComponentDiagram.SubsystemViews
```

Property Type

SubsystemViewCollection

ModuleDiagram Class Methods

The following table summarizes the MethodDiagram Class methods.

Table 97 ModuleDiagram Class Methods Summary

Property	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element Class methods
Diagram Methods	Inherits all Diagram Class methods
AddComponentView	Adds a component view to a component diagram
AddSubsystemView	Adds a subsystem view to a component diagram
GetModules Method	Retrieves the collection that contains all modules belonging to the module diagram
GetSelectedModules Method	Retrieves the modules currently selected in the module diagram
GetSelectedSubsystems Method	Retrieves the subsystems currently selected in the module diagram
GetSubsystems Method	Retrieves the collection that contains all subsystems belonging to the module diagram
RemoveComponentView	Removes a component view from a component diagram
RemoveSubsystemView	Removes a subsystem view from a component diagram

ModuleDiagram.AddComponentView Method

Note: *This method replaces AddModule.*

Description

This method adds a component view to a component diagram.

Syntax

```
Set objComponentView =
      objComponentDiagram.AddComponentView (objComponent)
```

Element	Description
objComponentView As ComponentView	Returns the component view object added to the component diagram
objComponentDiagram As ModuleDiagram	Component diagram to which the component view is being added
objComponent As Module	Component whose view is being added to the diagram

See also

ModuleDiagram.RemoveComponentView Method

ModuleDiagram.AddSubsystemView Method

Note: *This method replaces AddSubsystem.*

Description

This method adds a subsystem view to a component diagram.

Syntax

```
Set objSubsystemView =
      objComponentDiagram.AddSubsystemView (objSubsystem)
```

Element	**Description**
objSubsystemView As SubsystemView	Returns the subsystem view object added to the component diagram
objComponentDiagram As ModuleDiagram	Component diagram to which the subsystem view is being added
objSubsystem As Subsystem	Subsystem whose view is being added to the diagram

See also

ModuleDiagram.RemoveSubsystemView Method

ModuleDiagram.GetModules Method

Description

This method retrieves the collection of modules belonging to the module diagram.

Syntax

`Set theModules = theDiagram.GetModules ()`

Element	**Description**
theModules As ModuleCollection	Returns the collection of modules belonging to the module diagram
theDiagram As ModuleDiagram	Module diagram from which to retrieve the modules

ModuleDiagram.GetSelectedModules Method

Description

This method retrieves the collection of currently selected modules from a module diagram.

Syntax

`Set theModules = theDiagram.GetSelectedModules ()`

Element	**Description**
theModules As ModuleCollection	Returns the collection of currently selected modules from the module diagram
theDiagram As ModuleDiagram	Module diagram from which to retrieve the modules

ModuleDiagram.GetSelectedSubsystems Method

Description

This method retrieves the collection of currently selected subsystems from a module diagram.

Syntax

Set *theSubsystems* = *theDiagram*.**GetSelectedSubsystems** ()

Element	Description
theSubsystems As SubsystemCollection	Returns the collection of currently selected subsystems from the module diagram
theDiagram As ModuleDiagram	Module diagram from which to retrieve the modules

ModuleDiagram.GetSubsystems Method

Description

This method retrieves the collection of subsystems from the module diagram.

Syntax

Set *theSubsystems* = *theDiagram*.**GetSubsystems** ()

Element	Description
theSubsystems As SubsystemCollection	Returns the collection of subsystems belonging to the module diagram
theDiagram As ModuleDiagram	Module diagram from which to retrieve the subsystems

ModuleDiagram.RemoveComponentView Method

Description

This method removes a component view from a component diagram.

Syntax

```
blnIsRemoved = objComponentDiagram.RemoveComponentView
      (objComponentView)
```

Element	Description
blnIsRemoved As Boolean	Returns a value of TRUE if the component view object is successfully removed from the component diagram
objComponentDiagram As ModuleDiagram	Component diagram from which the component view is being removed
objComponentView As ComponentView	Component view being removed from the diagram

See also

ModuleDiagram.AddComponentView Method

ModuleDiagram.RemoveSubsystemView Method

Description

This method removes a subsystem view from a component diagram.

Syntax

blnIsRemoved = *objComponentDiagram*.**RemoveSubsystemView** (*objSubsystemView*)

Element	Description
blnIsRemoved As Boolean	Returns a value of TRUE if the subsystem view object is successfully removed from the component diagram
objComponentDiagram As ModuleDiagram	Component diagram from which the subsystem view is being removed
objSubsystemView As SubsystemView	Subsystem view being removed from the diagram

See also

ModuleDiagram.AddSubsystemView Method

ModuleVisibilityRelationship Class

The ModuleVisibilityRelationship class describes the context and supplier relationship between modules. It inherits all of the RoseItem properties and methods.

ModuleVisibilityRelationship Class Properties

The following table summarizes the ModuleVisibilityRelationship Class properties.

Table 98 ModuleVisibilityRelationship Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
ContextModule	Name of the relation's context module
ContextSubsystem	Name of the relation's context subsystem
SupplierModule	Name of the relation's supplier module
SupplierSubsystem	Name of the relation's supplier subsystem

ModuleVisibilityRelationship.ContextModule Property

Description

Indicates the context module belonging to the ModuleVisibilityRelationship.

Note: *This property is read-only.*

Syntax

ModuleVisibilityRelationship.**ContextModule**

Property Type

Module

ModuleVisibilityRelationship.ContextSubsystem Property

Description

Indicates the context subsystem belonging to the ModuleVisibilityRelationship.

Note: *This property is read-only.*

Syntax

`ModuleVisibilityRelationship`.**`ContextSubsystem`**

Property Type

Subsystem

ModuleVisibilityRelationship.SupplierModule Property

Description

Indicates the supplier module belonging to the Module Visibility Relationship.

Note: *This property is read-only.*

Syntax

`ModuleVisibilityRelationship`.**`SupplierModule`**

Property Type

Module

ModuleVisibilityRelationship.SupplierSubsystem Property

Description

Indicates the supplier subsystem belonging to the Module Visibility Relationship.

Note: *This property is read-only.*

Syntax

`ModuleVisibilityRelationship`.**`SupplierSubsystem`**

Property Type

Subsystem

ModuleVisibilityRelationship Class Methods

The following table summarizes the ModuleVisibilityRelationship Class methods.

Table 99 ModuleVisibilityRelationship Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods

NotationTypes Enumeration

NotationTypes is an enumeration that defines values corresponding to the types of notation that can be used to determine how Rational Rose displays diagrams. NotationTypes can be used in the Notation Model property.

The following table describes the valid values for the NotationTypes enumeration.

Table 100 NotationTypes Enumeration Valid Values

Value	Integer Value	Description
BoochNotation	0	Use to set notation to Booch in Rational Rose For example, use **BoochNotation** in the Notation Model property to display Rational Rose diagrams and objects using Booch icons and terminology.
OMTNotation	1	Use to set notation to OMT in Rational Rose For example, use **OMTNotation** in the Notation Model property to display Rational Rose diagrams and objects using OMT icons and terminology.
UMLNotation	2	Use to set notation to UML in Rational Rose For example, use **UMLNotation** in the Notation Model property to display Rational Rose diagrams and objects using UML icons and terminology.

Note: *Rational Rose Automation users may use the value (e.g., **BoochNotation**) in their methods. Rational Rose Script users must use the integer value (e.g., 0) in their methods.*

NoteView Class

The note view class inherits the RoseItem properties and methods that determine the size and placement of the note view on a diagram.

NoteView Class Properties

The following table summarizes the NoteView Class properties.

Table 101 NoteView Class Properties Summary

Property	**Description**
Element	Inherits all Element Class properties
RoseItemView	Inherits all RoseItemView Class properties
Text	Contains the text that appears in the note view

NoteView.Text Property

Description

Contains the text that appears in the NoteView object.

Syntax

`NoteView``.`**`Text`**

Property Type

String

NoteView Class Methods

The following table summarizes the NoteView Class methods.

Table 102 NoteView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
RoseItemView Methods	Inherits all RoseItemView methods
AddAttachmentToView	Anchors a NoteView to a RoseItemView
GetDiagramLink	Returns the diagram linked to the NoteView's diagram, via the NoteView
GetNoteViewType	Returns the Type value of a NoteView object, which indicates whether the note is free-floating or appears in a box

NoteView. AddAttachmentToView Method

Description

This method anchors the specified NoteView to the specified RoseItemView, and returns the anchor object. This method also allows you to specify text to display on your anchor.

Syntax

```
Set objNoteAnchor = objNoteView.AddAttachmentToView
       (objRoseItemView, strLabelText)
```

Element	Description
objNoteAnchor As RoseItemView	Newly created note anchor
objNoteView As NoteView	NoteView being anchored to a RoseItemView
objRoseItemView As RoseItemView	RoseItemView to which a NoteView is being anchored
strLabelText As String	Text label to display on the newly created anchor

NoteView. GetDiagramLink Method

Description

This method returns the diagram linked to the specified NoteView's diagram, via the NoteView. If the NoteView is not linked to a diagram, this method returns **Nothing**. To retrieve the diagram on which the NoteView appears, use the RoseItemView.ParentDiagram property.

Syntax

```
Set objDiagram = objNoteView.GetDiagramLink ( )
```

Element	Description
objDiagram As Diagram	Returns the linked diagram
objNoteView As NoteView	NoteView whose linked diagram is being retrieved

NoteView.GetNoteViewType Method

Description

Returns the Type value of a NoteView object.

Syntax

```
theType = theNoteView.GetNoteViewType ()
```

Element	Description
theType As Integer	Retrieves the integer value that corresponds to the NoteView type 1 = Free floating text label 2 = Note with box
theNoteView As NoteView	Instance of the NoteView whose type is being retrieved

ObjectFlow Class Overview

The ObjectFlow class is an abstract class that exposes Rose's object flow functionality in the extensibility interface. With the properties and methods of the ObjectFlow class, you can:

- Retrieve information about object flows such as name, documentation, and stereotype
- Retrieve information about objects associated with object flows such as supplier name
- Retrieve objects associated with object flows such as external documents, state machine owner, Rose application, model, supplier, and client
- Determine if the object flow has a client and supplier
- Open specification sheets for object flows
- Add and delete external documents
- Create and retrieve tool and property settings for object flows

The ObjectFlow class corresponds to object flows in the Rose user interface.

ObjectFlow Class Properties

The following table summarizes the ObjectFlow class properties.

Table 103 ObjectFlow Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Properties	Inherits all Relation class properties

ObjectFlow Class Methods

The following table summarizes the ObjectFlow class methods.

Table 104 ObjectFlow Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
Relation Methods	Inherits all Relation class methods

ObjectInstance Class

The object class exposes properties and methods that:

- Determine the characteristics of objects in a model (for example, the class associated with the object and whether multiple instances of the object exist)
- Allow you to retrieve objects from a model

ObjectInstance Class Properties

The following table summarizes the ObjectInstance Class properties.

Table 105 ObjectInstance Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
ClassName	Contains the class name associated with the object
Links	Contains the collection of links belonging to the object instance
MultipleInstances	Indicates whether there are multiple instances of the object
Persistence	Indicates whether the object instance is persistent or static

ObjectInstance.ClassName Property

Description

Specifies the class to which the object instance belongs.

Syntax

ObjectInstance.**ClassName**

Property Type

String

ObjectInstance.MultipleInstances Property

Description

Indicates whether there are multiple instances of the object.

Syntax

ObjectInstance.**MultipleInstances**

Property Type

Boolean

ObjectInstance.Links Property

Description

Contains the collection of links belonging to the object instance.

Note: *This property is read-only.*

Syntax

ObjectInstance.**Links**

Property Type

LinkCollection

ObjectInstance.Persistence Property

Description

This property indicates whether the object instance is persistent or static.

Syntax

ObjectInstance.**Persistence**

Property Type

Integer

Values

0 = Persistent

1 = Static

ObjectInstance Class Methods

The following table summarizes the ObjectInstance Class methods.

Table 106 ObjectInstance Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddLink	Adds a link to the object instance
DeleteLink	Deletes a link from the object instance
GetClass	Retrieves the class to which the object instance belongs
IsClass	Returns a value of True if the object instance is a class

ObjectInstance.AddLink Method

Description

This method adds a link to the object instance.

Syntax

```
Set theLink = theObjectInstance.AddLink (theName,
      ToInstance)
```

Element	**Description**
theLink As Link	Returns the link being added to the object instance ***Note:*** *If you try to add a second Link object between the same instances and the Scenario diagram is a Sequence diagram, an error occurs and a Nothing object is returned.*
theObjectInstance As ObjectInstance	Object instance to which the link is being added
theName As String	Name of the message belonging to the link
ToInstance As ObjectInstance	Object Instance to receive the message belonging to the link

ObjectInstance.DeleteLink Method

Description

This method deletes a link from the object instance.

Syntax

IsDeleted = *theObjectInstance*.**DeleteLink** (*theLink*)

Element	Description
IsDeleted As Boolean	Returns a value of True when the link is successfully deleted
theObjectInstance As ObjectInstance	Object instance from which the link is being deleted
theLink As Link	Link being deleted from the object instance

ObjectInstance.GetClass Method

Description

This method retrieves the class to which the object instance belongs.

Syntax

Set *theClass* = *theObjectInstance*.**GetClass** ()

Element	Description
theClass As Class	Returns the class to which the object instance belongs
theObjectInstance As ObjectInstance	Object instance whose class is being retrieved

ObjectInstance.IsClass Method

Description

This method indicates whether the object instance is a class.

Syntax

IsClass = *theObject*.**IsClass** ()

Element	Description
IsClass As Boolean	Returns a value of True if the object instance is a class
theObject As ObjectInstance	Object instance being checked

Operation Class

Objects in a class carry out their defined responsibilities by using operations. Each operation performs a single, cohesive function. The operation class exposes properties and methods that:

- Determine operation characteristics
- Add or remove parameters from operations
- Allow you to retrieve operations

Operation Class Properties

The following table summarizes the Operation Class properties.

Table 107 Operation Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Concurrency	Controls operation concurrency
Exceptions	Identifies the set of exceptions that can be raised by an operation
ExportControl Property	Controls attribute visibility
Parameters	Contains the collection of parameters belonging to the operation
ParentClass	Specifies the class to which the operation belongs
Postconditions	Specifies the invariants that are satisfied by the operation (exit behavior of the operation)
Preconditions	Specifies the invariants assumed by the operation (entry behavior of an operation)
Protocol	Specifies the set of operations that a client may perform on an object
Qualification	Identifies language-specific features used to qualify an operation
ReturnType	Specifies the return type for the operation
Semantics	Specifies the action of an operation
Size	Identifies the relative or absolute amount of storage used when the operation is called
Time	Identifies the relative or absolute amount of time required to complete the operation
Virtual	Specifies this as a virtual operation

Operation.Concurrency Property

Description

The Operation Concurrency property is a rich data type. The following table describes the valid forms of expressing the Operation Concurrency rich data type for the Operation class.

Table 108 Operation.Concurrency Rich Data Types

Rich Data Type	**Description**
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Sequential ■ Guarded ■ Synchronous

Note: *This property is read-only.*

Syntax

```
Operation.Concurrency
```

Property Type

Concurrency

Operation.Exceptions Property

Description

Identifies the set of exceptions that can be raised by an operation.

Syntax

```
Operation.Exceptions
```

Property Type

String

Operation.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the Operation object. The following table describes the valid forms of expressing the ExportControl rich data type for the Operation class.

Table 109 Operation.ExportControl Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible string values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControl property is read-only. Its Value and Name, however, are read/write.*

Syntax

```
theOperation.ExportControl.Name = "PrivateAccess"

theOperation.ExportControl.Value = 2

theNameStr = theOperation.ExportControl.Name

theValue = theOperation.ExportControl.Value
```

Property Type

RichType or OperationExportControl

Operation.Parameters Property

Description

Defines the collection of parameters that is valid for the operation.

Note: *This property is read-only.*

Syntax

`Operation`.**`Parameters`**

Property Type

ParameterCollection

Operation.ParentClass Property

Description

Specifies the class to which the operation belongs.

Note: *This property is read-only.*

Syntax

`Operation`.**`ParentClass`**

Property Type

Class

Operation.Postconditions Property

Description

Controls invariants that are satisfied by the operation; that is, the exit behavior of the operation.

Syntax

Operation.**Postconditions**

Property Type

String

Operation.Preconditions Property

Description

Controls invariants assumed by the operation; that is, the entry behavior of an operation.

Syntax

Operation.**Preconditions**

Property Type

String

Operation.Protocol Property

Description

Specifies the set of operations that a client may perform on an object and the legal order in which the operations can be called.

Syntax

Operation.**Protocol**

Property Type

String

Operation.Qualification Property

Description

Identifies language-specific features used to qualify an operation.

Syntax

`Operation`.**`Qualification`**

Property Type

String

Operation.ReturnType Property

Description

Determines the object type to be returned by an operation; can be set to any valid data type, rich data type or object type.

Syntax

`Operation`.**`ReturnType`**

Property Type

String

Operation.Semantics Property

Description

Controls the action of an operation.

Syntax

`Operation`.**`Semantics`**

Property Type

String

Operation.Size Property

Description

Identifies the relative or absolute amount of storage used when the operation is called.

Syntax

`Operation``.`**`Size`**

Property Type

String

Operation.Time Property

Description

Identifies the relative or absolute amount of time required to complete the operation.

Syntax

`Operation``.`**`Time`**

Property Type

String

Operation.Virtual Property

Description

Indicates whether the operation is virtual.

Syntax

`Operation``.`**`Virtual`**

Property Type

Boolean

Operation Class Methods

The following table summarizes the Operation Class methods.

Table 110 Operation Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddParameter	Adds a parameter to an operation
DeleteParameter	Deletes a parameter from an operation
RemoveAllParameters	Removes all parameters from the operation

Operation.AddParameter Method

Description

This method creates a new parameter and adds it to an operation.

Syntax

```
Set theParameter = theObject.AddParameter (ParameterName,
      ParameterType, InitValue, Position)
```

Element	**Description**
theParameter As Parameter	Returns the parameter being added to the operation
theObject As Operation	Operation to which the parameter is being added
ParameterName As String	Name of the parameter being added to the operation
ParameterType As String	Type of parameter being added to the operation
InitValue As String	Initial value of the added parameter
Position As Integer	Order of the parameter in the operation’s parameter list

Operation.DeleteParameter Method

Description

This method deletes a parameter from an operation.

Syntax

`Deleted` = *`theObject`*.**`DeleteParameter`** (*`theParameter`*)

Element	Description
Deleted As Boolean	Returns a value of True when the specified parameter is deleted from the operation
theObject As Operation	Operation from which the parameter is being deleted
theParameter As Parameter	Parameter being deleted from the operation

Operation.RemoveAllParameters Method

Description

This subroutine removes all parameters from an operation.

Syntax

`theObject`.**`RemoveAllParameters`**

Element	Description
theObject As Operation	Operation from which the parameters are being removed

Package Class

The Package Class is a container for the model elements that correspond to the UML Package concept.

Package class methods allow you to determine whether a package is the root package in a model, as well as to obtain the OLE object associated with the package.

Package Class Properties

The following table summarizes the Package Class properties.

Table 111 Package Class Properties Summary

Property	**Description**
Element Class Properties	Inherits all Element class properties
RoseItem Class Properties	Inherits all RoseItem class properties

Package Class Methods

The following table summarizes the Package Class methods.

Table 112 Package Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
ControllableUnit Methods	Inherits all ControllableUnit class methods
IsRootPackage	Indicates whether the package is the root package (category) of the model

Package.IsRootPackage Method

Description

This method finds out if the specified package is the root package (category) of the model.

Syntax

IsRoot = *thePackage*.**IsRootPackage** ()

Element	Description
IsRoot As Boolean	Returns a value of True if the package is the root package (category) of the model
thePackage As Package	Package being checked as root package

Parameter Class

Parameters further qualify the behavior of an operation. The parameter class exposes properties and methods that:

- Determine the parameter characteristics such as type and initial value
- Allow you to retrieve parameters

Parameter Class Properties

The following table summarizes the Parameter Class properties.

Table 113 Parameter Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Const	Indicates whether the parameter is a constant
InitValue	Sets the initial value of the parameter
Type	Indicates the data type of the parameter

Parameter.Const Property

Description

Indicates that the parameter is a constant.

Syntax

Object.**Const**

Property Type

Boolean

Parameter.InitValue Property

Description

Indicates the initial value of the parameter object.

Syntax

Object.**InitValue**

Property Type

String

Parameter.Type Property

Description

Indicates the data type of the parameter object.

Syntax

Object.**Type**

Property Type

String

Parameter Class Methods

The following table summarizes the Parameter Class methods.

Table 114 Parameter Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods

PathMap Class

Use the PathMap class to create and edit path map entries for the current model. For example, you can create entries to define paths to controlled units, to scripts executed from the Rational Rose menu, and to the root directory for a multi-user project.

Executing PathMap class methods is equivalent to updating the PathMap dialog in the Rational Rose user interface.

PathMap Class Properties

There are no PathMap class properties.

PathMap Class Methods

The following table summarizes the PathMap Class methods.

Table 115 PathMap Class Methods Summary

Method	**Description**
AddEntry	Adds an entry to the current model's path map definition
DeleteEntry	Deletes an entry from the current model's path map definition
GetActualPath	Retrieves the actual path associated with a given virtual path symbol
GetActualPathWithContext	Retrieves the actual path within context associated with a specified path map object
Get Virtual Path	Retrieves the virtual path associated with a given actual path
GetVirtualPathWithContext	Retrieves the virtual path symbol, within context, that corresponds to the given actual path
HasEntry	Determines whether a given string is defined in the current model's path map

PathMap.AddEntry Method

Description

This method adds an entry to the current application's PathMap definition.

Syntax

```
IsAdded = theObject.AddEntry (theSymbol, theActualPath,
      theComment)
```

Element	Description
IsAdded As Boolean	Returns a value of true when the entry is successfully added
theObject As PathMap	PathMap to which the entry is being added
theSymbol As String	Virtual symbol being added to the PathMap. For example, $SCRIPT_PATH
theActualPath As String	Actual path to which the virtual symbol refers
theComment As String	Description of the PathMap entry being added

PathMap.DeleteEntry Method

Description

This method deletes an entry from the current application's PathMap definition.

Syntax

```
IsDeleted = theObject.DeleteEntry (theSymbol)
```

Element	Description
IsDeleted As Boolean	Returns a value of true when the entry is successfully deleted
theObject As PathMap	PathMap from which the entry is being deleted
theSymbol As String	Virtual symbol for the entry being deleted from the PathMap. For example, $SCRIPT_PATH

PathMap.GetActualPath Method

Description

This method retrieves from the PathMap the actual path that corresponds to the given virtual symbol.

Syntax

`theActualPath = theObject.GetActualPath (theSymbol)`

Element	Description
theActualPath As String	Returns the actual path given the virtual symbol
theObject As PathMap	PathMap from which to retrieve the actual path
theSymbol As String	Virtual symbol whose corresponding actual path is being retrieved

PathMap.GetActualPathWithContext Method

Overview

Along with GetVirtualPathWithContext, this method is useful to Add-Ins that want to store files (scripts, controlled units) relative to their installed directory. For example, you could set up virtual paths to be:

anAddInScripts = “&\Script”

anAddInControlledUnits = “&\CntrlUnits”

Then, you can use GetActualPathWithContext to get the entire actual path.

Description

This method retrieves the actual path within context associated with a specified PathMap object.

Note: *Context is the directory path that is substituted for the ampersand (&) if one is encountered in the actual path.*

Example

If `$MyPath = &\Programs` and

`Set thePath = thePathMap.GetActualPathWithContext ("$MyPath", "D:\MyInst")` then

thePath = "D:\MyInst\Programs"

Syntax

```
Set theActualPathWithContext =
      theObject.GetActualPathWithContext (theVirtualPath,
      theContext)
```

Element	Description
theActualPathWithContext As String	Returns the actual path given the virtual path and context
TheObject As PathMap	PathMap from which to retrieve the actual path
theVirtualPath As String	Virtual path symbol whose corresponding actual path is being retrieved
theContext As String	The directory path to be substituted into the virtual path

PathMap.GetObject Method

Description

This method retrieves the OLE object associated with a specified PathMap object.

Note: *This method is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

```
Set thePathMapObject = theObject.GetObject ()
```

Element	Description
thePathMapObject As PathMap	Returns the OLE automation interface object associated with the PathMap object
theObject As PathMap	Path Map whose OLE object is being retrieved

PathMap.GetVirtualPath Method

Description

This method retrieves the virtual path that corresponds to the given actual path.

Syntax

```
theVirtualPath = theObject.GetVirtualPath (theActualPath)
```

Element	Description
theVirtualPath As String	Returns the virtual path given the actual path
theObject As PathMap	PathMap from which to retrieve the virtual path
theActualPath As String	Actual path whose corresponding virtual path is being retrieved

PathMap.GetVirtualPathWithContext Method

Overview

Along with GetActualPathWithContext, this method is useful to Add-Ins that want to store files (e.g., scripts, controlled units) relative to their installed directory. For example, you could set up virtual paths to be:

anAddInScripts = "&\Script"

anAddInControlledUnits = "&\CntrlUnits"

Then, you can use GetVirtualPathWithContext to get the virtual path from an actual path and context.

Description

This method retrieves the virtual path, within context, that corresponds to the given actual path.

Note: *Context is the directory path that is substituted for the ampersand (&) if one is encountered in the actual path.*

Example

If `$virtual_path = "&\Programs"` in the Rational Rose Path Map and

```
Set theVirtualPathSymbol =
myPathMap.GetVirtualPathWithContext
("D:\MyInst\Programs", "D:\MyInst")
```
then

theVirtualPathSymbol = "$virtual_path"

Syntax

```
Set theVirtualPathWithContext =
      theObject.GetVirtualPathWithContext (theActualPath,
      theContext)
```

Element	Description
theVirtualPathWithContext As String	Returns the virtual path symbol given the actual path and context
TheObject As PathMap	PathMap from which to retrieve the virtual path symbol
theActualPath As String	Actual path whose corresponding virtual path symbol is being retrieved
theContext As String	The directory path to be removed from the actual path

PathMap.HasEntry Method

Description

This method checks the PathMap for an entry based on the given virtual path symbol.

Syntax

```
HasEntry = theObject.HasEntry (theSymbol)
```

Element	Description
HasEntry As Boolean	Returns a value of True if the PathMap has an entry for the given virtual path symbol
theObject As PathMap	PathMap being checked
theSymbol As String	Virtual symbol to search for in the PathMap

Process Class

A process is the execution of one thread of control in an object-oriented program or system. The process class exposes properties and methods that allow you to define and manipulate the characteristics of processes.

Process Class Properties

The following table summarizes the Process Class properties.

Table 116 Process Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element properties
RoseItem Attributes	Inherits all RoseItem properties
MyProcessor	Defines the processor that belongs to the process
Priority	Sets the priority of the process

Process.MyProcessor Property

Description

Identifies the processor as belonging to a process.

Note: *This property is read-only.*

Syntax

`Process``.`**`MyProcessor`**

Property Type

Processor

Process.Priority Property

Description

Indicates the relative priority of a process; otherwise the process is static.

Syntax

Process.**Priority**

Property Type

String

Process Class Methods

The following table summarizes the Process Class methods.

Table 117 Process Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods

Processor Class

A processor is hardware that is capable of executing programs. Processors are assigned to implement processes. The processor class exposes properties and methods that allow you to define and manipulate the characteristics of processors in a model.

For example, you can add or remove processes or define the scheduling type for a processor.

Processor Class Properties

The following table summarizes the Processor Class properties.

Table 118 Processor Class Properties Summary

Property	**Description**
Element	Inherits all Element properties
RoseItem properties	Inherits all RoseItem properties
Characteristics	Defines the characteristics belonging to the processor
Connections	Specifies the collection of connections for the processor
Processes	The collection of processes belonging to the processor
Scheduling	Indicates the schedule type for a class

Processor.Characteristics Property

Description

Defines the characteristics of the processor.

Syntax

Processor.**Characteristics**

Property Type

String

Processor. Connections Property

Description

This property specifies the collection of connections for the processor.

Note: *This property is read-only.*

Syntax

```
Set objConnectionRelationCollection = objProcessor.Connections
```

Property Type

ConnectionRelationCollection

Processor.Processes Property

Description

Defines the collection of processes that belong to the processor.

Note: *This property is read-only.*

Syntax

```
Processor.Processes
```

Property Type

ProcessCollection

Processor.Scheduling Property

Description

The Scheduling property is a rich data type. The following table describes the valid forms of expressing the Scheduling rich data type.

Table 119 Processor.Scheduling Rich Data Type

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Preemptive ■ NonPreemptive ■ Cyclic ■ Executive ■ Manual

Note: *This property is read-only.*

Syntax

```
Processor.Scheduling
```

Property Type

Schedule

Processor Class Methods

The following table summarizes the Processor Class methods.

Table 120 Processor Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddDeviceConnection	Adds a device connection to the processor
AddProcessorConnection Method	Adds a processor connection to the processor
AddProcess	Adds a process to the processor
DeleteProcess	Deletes a process from the processor
GetConnectedDevices	Retrieves connected devices from a processor
GetConnectedProcessors	Retrieves connected processors from a processor
RemoveDeviceConnection	Removes a device connection from the processor
RemoveProcessorConnection	Removes a processor connection from the processor

Processor.AddDeviceConnection Method

Description

Creates a new device connection and adds it to the processor.

Syntax

Connected = *theObject*.**AddDeviceConnection** (*theDevice*)

Element	Description
Connected As Boolean	Returns a value of True when the device is connected
theObject As Processor	Processor to which the connection is being added
theDevice As Device	Device connection being added

Processor.AddProcess Method

Description

Creates a new process and adds it to the processor.

Syntax

Set *theProcess* = *theObject*.**AddProcess** (*ProcessName*)

Element	Description
theProcess As Process	Returns the process being added to the processor
theObject As Processor	Processor to which the process is being added
ProcessName As String	Name of the process being added

Processor.AddProcessorConnection Method

Description

Creates a new processor connection and adds it to the processor.

Syntax

`Connected` = *`theObject`*.**`AddProcessorConnection`** (*`theProcessor`*)

Element	Description
Connected As Boolean	Returns a value of True when the processor is connected
theObject As Processor	Processor to which the connection is being added
theProcessor As Processor	Processor connection being added

Processor.DeleteProcess Method

Description

Deletes a process from a processor.

Syntax

`Deleted` = *`theObject`*.**`DeleteProcess`** (*`theProcess`*)

Element	Description
Deleted As Boolean	Returns a value of True when the process is deleted
theObject As Processor	Processor from which the process is being deleted
theProcess As Process	Process being deleted

Processor.GetConnectedDevices Method

Description

Retrieves the collection of devices that are connected to this processor.

Syntax

```
Set theDevices = theObject.GetConnectedDevices ()
```

Element	Description
theDevices As DeviceCollection	Returns the collection of devices that are connected to the specified processor
theObject As Processor	Processor whose connected devices are being retrieved

Processor.GetConnectedProcessors Method

Description

This method retrieves the collection of processors that are connected to this processor.

Syntax

```
Set theProcessors = theObject.GetConnectedProcessors ()
```

Element	Description
theProcessors As ProcessorCollection	Returns the collection of processors that are connected to the specified processor
theObject As Processor	Processor whose connected processors are being retrieved

Processor.RemoveDeviceConnection Method

Description

Removes a device connection from the processor.

Syntax

Removed = *theObject*.**RemoveDeviceConnection** (*theDevice*)

Element	Description
Removed As Boolean	Returns a value of True when the device connection is removed
theObject As Processor	Processor from which the connection is being removed
theDevice As Device	Device connection being removed

Processor.RemoveProcessorConnection Method

Description

Removes a processor connection from the processor.

Syntax

```
Removed = theObject.RemoveProcessorConnection
      (theProcessor)
```

Element	Description
Removed As Boolean	Returns a value of True when the processor connection is removed
theObject As Processor	Processor from which the connection is being removed
theProcessor As Processor	Processor connection being removed

Property Class

The Property class exposes properties and methods that:

- Determine the characteristics of properties in a model (for example, property name and type, as well as the development tool associated with the property)
- Allow you to retrieve properties from a model

Property Class Properties

The following table summarizes the Property Class properties.

Table 121 Property Class Properties Summary

Property	Description
Name	Name of the property
Value	Current value of the property
ToolName	Name of the software development tool with which the property is associated
Type	Indicates the type of information stored by the property

Property.Name Property

Description

Indicates the name of the property (without specifying a path).

Syntax

Property.**Name**

Property Type

String

Property.ToolName Property

Description

Indicates the name of the tool to be used for code generation for this model. For example, C++, PowerBuilder, Visual Basic, etc.

Note: *This property is read-only.*

Syntax

`Property`.**`ToolName`**

Property Type

String

Property.Type Property

Description

Indicates the type of information stored by the property.

Note: *This property is read-only.*

Syntax

`Property`.**`Type`**

Property Type

String

Values

String

Integer

Float

Char

Boolean

Enumeration

Note: *Other values may be valid if user-defined enumerated types exist.*

Property.Value Property

Description

Indicates the value of the property.

Syntax

Property.**Value**

Property Type

String

Property Class Methods

The following table summarizes the Property Class methods.

Table 122 Property Class Methods Summary

Method	Description
GetObject	Returns the OLE interface object associated with this object

Property.GetObject Method

Description

This method retrieves the OLE object associated with a specified Property object.

Note: *This method is only valid for Rational Rose Script; it does not exist in Rational Rose Automation.*

Syntax

Set *theOLEObject* = *theObject*.**GetObject** ()

Element	Description
theOLEObject As Object	Returns the OLE automation interface object associated with the specified object
theObject As Property	Object whose OLE object is being retrieved

RealizeRelation Class

A realize relationship between a logical class and a component class shows that the component class realizes the operations defined by the logical class.

RealizeRelation Class Properties

The following table summarizes the RealizeRelation Class properties.

Table 123   RealizeRelation Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Relation Class Properties	Inherits all Relation class properties

RealizeRelation Class Methods

The following table summarizes the RealizeRelation Class methods.

Table 124   RealizeRelation Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
Relation Methods	Inherits all Relation class methods
GetContextClass	Retrieves the Realize relation’s context (client) class
GetContextComponent	Retrieves the Realize relation’s context (client) component
GetSupplierClass	Retrieves the Realize relation’s supplier class
GetSupplierComponent	Retrieves the Realize relation’s supplier component

RealizeRelation.GetContextClass Method

Description

This method retrieves the Realize relation's context (client) class.

Syntax

```
Set theClass = theRealizeRelation.GetContextClass ()
```

Element	Description
theClass As Class	Returns the realize relation's context (client) class
theRealizeRelation As RealizeRelation	RealizeRelation whose context class is being retrieved

RealizeRelation.GetContextComponent Method

Description

This method retrieves the Realize relation's context (client) component (or module).

Syntax

```
Set theComponent = theRealizeRelation.GetContextComponent
        ()
```

Element	Description
theComponent As Component	Returns the realize relation's context (client) component (or module)
theRealizeRelation As RealizeRelation	RealizeRelation whose context component is being retrieved

RealizeRelation.GetSupplierClass Method

Description

This method retrieves the Realize relation’s supplier class.

Syntax

```
Set theClass = theRealizeRelation.GetSupplierClass ()
```

Element	Description
theClass As Class	Returns the realize relation’s supplier class
theRealizeRelation As RealizeRelation	RealizeRelation whose supplier class is being retrieved

RealizeRelation.GetSupplierComponent Method

Description

This method retrieves the Realize relation’s supplier component (or module).

Syntax

```
Set theComponent = theRealizeRelation.GetSupplierComponent
        ()
```

Element	Description
theComponent As Component	Returns the realize relation’s supplier component
theRealizeRelation As RealizeRelation	RealizeRelation whose supplier component is being retrieved

Relation Class

All relations (ClassRelation, Inherits, Has, Realizes) inherit from the Relation Class. Relation Class properties and methods allow you to retrieve the client and supplier information for the relations in a model.

Relation Class Properties

The following table summarizes the Relation Class properties.

Table 125 Relation Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
SupplierName	Specifies the name of the supplier belonging to the relation

Relation.SupplierName Property

Description

Specifies the name of the supplier belonging to the relation.

Syntax

`Relation`.**`SupplierName`**

Property Type

String

Relation Class Methods

The following table summarizes the Relation Class methods.

Table 126 Relation Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
HasClient	Determines whether the relation has a client
HasSupplier	Determines whether the relation has a supplier
GetClient	Retrieves the RoseItem that is the client of the relation
GetSupplier	Retrieves the RoseItem that is the supplier of the relation

Relation.GetClient Method

Description

This method retrieves the RoseItem that is the client belonging to the Relation.

Syntax

`theRoseItem = theRelation.GetClient ()`

Element	**Description**
theRoseItem As RoseItem	Returns the RoseItem that is the client belonging to the relation
theRelation As Relation	Relation whose client is being retrieved

Relation.GetSupplier Method

Description

This method retrieves the RoseItem that is the supplier belonging to the Relation.

Syntax

`theRoseItem` = *`theRelation`*.**`GetSupplier`** ()

Element	Description
theRoseItem As RoseItem	Returns the RoseItem that is the supplier belonging to the relation
theRelation As Relation	Relation whose supplier is being retrieved

Relation.HasClient Method

Description

This method indicates whether the relation has a client.

Syntax

`HasClient` = *`theRelation`*.**`HasClient`** ()

Element	Description
HasClient As Relation	Returns a value of True if the relation has a client
theRelation As Relation	Relation being checked for a client

Relation.HasSupplier Method

Description

This method indicates whether the relation has a supplier.

Syntax

HasSupplier = *theRelation*.**HasSupplier** ()

Element	Description
HasSupplier As Relation	Returns a value of True if the relation has a supplier
theRelation As Relation	Relation being checked for a supplier

Role Class

Roles denote the purpose or capacity in which one class associates with another. The role class exposes properties and methods that:

- Determine the characteristics of roles
- Allow you to retrieve roles from a model

Role Class Properties

The following table summarizes the Role Class properties.

Table 127 Role Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Relation Properties	Inherits all Relation properties
Aggregate	Indicates whether the role is an aggregate class
AssociateItem	Specifies the item to which the role belongs
Association	Specifies an association belonging to the role
Cardinality	Determines the cardinality of the role

Property	Description
Class	Specifies a class belonging to the role
Constraints	Specifies the Role’s constraints
Containment	Indicates the containment relationship of the role
ExportControl	Controls attribute visibility
Friend	Indicates whether the role is a Friend, allowing access to its non-public attributes and operations
Keys	Specifies a collection of keys belonging to the role
Navigable	Indicates whether the role is navigable
Static	Indicates that the role is static
UseCase	Specifies the use case, if any, for the role

Role.Aggregate Property

Description

Indicates whether the role is an aggregate class.

Syntax

Role.**Aggregate**

Property Type

Boolean

Role.AssociateItem Property

Description

This property specifies the item to which the role belongs.

Note: *This property is read-only.*

Syntax

Set *theItem* = *theRole*.**AssociateItem**

Property Type

RoseItem

Role.Association Property

Description

Specifies an association belonging to the role.

Note: *This property is read-only.*

Syntax

Role.**Association**

Property Type

Association

Role.Cardinality Property

Description

Specifies role cardinality.

Syntax

Role.**Cardinality**

Property Type

String

Role.Class Property

Description

Specifies a class belonging to the role.

Note: *This property is read-only.*

Syntax

Role.**Class**

Property Type

Class

Role.Constraints Property

Description

Specifies any constraints (expressions of semantic conditions that must be preserved) on the role.

Syntax

Role.**Constraints**

Property Type

String

Role.Containment Property

Description

The Containment property is a rich data type. The following table describes the valid forms of expressing the Containment rich data type for the Role class.

Table 128 Role.Containment Rich Data Types

Rich Data Type	**Description**
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As StringCollection	The list of all possible values for the collection Valid values are: ■ Unspecified ■ ByValue ■ ByReference

Note: *This property is read-only.*

Syntax

```
Role.Containment
```

Property Type

Containment

Role.ExportControl Property

Description

The ExportControl property is a rich data type that controls access to the Role object. The following table describes the valid forms of expressing the ExportControl rich data type for the Role class.

Table 129 Role.ExportControl Rich Data Types

Rich Data Type	Description
Value As Integer	The current integer value
Name As String	The current value of the class as a string
Types As RichTypeValuesCollection	The read-only list of all possible values for the collection Valid values are: ■ PublicAccess ■ ProtectedAccess ■ PrivateAccess ■ ImplementationAccess

Note: *The ExportControl property is read-only. Its Value and Name, however, are read/write.*

Syntax

```
theRole.ExportControl.Name = “PrivateAccess”
```

```
theRole.ExportControl.Value = 2
```

```
theNameStr = theRole.ExportControl.Name
```

```
theValue = theRole.ExportControl.Value
```

Property Type

RichType or RoleExportControl

Role.Keys Property

Description

Specifies the keys belonging to the role.

Note: *This property is read-only.*

Syntax

`Role.Keys`

Property Type

AttributeCollection

Role.Navigable Property

Description

Indicates whether the role is navigable.

Syntax

`Role.Navigable`

Property Type

Boolean

Role.Static Property

Description

Indicates whether the role is static.

Syntax

`Role.Static`

Property Type

Boolean

Role.UseCase Property

Description

This property specifies the use case, if any, for the role. If no use case is associated with the role, this property returns **Nothing**.

Note: *This property is read-only.*

Syntax

`Set` *`theUseCase`* `=` *`theRole`*`.`**`UseCase`**

Property Type

UseCase

Role.Friend Property

Description

Indicates whether the role is a Friend, allowing access to its non-public attributes and operations.

Syntax

`Role``.`**`Friend`**

Property Type

Boolean

Role Class Methods

The following table summarizes the Role Class methods.

Table 130 Role Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddKey	Returns a key based on a specified attribute name and type
DeleteKey	Deletes a key from a role
GetClassName	Retrieves the class name associated with the role

Role.DeleteKey Method

Description

This method deletes a key from a role.

Syntax

`Deleted` = *`theObject`*.**`DeleteKey`** (*`theAttribute`*)

Element	**Description**
Deleted As Boolean	Set to True when the key is deleted
theObject As Role	Role from which the key is being deleted
theAttribute As Attribute	Name of the attribute whose key is being deleted

Role.AddKey Method

Description

This method returns a key for a role based on a specified attribute name and type.

Syntax

`Set` *`theKey`* `=` *`theObject`*`.`**`AddKey`** `(`*`theAttrName, theAttrType`*`)`

Element	**Description**
theKey As Attribute	Returns the key as an attribute
theObject As Role	Role to which the key is being added
theAttrName As String	Name of the attribute to use as a key
theAttrType As String	Attribute type to use as a key

Role.GetClassName Method

Description

This method returns the name of the class belonging to the role.

Syntax

`theName` `=` *`theObject`*`.`**`GetClassName`** `()`

Element	**Description**
theName As String	Returns the name of the class belonging to the role. If the class does not exist, a name other than a class name may be returned by the method.
theObject As Role	Role whose class name is being retrieved

RoseAddIn Class

RoseAddIn class properties and methods describe and control the characteristics of the add-ins that are part of the currently active Rational Rose application.

For example, you can:

- Find out whether a RoseAddIn is active
- Activate or deactivate a RoseAddIn
- Define the path to the add-in's menu, property, and help files
- Execute scripts that are specific to the RoseAddIn

RoseAddIn Class Properties

The following table summarizes the RoseAddIn class properties.

Table 131 RoseAddIn Class Properties Summary

Property	Description
CompanyName	Specifies the name of the Company that created the RoseAddIn
Copyright	Specifies copyright information for the RoseAddIn
EventHandler	Specifies an instance of a custom OLE object implemented by the RoseAddIn developer to provide access to the RoseAddIn from other applications
FundamentalTypes	Specifies the collection of Fundamental Types that are specific to this RoseAddIn
HelpFilePath	Specifies the path to the RoseAddIn's help file
InstallDirectory	Directory in which the RoseAddIn's executable is installed
MenuFilePath	Specifies the path to the RoseAddIn's menu file
Name	Name of the RoseAddIn
PropertyFilePath	Specifies the path to the RoseAddIn's property file
RootRegistryPath	Specifies the complete registry tree path (from the root) that allows access to the registry entries for this RoseAddIn

Property	Description
ServerName	Specifies the OLE class name that corresponds to the RoseAddIn's EventHandler object
ToolNames	Specifies the collection of tool names belonging to the RoseAddIn. (Each tool defines its own property sets and corresponds to a tab in the property specification dialog.)
Version	Specifies the version number of the RoseAddIn

RoseAddIn.CompanyName Property

Description

Specifies the name of the company that created the RoseAddIn.

Note: *This property is read-only.*

Syntax

RoseAddIn.**CompanyName**

Property Type

String

RoseAddIn.Copyright Property

Description

Specifies the copyright information for the RoseAddIn.

Note: *This property is read-only.*

Syntax

RoseAddIn.**Copyright**

Property Type

String

RoseAddIn.EventHandler Property

Description

Specifies an instance of a custom OLE object implemented by the RoseAddIn developer to provide access to the RoseAddIn from other applications.

Note: *This property is read-only.*

Syntax

`RoseAddIn.EventHandler`

Property Type

Object

RoseAddIn.FundamentalTypes Property

Description

Specifies the collection of Fundamental Types that are specific to this RoseAddIn.

Note: *This property is read-only.*

Syntax

`RoseAddIn.FundamentalTypes`

Property Type

StringCollection

RoseAddIn.HelpFilePath Property

Description

Specifies the path to the RoseAddIn's help file.

Note: *This property is read-only.*

Syntax

RoseAddIn.**HelpFilePath**

Property Type

String

RoseAddIn.InstallDirectory Property

Description

Specifies the directory in which the RoseAddIn's executable is installed.

Note: *This property is read-only.*

Syntax

RoseAddIn.**InstallDirectory**

Property Type

String

RoseAddIn.MenuFilePath Property

Description

Specifies the path to the RoseAddIn's menu file.

Note: *This property is read-only.*

Syntax

`RoseAddIn.MenuFilePath`

Property Type

String

RoseAddIn.Name Property

Description

Specifies the name of the RoseAddIn.

Note: *This property is read-only.*

Syntax

`RoseAddIn.Name`

Property Type

String

RoseAddIn.PropertyFilePath Property

Description

Specifies the path to the RoseAddIn's property file.

Note: *This property is read-only.*

Syntax

RoseAddIn.**PropertyFilePath**

Property Type

String

RoseAddIn.RootRegistryPath Property

Description

Specifies the complete registry tree path (from the root) that allows access to the registry entries for this RoseAddIn.

Note: *This property is read-only.*

Syntax

RoseAddIn.**RootRegistryPath**

Property Type

String

RoseAddIn.ServerName Property

Description

Specifies the OLE class name (created by the RoseAddIn developer) that corresponds to the RoseAddIn's EventHandler object.

Note: *This property is read-only.*

Syntax

RoseAddIn.**ServerName**

Property Type

String

RoseAddIn.ToolNames Property

Description

Specifies the collection of tool names belonging to the RoseAddIn. (Each tool defines its own property sets and corresponds to a tab in the property specification dialog.)

Note: *This property is read-only.*

Syntax

RoseAddIn.**ToolNames**

Property Type

StringCollection

RoseAddIn.Version Property

Description

Specifies the version number of the RoseAddIn.

Note: *This property is read-only.*

Syntax

```
RoseAddIn.Version
```

Property Type

String

RoseAddIn Class Methods

The following table summarizes the RoseAddIn Class methods.

Table 132 RoseAddIn Class Methods Summary

Method	**Description**
RoseObject methods	Inherits all RoseObject methods
Activate	Activates the specified RoseAddIn
AddContextMenuItem	Adds the specified ContextMenuItem to the Rational Rose shortcut menu
Deactivate	Deactivates the specified RoseAddIn
ExecuteScript	Executes the source or compiled image of a script that resides in the RoseAddIn's install directory
GetContextMenuItems	Returns the ContextMenuItemCollection for the specified Rational Rose add-in
GetDisplayName	Returns the RoseAddIn's Display Name, as shown in the RoseAddIn manager
IsActive	Determines whether the specified RoseAddIn is currently active

Method	Description
IsLanguageAddIn	Determines whether the specified RoseAddIn is a programming language
ReadSetting	Retrieves a registry setting for this RoseAddIn
WriteSetting	Creates a registry entry for this RoseAddIn

RoseAddIn.Activate Method

Description

This subroutine activates the specified RoseAddIn.

Syntax

theAddIn.**Activate**

Element	Description
theAddIn As RoseAddIn	RoseAddIn to activate

RoseAddIn.AddContextMenuItem Method

Description

This method creates and adds the specified ContextMenuItem to the Rational Rose shortcut menu.

Note: *This call is only available for Rational Rose add-ins, i.e., those Rational Rose Automation clients who are registered in the RoseAddIn Manager.*

An add-in should add context menu items when it gets the OnActivate event.

Syntax

Set *theCntxtMenuItem* = *theRoseAddIn*.**AddContextMenuItem** (*itemType, fullCaption, internalName*)

Element	Description
theCntxtMenuItem As ContextMenuItem	Returns a ContextMenuItem.
theRoseAddIn As RoseAddIn	Add-in automation client that is adding the menu item to the shortcut menu.
itemType As ContextMenuItemType	Enumeration that defines default values corresponding to the types of items to which shortcut menu options can be applied.
fullCaption As String	Keyword or Text to be displayed on the shortcut menu. **Valid values**: ■ "*caption*" Text to display on the shortcut menu ■ "**Submenu** *caption*" Keyword, **Submenu**, followed by the text to display on the shortcut menu. When this item is chosen, Rational Rose displays a submenu. ■ "**EndSubmenu**" Use keyword, **EndSubmenu**, after the last element in a submenu of the shortcut menu. ■ "**Separator**" Keyword, **Separator**, draws a line to separate the menu options above from those below. ***Note:*** *To define access keys, place the ampersand (&) in front of the letter in the shortcut menu caption. As a result, when Rational Rose displays that shortcut menu item, the appropriate letter is underlined.*
internalName As String	String that represents the internal name of the menu item. This string is NOT localized. It is used to indicate which menu item is selected by the user. It is up to the add-in to map this internal name to the requested action. For **Submenus**, **Separators**, and **EndSubmenus**, use an "empty" name here (""), since these types of items do not map to add-in defined actions.

RoseAddIn.Deactivate Method

Description

This subroutine deactivates the specified RoseAddIn.

Syntax

theAddIn.**Deactivate**

Element	Description
theAddIn As RoseAddIn	RoseAddIn to deactivate

RoseAddIn.ExecuteScript Method

Description

This subroutine executes the source or compiled image of a script contained in the specified file. You can specify the file without its extension. If the script is currently open in the script editor, Rational Rose will execute the open script. Otherwise, Rational Rose will search for the source script (.ebs) and execute it, if found. If not found, Rational Rose will search for and execute the compiled script (.ebx file).

Syntax

theAddIn.**ExecuteScript** *FileName*

Element	Description
theAddIn As RoseAddIn	RoseAddIn in which the script is being executed
FileName As String	File that contains the script to be executed

RoseAddIn.GetContextMenuItems Method

Description

This method returns the collection of ContextMenuItems from the specified Rational Rose add-in for the specified type of shortcut menu item. After using GetContextMenuItems, you can iterate through the collection by using GetAt and set the MenuState property accordingly.

Note: *This is a read-only collection.*

Syntax

```
Set theCntxtMenuItems = theRoseAddIn.GetContextMenuItems
       (itemType)
```

Element	Description
theCntxtMenuItems As ContextMenuItemCollection	Returns the collection of ContextMenuItems for the specified add-in
theRoseAddIn As RoseAddIn	Add-in from which to retrieve ContextMenuItems
itemType As ContextMenuItemType	Enumeration that defines the values corresponding to the types of items to which shortcut menu options can be applied

RoseAddIn.GetDisplayName Method

Description

This method returns the add-in's Display Name, as shown in the Rational Rose Add-In Manager.

Syntax

`Set` *`theDisplayName`* `=` *`theRoseAddIn`*`.`**`GetDisplayName`** `()`

Element	**Description**
theDisplayName As String	Returns the name displayed in the Rational Rose Add-In Manager and the Registry for the specified add-in. This is NOT the same as the name returned by the Name property of RoseAddIn.
theRoseAddIn As RoseAddIn	RoseAddIn from which to retrieve the display name.

RoseAddIn.IsActive Method

Description

This method determines whether the specified RoseAddIn is currently active.

Syntax

`IsActive` `=` *`theAddIn`*`.`**`IsActive`** `()`

Element	**Description**
IsActive As Boolean	Returns a value of True if the specified RoseAddIn is currently active
theAddIn As RoseAddIn	RoseAddIn being checked

RoseAddIn.IsLanguageAddIn Method

Description

This method determines whether the specified RoseAddIn is a programming language.

Syntax

`IsLanguage = theAddIn.IsLanguageAddIn ()`

Element	Description
IsLanguage As Boolean	Returns a value of True if the specified RoseAddIn is a programming language
theAddIn As RoseAddIn	RoseAddIn being checked

RoseAddIn.ReadSetting Method

Description

This method retrieves a registry setting for this RoseAddIn, given a section, entry, and default value.

Syntax

`theString = theAddIn.ReadSetting (Section, Entry, Default)`

Element	Description
theString As String	Returns the actual value of registry setting given its section, entry, and default value. If no corresponding entry exists, returns the specified default value
theAddIn As RoseAddIn	The RoseAddIn whose registry entry is being retrieved
theSection As String	Section name of the registry entry. For example, PathMap
theEntry As String	Name of the entry. For example, $SCRIPT_PATH
theDefault As String	Default value of the entry

RoseAddIn.WriteSetting Method

Description

This method creates a custom registry setting for the RoseAddIn, given a section, entry, and default value for the setting.

Syntax

```
IsWritten = theAddIn.WriteSetting (theSection, theEntry,
      theValue)
```

Element	Description
IsWritten As Boolean	Returns a value of True when the entry is successfully added to the registry
theAddIn As RoseAddIn	RoseAddIn for which the registry setting is being created
theSection As String	User-defined section name for the custom entry
theEntry As String	User-defined entry name
theValue As String	User-defined default value for the custom entry

RoseAddInEventTypes Enumeration

RoseAddInEventTypes is an enumeration that defines values corresponding to the types of events that are sent to Rational Rose Add-Ins. RoseAddInEventTypes can be used in the DisableEvents and EnableEvents RoseAddInManager methods.

The following table describes the valid values for the RoseAddInEventTypes enumeration.

Table 133 RoseAddInEventTypes Enumeration Valid Values

Value	Integer Value	Description
rsOnNewModel	2	Represents the OnNewModel event For example, use **rsOnNewModel** in the DisableEvents and EnableEvents RoseAddInManager methods to disable and enable the OnNewModel event

Note: *Rational Rose Automation users may use the value (e.g.,* ***rsOnNewModel****) in their methods. Rational Rose Script users must use the integer value (e.g., 2) in their methods.*

RoseAddInManager Class

The RoseAddInManager class has a single property, the AddIns property, which contains the collection of AddIns available to the currently active Rational Rose executable.

The RoseAddInManager class inherits all RoseObject methods, but has no methods of its own.

RoseAddInManager Class Properties

The following table summarizes the RoseAddInManager Class properties.

Table 134 RoseAddInManager Class Properties Summary

Property	**Description**
AddIns	Specifies the collection of RoseAddIns managed by the RoseAddInManager

RoseAddInManager.AddIns Property

Description

This property specifies the collection of add-ins managed by the RoseAddInManager.

Note: *This property is read-only.*

Syntax

```
Set theAddInCollection = theRoseAddInManager.AddIns
```

Property Type

RoseAddInCollection

RoseAddInManager Class Methods

The following table summarizes the RoseAddInManager Class methods.

Table 135 RoseAddInManager Class Methods Summary

Method	**Description**
RoseObject methods	Inherits all RoseObject class methods
DisableEvents	Disables the specified event
EnableEvents	Enables the specified event

RoseAddInManager.DisableEvents Method

Description

This method disables the specified event. In other words, this method tells Rational Rose to stop sending messages for this event to your OLE server.

Warning! *Use this method with care. Make sure you re-enable the event before you need to use it again.*

Examples

```
1. Set theDisabledEvent = theRoseAddInMgr.DisableEvents
   (rsOnNewModel)
2. Set theDisabledEvent = theRoseAddInMgr.DisableEvents
   (0010)
```

Syntax

Set *theDisabledEvent* = *theRoseAddInMgr*.**DisableEvents**
    (*theEvent*)

Element	**Description**
theDisabledEvent As Long	Returns the disabled event Store this value for use in the EnableEvents method
theRoseAddInMgr As RoseAddInManager	Specifies the RoseAddInManager
theEvent As Long	Indicates the Events to be disabled

RoseAddInManager.EnableEvents Method

Description

This method enables the specified event. In other words, this method tells Rational Rose to start sending messages again for this event to your OLE server.

Examples

```
1. Set theEnabledEvent = theRoseAddInMgr.EnableEvents
   (theDisabledEvent)
2. Set theEnabledEvent = theRoseAddInMgr.EnableEvents
   (rsOnNewModel)
3. Set theEnabledEvent = theRoseAddInMgr.EnableEvents
   (0010)
```

Note: *In example 1, `theDisabledEvent` is the value returned from an earlier DisableEvents call. We can now use it to quickly enable the disabled event.*

Syntax

`Set` *`theEnabledEvent`* = *`theRoseAddInMgr`*.**`EnableEvents`**
(*`theEvent`*)

Element	Description
theEnabledEvent As Long	Returns the Event that was enabled
theRoseAddInMgr As RoseAddInManager	Specifies the RoseAddInManager
theEvent As Long	Indicates the Event to be enabled

RoseItem Class

Just about every RoseItem is a model element and therefore inherits all Element properties and methods. Use RoseItem properties and methods to specify or manipulate RoseItem documentation, stereotypes, external documents, as well as to open an item's specification

RoseItem Class Properties

The following table summarizes the RoseItem Class properties.

Table 136 RoseItem Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
Documentation	Specifies the documentation belonging to the item
ExternalDocuments	Specifies the collection of external documents belonging to the item
LocalizedStereotype	Localized equivalent of the stereotype property
StateMachineOwner	Specifies the state machine owner associated with the item
Stereotype	Specifies the RoseItem's stereotype

RoseItem.Documentation Property

Description

Specifies the documentation belonging to the RoseItem.

Syntax

RoseItem.**Documentation**

Property Type

String

RoseItem.ExternalDocuments Property

Description

Specifies the external documents belonging to the RoseItem.

Note: *This property is read-only.*

Syntax

RoseItem.**ExternalDocuments**

Property Type

ExternalDocumentCollection

RoseItem.LocalizedStereotype Property

Description

Specifies the localized equivalent of the RoseItem stereotype.

Syntax

RoseItem.**LocalizedStereotype**

Property Type

String

RoseItem.StateMachineOwner Property

Description

This property specifies the state machine owner associated with the item. StateMachineOwner is used to retrieve the item's state machine. You can then use the state machine to retrieve:

- activity and statechart diagrams
- activities
- states
- decisions
- synchronizations
- swimlanes

Note: *This property is read-only.*

Syntax

`Set` *`myStateMachineOwner`* = *`myRoseItem`*.**`StateMachineOwner`**

Property Type

StateMachineOwner

RoseItem.Stereotype Property

Description

Specifies the stereotype of the RoseItem.

Syntax

`RoseItem`.**`Stereotype`**

Property Type

String

RoseItem Class Methods

The following table summarizes the RoseItem Class methods.

Table 137 RoseItem Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
AddExternalDocument	Adds an external document to an item
DeleteExternalDocument	Deletes an external document from an item
GetRoseItem	Retrieves the Rational Rose item as an object
OpenCustomSpecification	If a custom specification exists, opens the custom specification window for a RoseItem
OpenSpecification	Opens the Rational Rose default specification window for a RoseItem

RoseItem.AddExternalDocument Method

Description

This method creates a new external document and adds it to a RoseItem.

Syntax

```
Set theExternalDoc = theObject.AddExternalDocument
      (theName, theType)
```

Element	Description
theExternalDoc As ExternalDocument	Returns the ExternalDocument object added to the Rational Rose item
theObject As RoseItem	Rational Rose item to which the document is being added
theName As String	Name of the document being added
theType As Integer	Type of document being added Valid values are: 1 = Path 2 = URL

RoseItem.DeleteExternalDocument Method

Description

This method deletes an external document from a RoseItem.

Syntax

`Deleted = theObject.DeleteExternalDocument (theDocument)`

Element	Description
deleted As Boolean	Returns a value of true when the document is deleted from the RoseItem
theObject As RoseItem	Rose item from which the document is being deleted
theDocument As ExternalDocument	Instance of the document being deleted

RoseItem.GetRoseItem Method

Description

This method retrieves a RoseItem as an object.

Note: *Use this method to convert classes derived from RoseItem into RoseItem objects.*

Syntax

`Set theRoseItem = theObject.GetRoseItem ()`

Element	Description
theRoseItem As RoseItem	Returns the Rational Rose item as an object
theObject As RoseItem	Instance of the Rational Rose item being returned

RoseItem.OpenCustomSpecification Method

Description

If you have defined a custom specification window in your add-in, this method opens your add-in's custom specification window for the specified RoseItem. To open the Rational Rose default specification window, use OpenSpecification.

Syntax

isOpened = *theRoseItem*.**OpenCustomSpecification** ()

Element	Description
isOpened As Boolean	Returns a value of True when your add-in's custom specification is successfully opened
theRoseItem As RoseItem	RoseItem whose custom specification is being opened

RoseItem.OpenSpecification Method

Description

This method opens the Rational Rose default specification window for the specified RoseItem. To open your add-in's custom specification window, use OpenCustomSpecification.

Syntax

isOpened = *theRoseItem*.**OpenSpecification** ()

Element	Description
isOpened As Boolean	Returns a value of True when the Rational Rose default specification is successfully opened
theRoseItem As RoseItem	RoseItem whose Rational Rose default specification is being opened

RoseItemView Class

The RoseItemView class exposes properties and methods that determine the size and placement of a RoseItem on a diagram.

RoseItemView Class Properties

The following table summarizes the RoseItemView Class properties.

Table 138 RoseItemView Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
Name	Specifies the name of the item view
Item	Specifies the Rational Rose item represented by the RoseItemView
ParentDiagram	Specifies the diagram that contains the RoseItemView
ParentView	Specifies the RoseItemView that contains the RoseItemView
StereotypeDisplay	Indicates how to display the Rose item view's stereotype
SubViews	Specifies the collection of item views that belong to the RoseItemView
XPosition	Specifies the value of the horizontal coordinate (x) for the center point of the view
YPosition	Specifies the value of the vertical coordinate (y) for the center point of the view
Height	Specifies the height of the Rational Rose item view
Width	Specifies the width of the Rational Rose item view
FillColor.Red	Sets the item view fill color to red
FillColor.Green	Sets the item view fill color to green
FillColor.Blue	Sets the item view fill color to blue

Property	**Description**
FillColor.Transparent	Sets the item view fill color to transparent
LineColor.Red	Sets the item view line color to red
LineColor.Green	Sets the item view line color to green
LineColor.Blue	Sets the item view line color to blue
Font.Red	Sets the text color to red
Font.Green	Sets the text color to green
Font.Blue	Sets the text color to blue
Font.FaceName	Sets the text's font name (such as Arial, Courier, etc.)
Font.Size	Sets the text's point size
Font.Bold	Indicates whether the text's font style is **Bold**
Font.Italic	Indicates whether the text's font style is *Italic*
Font.Underline	Indicates whether the text's font style is Underline
Font.StrikeThrough	Indicates whether the text's font style is ~~StrikeThrough~~
LineVertices	Specifies the collection of line vertices of the RoseItemView's associations and relationships

RoseItemView.Height Property

Description

Specifies the height of the Rational Rose item view object.

Syntax

```
RoseItemView.Height
```

Property Type

Integer

RoseItemView.LineVertices Property

Description

This property specifies the collection of line vertices of the RoseItemView's associations and relationships. For an example of how to use this property, see the sample code in the LineVertex class section of this reference guide.

Note: *This property is read-only.*

Syntax

`Set` *`theLineVertexColection`* = *`theRoseItemView`*.**`LineVertices`**

Property Type

LineVertexCollection

RoseItemView.Name Property

Description

Specifies the name of the Rational Rose item view.

Syntax

`RoseItemView`.**`Name`**

Property Type

String

RoseItemView.StereotypeDisplay Property

Description

This property indicates how to display the Rose item view's stereotype.

Syntax

```
intDisplayType = objRoseItemView.StereotypeDisplay
objRoseItemView.StereotypeDisplay = 3
```

Property Type

Integer with the following values:

Value	Result
0	None
1	Label
2	Decoration
3	Icon

RoseItemView.Width Property

Description

Specifies the width of the Rational Rose item view.

Syntax

```
RoseItemView.Width
```

Property Type

Integer

RoseItemView.XPosition Property

Description

Specifies the value of the horizontal coordinate (x) for the center point of the view.

Syntax

RoseItemView.**XPosition**

Property Type

Integer

RoseItemView.YPosition Property

Description

Specifies the value of the vertical coordinate (y) for the center point of the view.

Syntax

RoseItemView.**YPosition**

Property Type

Integer

Values for RoseItemView Color Properties

The following sections describe properties that you can use to define the color of RoseItemView objects in a diagram.

Each of the three available colors (Red, Blue, and Green) can have a value from 0 to 255. You can define the presence or absence of a particular color by setting its value between 0 (no color) and 255 (pure color). For example:

RoseItemView.LineColor.Red = 0 defines a line color that has no trace of red in it.

RoseItemView.LineColor.Red = 255 defines a pure red line.

RoseItemView.LineColor.Red = 100 defines a somewhat red line.

In addition, you can create custom colors by assigning levels of multiple colors to the same object. For example, the following three statements, when taken together, define a fill color that has red as the dominant color, but also contains some green and a trace of blue:

RoseItemView.FillColor.Red = 100

RoseItemView.FillColor.Green = 50

RoseItemView.FillColor.Blue = 25

Note: *Rational Rose uses color the way most color applications would use it. These principles are not unique to Rational Rose or the Rational Rose Extensibility Interface.*

RoseItemView.FillColor.Blue Property

Description

Specifies the amount of blue to use in the fill color for the RoseItemView object.

Syntax

`RoseItemView`.**`FillColor`**.**`Blue`**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.FillColor.Green Property

Description

Specifies the amount of green to use in the fill color for the RoseItemView object.

Syntax

`RoseItemView`.**`FillColor.Green`**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.FillColor.Red Property

Description

Specifies the amount of red to use in the fill color for the RoseItemView object.

Syntax

`RoseItemView`.**`FillColor.Red`**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.FillColor.Transparent Property

Description

Indicates whether the fill color of the RoseItemView object is transparent.

Syntax

`RoseItemView.FillColor.Transparent`

Property Type

Boolean

RoseItemView.Font.Blue Property

Description

Specifies the amount of blue to use in the text color of a RoseItemView object.

Syntax

`RoseItemView.Font.Blue`

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.Font.Bold Property

Description

Indicates whether the text's font style is **Bold**.

Syntax

`RoseItemView``.`**`Font.Bold`**

Property Type

Boolean

RoseItemView.Font.FaceName Property

Description

Specifies the text font name (such as Arial, Courier, etc.) of a RoseItemView object.

Syntax

`RoseItemView``.`**`Font.FaceName`**

Property Type

String

RoseItemView.Font.Green Property

Description

Specifies the amount of green to use in the text color of a RoseItemView object.

Syntax

`RoseItemView`.**`Font.Green`**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.Font.Italic Property

Description

Indicates whether the text's font style is *Italic*.

Syntax

`RoseItemView`.**`Font.Italic`**

Property Type

Boolean

RoseItemView.Font.Red Property

Description

Specifies the amount of red to use in the text color of a RoseItemView object.

Syntax

RoseItemView.**Font.Red**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.Font.Size Property

Description

Specifies the text point size for a RoseItemView object.

Syntax

RoseItemView.**Font.Size**

Property Type

String

RoseItemView.Font.StrikeThrough Property

Description

Indicates whether the text's font style is ~~Strikethrough~~.

Syntax

RoseItemView.**Font.StrikeThrough**

Property Type

Boolean

RoseItemView.Font.Underline Property

Description

Indicates whether the text's font style is Underline.

Syntax

RoseItemView.Font.Underline

Property Type

Boolean

RoseItemView.Item Property

Description

Specifies the RoseItem represented by this RoseItemView.

Note: *This property is read-only.*

Syntax

`RoseItemView`.**`Item`**

Property Type

RoseItem

RoseItemView.LineColor.Blue Property

Description

Specifies the amount of blue to use in the line color for the RoseItemView object.

Syntax

RoseItemView.**LineColor.Blue**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.LineColor.Green Property

Description

Specifies the amount of green to use in the line color for the RoseItemView object.

Syntax

RoseItemView.**LineColor.Green**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.LineColor.Red Property

Description

Specifies the amount of red to use in the line color for the RoseItemView object.

Syntax

RoseItemView.**LineColor.Red**

Property Type

Integer

Note: *See the previous discussion on values for RoseItemView color properties.*

RoseItemView.ParentDiagram Property

Description

Specifies the diagram that contains this RoseItemView.

Note: *This property is read-only.*

Syntax

RoseItemView.**ParentDiagram**

Property Type

Diagram

RoseItemView.ParentView Property

Description

Specifies the RoseItemView that contains this RoseItemView.

Note: *This property is read-only.*

Syntax

RoseItemView.**ParentView**

Property Type

RoseItemView

RoseItemView.SubViews Property

Description

Specifies the collection of item views that belong to the RoseItemView.

Note: *This property is read-only.*

Syntax

RoseItemView.**SubViews**

Property Type

ItemViewCollection

RoseItemView Class Methods

The following table summarizes the RoseItemView Class methods.

Table 139 RoseItemView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element methods
GetAttachedNotes	Retrieves the collection of note views attached to a Rose item view
GetDefaultHeight	Retrieves the default height of the RoseItemView (calculated by Rational Rose)
GetDefaultWidth	Retrieves the default width of the RoseItemView (calculated by Rational Rose)
GetMinHeight	Retrieves the minimum height of the RoseItemView (calculated by Rational Rose)
GetMinWidth	Retrieves the minimum width of the RoseItemView (calculated by Rational Rose)
HasItem	Indicates whether the RoseItemView has a corresponding RoseItem
HasParentView	Indicates whether the RoseItemView belongs to another RoseItemView
Invalidate	Redraws the specified RoseItemView on the computer screen
IsSelected	Indicates whether the RoseItemView is currently selected in the diagram
PointInView	Determines whether a given x,y coordinate lies within the specified RoseItemView
SetSelected	Selects the RoseItemView in the diagram
SupportsFillColor	Allows the RoseItemView to use fill color, if appropriate
SupportsLineColor	Allows the RoseItemView to use line color, if appropriate

RoseItemView.Invalidate Method

Description

This subroutine redraws the RoseItemView on the screen.

Syntax

```
theObject.Invalidate
```

Element	Description
theObject As RoseItemView	Instance of the RoseItemView being redrawn

RoseItemView.GetAttachedNotes Method

Description

This method retrieves the collection of note views that are attached, by a note anchor, to a Rose item view.

Syntax

```
Set colNoteViews = objRoseItemView.GetAttachedNotes ()
```

Element	Description
colNoteViews As NoteViewCollection	Collection of note views attached to a Rose item view
objRoseItemView As RoseItemView	Rose item view whose attached note views are being retrieved

RoseItemView.GetDefaultHeight Method

Description

This method retrieves the ideal height of the RoseItemView object, based on the object's formatting. This value is calculated by Rational Rose and cannot be set.

Syntax

theHeight = *theRoseItemView*.**GetDefaultHeight** ()

Element	Description
theHeight As Integer	Returns the ideal height of the RoseItemView, given the formatting of the object
theRoseItemView As RoseItemView	Specifies the RoseItemView whose ideal height you are determining

RoseItemView.GetDefaultWidth Method

Description

This method retrieves the ideal width of the RoseItemView object, based on the object's formatting. This value is calculated by Rational Rose and cannot be set.

Syntax

theWidth = *theRoseItemView*.**GetDefaultWidth** ()

Element	Description
theWidth As Integer	Returns the ideal width of the RoseItemView, given the formatting of the object
theRoseItemView As RoseItemView	Specifies the RoseItemView whose ideal width you are determining

RoseItemView.GetMinHeight Method

Description

This method retrieves the minimum height of the RoseItemView object, based on the object's formatting. This value is calculated by Rational Rose and cannot be set.

Syntax

theHeight = *theRoseItemView*.**GetMinHeight** ()

Element	**Description**
theHeight As Integer	Returns the minimum height of the RoseItemView, given the formatting of the object
theRoseItemView As RoseItemView	Specifies the RoseItemView whose minimum height you are determining

RoseItemView.GetMinWidth Method

Description

This method retrieves the minimum width of the RoseItemView object, based on the object's formatting. This value is calculated by Rational Rose and cannot be set.

Syntax

theWidth = *theRoseItemView*.**GetMinWidth** ()

Element	**Description**
theWidth As Integer	Returns the minimum width of the RoseItemView, given the formatting of the object
theRoseItemView As RoseItemView	Specifies the RoseItemView whose minimum width you are determining

RoseItemView.HasItem Method

Description

This method indicates whether the RoseItemView has a corresponding RoseItem.

Syntax

```
HasItem = theRoseItemView.HasItem ()
```

Element	Description
HasItem As Boolean	Returns a value of True if the RoseItemView has a corresponding RoseItem
theRoseItemView As RoseItemView	Specifies the RoseItemView being checked for a RoseItem

RoseItemView.HasParentView Method

Description

This method indicates whether the RoseItemView belongs to another RoseItemView.

Syntax

```
HasParentView = theRoseItemView.HasParentView ()
```

Element	Description
HasParentView As Boolean	Returns a value of True if the RoseItemView belongs to another RoseItemView
theRoseItemView As RoseItemView	Specifies the RoseItemView being checked for a parent view

RoseItemView.IsSelected Method

Description

This method indicates whether the RoseItemView is currently selected in the diagram.

Syntax

IsSelected = *theRoseItemView*.**IsSelected** ()

Element	Description
IsSelected As Boolean	Returns a value of True if the RoseItemView is currently selected in the diagram
theRoseItemView As RoseItemView	Specifies the RoseItemView being checked for in the diagram

RoseItemView.PointInView Method

Description

This method determines whether a given x,y coordinate lies within the specified RoseItemView.

Syntax

IsInView = *theRoseItemView*.**PointInView** ()

Element	Description
IsInView As Boolean	Returns a value of True if the given x,y coordinate lies within the specified RoseItemView
theRoseItemView As RoseItemView	Specifies the RoseItemView being checked for a RoseItem

RoseItemView.SetSelected Method

Description

This subroutine selects the given RoseItemView in the diagram.

Syntax

`theRoseItemView.SetSelected Selected`

Element	Description
theRoseItemView As RoseItemView	RoseItemView to select
Selected As Boolean	Set to True to select the RoseItemView in the diagram; set to False to deselect the RoseItemView in the diagram

RoseItemView.SupportsFillColor Method

Description

This method causes the RoseItemView to support fill color, if the type of RoseItemView can support fill color. For example, a RoseItemView that represents a class can use a fill color. However, a RoseItemView that represents a relationship line cannot support fill color. (It can, however, support a line color.)

Syntax

`SupportsFill = theRoseItemView.SupportsFillColor ()`

Element	Description
SupportsFill As Boolean	Returns a value of True if the specified RoseItemView is to support a fill color
theRoseItemView As RoseItemView	Specifies the RoseItemView to support fill color

RoseItemView.SupportsLineColor Method

Description

This method causes the RoseItemView to support line color, if the type of RoseItemView can support line color. For example, a RoseItemView that represents a relationship line can support line color. However, a RoseItemView that displays a metafile cannot support a line color.

Syntax

SupportsLine = *theRoseItemView*.**SupportsLineColor** ()

Element	Description
SupportsLine As Boolean	Returns a value of True if the specified RoseItemView supports line color
theRoseItemView As RoseItemView	Specifies the RoseItemView to support line color

RoseObject Class

Most elements in a Rational Rose model derive, either directly or indirectly, from the RoseObject class. When you retrieve a model element as an object, you may not know what type of object you have retrieved.

Using RoseObject class methods, you can:

- Retrieve an object as a Rational Rose object
- Determine the type of the object
- Convert the object back into its original type

Predetermining the type of an object can be particularly useful in avoiding runtime errors in Rational Rose Script.

RoseObject Class Properties

There are no RoseObject Class properties.

RoseObject Class Methods

The following table summarizes the RoseObject Class methods.

Table 140 RoseObject Class Methods Summary

Method	**Description**
CanTypeCast	Determines whether a RoseObject can be set to a specified type (Rational RoseScript only)
GetObject	Retrieves the object's OLE automation object
IdentifyClass	Identifies the class of a Rational RoseObject
IsClass	Checks whether a Rational RoseObject is an instance of a specified class
TypeCast	Converts a Rational RoseObject to its original type (Rational RoseScript only)

RoseObject.CanTypeCast Method

Description

This method determines whether a RoseObject can be set to a specified type. Checking for proper typecasting can prevent runtime errors in Rational Rose scripts.

Note: *Valid for Rational Rose Script only. For automation, use the IsClass or the IdentifyClass method.*

Syntax

CanTypeCast = *theRoseObject*.**CanTypeCast** (*theVariant*)

Element	Description
CanTypeCast As Boolean	Returns a value of True if the specified RoseObject can be typecast as the specified variant
theRoseObject As RoseObject	RoseObject whose type casting is being checked
theVariant As Variant	Variant whose data type is being compared to the original type of the RoseObject. ***Note:*** *A variant is a data type used to declare variables that can hold one of many different types of data.* Check the online help index to find more information on Variants.

Example

The following Rational RoseScript code fragment uses CanTypeCast to check whether an unknown object is an Element. If the object's type is Element, the TypeCast method is used to convert the object into an Element.

```
Sub MyProc (theObject As RoseObj)
   Dim the Element As Element
   if theObject.CanTypeCast (theElement) then
      Set theElement = theObject.TypeCast (theElement)
   end if
end Sub
```

RoseObject.GetObject Method

Description

This method retrieves the object's OLE interface object.

Note: *This method is only valid for Rational Rose Script; it has no meaning in Rational Rose Automation.*

Syntax

```
Set theOLEObject = theObject.GetObject ()
```

Element	Description
theOLEObject As Object	Returns the OLE automation interface object associated with the specified object
theObject As Element	Instance of the object whose OLE interface object is being returned

RoseObject.IdentifyClass Method

Description

This method identifies the class of a Rational RoseObject.

Note: *For Rational RoseScript, use the CanTypeCast method*

Syntax

```
theString = theRoseObject.IdentifyClass ()
```

Element	Description
theString As String	Returns the RoseObject's class name
theRoseObject As RoseObject	RoseObject whose class is being identified

RoseObject.IsClass Method

Description

This method determines whether an object is a specified class.

Note: *For Rational RoseScript, use the CanTypeCast method.*

Syntax

`IsClass = theRoseObject.IsClass (theClassName)`

Element	Description
IsClass As Boolean	Returns a value of True if its class matches the specified class name
theRoseObject As RoseObject	RoseObject whose class is being checked
theClassName As String	Name of the class for which the RoseObject is being checked

Example

The following Rational Rose automation code fragment uses the IsClass method to check an object's class and conditionally convert the object to that class.

```
If theObject.IsClass ("Element") then
   Set theElement = theObject
end If
```

RoseObject.TypeCast Method

Description

This method converts a Rational Rose object to a specified type and returns it as a variant. A variant is a data type used to declare variables that can hold one of many different types of data.

Check the online help index to find more information on Variants.

Note: *Valid for Rational Rose Script only. For automation, use the IsClass method and then set the object equal to the class.*

Syntax

```
Set theVariant = theRoseObject.TypeCast (theParameter)
```

Element	**Description**
theVariant As Variant	Returns the type as a variant
theRoseObject As RoseObject	Object whose type is being retrieved
theParameter As Variant	Type to which the object will be converted (for example, Element, Operation, Subsystem, etc.)

ScenarioDiagram Class

A scenario is an instance of a use case; it is an outline of events that occur during system execution.

Scenario diagrams allow you to create a visual representation of a scenario. The scenario diagram class exposes properties and methods that allow you to create scenario diagrams and add and delete messages and objects to them.

ScenarioDiagram Class Properties

The following table summarizes the ScenarioDiagram Class properties.

Table 141 ScenarioDiagram Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
Diagram Class Properties	Inherits all Diagram Class properties
InstanceViews	Contains the collection of instance views belonging to the diagram

ScenarioDiagram.InstanceViews Property

Description

Contains the collection of instance views belonging to the object instance.

Note: *This property is read-only.*

Syntax

ObjectInstance.**InstanceViews**

Property Type

InstanceViewCollection

ScenarioDiagram Class Methods

The following table summarizes the ScenarioDiagram Class methods.

Table 142 ScenarioDiagram Class Methods Summary

Method	**Description**
Element Methods	Inherits all Element methods
Diagram Methods	Inherits all Diagram methods
AddInstance	Creates and adds an object instance view to a scenario diagram
AddInstanceView	Adds an existing object instance view to a scenario diagram
CreateMessage	Creates a message and adds it to a scenario diagram
DeleteInstance	Deletes an object instance from the scenario diagram
GetDiagramType	Retrieves the value of the type of diagram
GetMessages	Retrieves the collection of messages that belong to the scenario diagram
GetSelectedLinks	Retrieves the currently selected links from the scenario diagram
GetSelectedMessages	Retrieves the set of messages currently selected in the scenario diagram
GetSelectedObjects	Retrieves the set of objects currently selected in the scenario diagram
RemoveInstanceView	Removes an instance view from the scenario diagram

ScenarioDiagram.AddInstance Method

Description

This method creates a new object instance and adds a view of the new object instance to a scenario diagram. To add a view of an existing object instance to a scenario diagram, use the **ScenarioDiagram.AddInstanceView** method.

Syntax

```
Set objObjectInstance = objScenarioDiagram.AddInstance
      (strName, strClassName)
```

Element	Description
objObjectInstance As ObjectInstance	Object instance that was created and whose view was added to the diagram
objScenarioDiagram As ScenarioDiagram	Scenario diagram to which the newly created object instance's view is being added
strName As String	Name of the object instance to be added
strClassName As String	Name of the Class to which the object instance belongs. To avoid associating the object instance with a particular class, use "" for this parameter.

ScenarioDiagram.AddInstanceView Method

Description

This method adds a view of an existing object instance to a scenario diagram. To create a new object instance and add its view to a scenario diagram, use the **ScenarioDiagram.AddInstance** method.

Syntax

```
Set objInstanceView = objScenarioDiagram.AddInstanceView
      (objObjectInstance, blnAsClassInstance)
```

Element	**Description**
objInstanceView As InstanceView	Object instance view being added to the diagram
objScenarioDiagram As ScenarioDiagram	Scenario diagram to which the instance view is being added
objObjectInstance As ObjectInstance	Existing object instance for which the instance view is being added to the diagram
blnAsClassInstance As Boolean	Set to True if the object instance is a class instance

ScenarioDiagram.CreateMessage Method

Description

This method creates a new message in a scenario diagram and returns it in the specified message.

Syntax

```
Set theMessage = theScenarioDiagram.CreateMessage (theName,
      theSender, theReceiver, theSequence)
```

Element	Description
theMessage As Message	Returns the newly created message
theScenarioDiagram As ScenarioDiagram	ScenarioDiagram to which the new message is being added
theName As String	Name of the message to be created
theSender As ObjectInstance	Object that is to send the message
theReceiver As ObjectInstance	Object that is to receive the message
theSequence As Integer	Position of the message relative to other messages in the diagram. For example, if theSequence = 3, the message will be the third message in the diagram

ScenarioDiagram.CreateObject Method

Description

This method creates a new object in a scenario diagram and returns it in the specified object.

Syntax

```
Set theObjectInstance = theDiagram.CreateObject (theName,
      theClassName)
```

Element	Description
theObjectInstance As Object	Returns the newly created object
theDiagram As ScenarioDiagram	ScenarioDiagram to which the new object is being added
theName As String	Name of the object to be created
theClassName As String	Class to which the newly created object belongs

ScenarioDiagram.DeleteInstance Method

Description

This method adds an object instance to a scenario diagram and returns it in the specified object.

Syntax

```
IsDeleted = theScenarioDiagram.DeleteInstance (theInstance)
```

Element	Description
IsDeleted As Boolean	Returns a value of True when the instance is successfully deleted
theScenarioDiagram As ScenarioDiagram	ScenarioDiagram from which the object instance is being deleted
theInstance As ObjectInstance	Instance being deleted from the diagram

ScenarioDiagram.GetDiagramType Method

Description

This method retrieves the value of the scenario diagram type.

Syntax

theType = *theScenarioDiagram*.**GetDiagramType** ()

Element	Description
theType As Integer	Returns the value of the diagram type 1 = Sequence Diagram 2 = Collaboration Diagram
theScenarioDiagram As ScenarioDiagram	Scenario diagram from which to retrieve the diagram type

ScenarioDiagram.GetMessages Method

Description

This method retrieves a collection of messages from a scenario diagram.

Syntax

Set *theMessages* = *theScenarioDiagram*.**GetMessages** ()

Element	Description
theMessages As MessageCollection	Returns the collection of messages from the scenario diagram
theScenarioDiagram As ScenarioDiagram	Scenario diagram from which to retrieve the messages

ScenarioDiagram.GetObjects Method

Description

This method retrieves a collection of objects from a scenario diagram.

Syntax

Set *theObjects* = *theObject*.**GetObjects** ()

Element	**Description**
theObjects As ObjectInstanceCollection	Returns the collection of objects from the scenario diagram
theObject As ScenarioDiagram	Scenario diagram from which to retrieve the objects

ScenarioDiagram.GetSelectedLinks Method

Description

This method retrieves the currently selected links from a scenario diagram.

Syntax

Set *theLinks* = *theScenarioDiagram*.**GetSelectedLinks** ()

Element	**Description**
theLinks As LinkCollection	Returns collection of currently selected links belonging to the specified scenario diagram
theScenarioDiagram As ScenarioDiagram	ScenarioDiagram from which the links are being retrieved

ScenarioDiagram.GetSelectedMessages Method

Description

This method retrieves the collection of currently selected messages from a scenario diagram.

Syntax

```
Set theMessages = theScenarioDiagram.GetSelectedMessages ()
```

Element	Description
theMessages As MessageCollection	Returns the collection of currently selected messages from the scenario diagram
theScenarioDiagram As ScenarioDiagram	Scenario diagram from which to retrieve the messages

ScenarioDiagram.GetSelectedObjects Method

Description

This method retrieves the collection of currently selected objects from a scenario diagram.

Syntax

```
Set theObjects = theDiagram.GetSelectedObjects ()
```

Element	Description
theObjects As ObjectInstanceCollection	Returns the collection of currently selected objects from the scenario diagram
theDiagram As ScenarioDiagram	Scenario diagram from which to retrieve the messages

ScenarioDiagram.RemoveInstanceView Method

Description

This method removes an object instance from a scenario diagram.

Syntax

```
IsRemoved = theScenarioDiagram.RemoveInstanceView
      (theView)
```

Element	Description
IsRemoved As Boolean	Returns a value of True when the instance view is successfully removed
theScenarioDiagram As ScenarioDiagram	ScenarioDiagram from which the instance view is being deleted
theView As InstanceView	Instance view being removed from the diagram

State Class

The State Class specifies properties and methods that control the state information that you can get, set, and track for objects in a model. This state information includes:

- Parent states, substates, state kinds, state machines
- Actions
- Events
- Transitions
- History

State Class Properties

The following table summarizes the State Class properties.

Table 143 State Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
RoseItem Properties	Inherits all RoseItem class properties
StateVertex Properties	Inherits all StateVertex class properties
AbstractState Properties	Inherits all AbstractState class properties
History	Specifies which substate of a superstate to enter upon entering the superstate
StateKind	Rich type that indicates the type of state

State.History Property

Description

Specifies which substate of a superstate to enter upon entering the superstate:

- True indicates that upon return to a superstate, enter the most recently visited substate
- False indicates that upon return to a superstate, always enter the initial substate

Syntax

State.**History**

Property Type

Boolean

State.StateKind Property

Description

The StateKind property is a rich data type. The following table describes the valid forms of expressing the StateKind rich data type for the State class.

Table 144   State.StateKind Rich Data Types

Rich Data Type	Description
StateKind As String	Indicates the type of state. Valid values are: ■ Normal (default) ■ Initial ■ Final

Note: *This property is read-only.*

Syntax

`State`.**`StateKind`**

Property Type

StateKind

State Class Methods

The following table summarizes the State Class methods.

Table 145 State Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
RoseItem Methods	Inherits all RoseItem class methods
StateVertex Methods	Inherits all StateVertex class methods
AbstractState Methods	Inherits all AbstractState class methods
AddState	Adds a state to a state
DeleteState	Deletes a state from the state
RelocateState	Relocates a state to the given state

State.AddState Method

Description

This method adds a substate to the specified state.

Syntax

Set *theSubState* = *theState*.**AddState** (*theName*)

Element	**Description**
theSubState As State	Returns the substate being added to the state; this state becomes a substate of the state to which it is being added
theState As State	The state to which the substate is being added
theName As String	Name of the substate to add

State.DeleteState Method

Description

This method deletes a substate from a state.

Syntax

IsDeleted = *theState*.**DeleteState** (*theSubState*)

Element	Description
IsDeleted As Boolean	Returns a value of True when the state is successfully deleted
theState As State	State from which the substate is being deleted
theSubState as State	Substate being deleted from the state

State.RelocateState Method

Description

This method relocates a substate to another specified state.

Syntax

IsRelocated = *theState*.**RelocateState** (*theRelocatedState*)

Element	Description
IsRelocated As Boolean	Returns a value of True when the state is successfully relocated
theState As State	State to which the substate is being relocated
theRelocatedState As State	Substate being relocated

StateDiagram Class

The StateDiagram class is an abstract class that exposes Rose's state diagram functionality in the extensibility interface. With the properties and methods of the StateDiagram class, you can:

- Retrieve and set information about the state diagram such as name, documentation, zoom factor
- Retrieve objects associated with the state diagram such as application, model, external documents, parent state machine
- Add objects to the diagram such as relation views, activity views, decision views, state views, swimlane views, and synchronization views
- Retrieve all or selected objects from the diagram such as items, item views, activities, activity views, decisions, decision views, states, state views, synchronizations, synchronization views, swimlane views, transitions, note views
- Delete activity views, decision views, state views, swimlane views, synchronization views, note views
- Determine whether the diagram is visible, or active
- Create and retrieve tool and property settings for state diagrams
- Activate, redraw, layout, or update a particular state diagram
- Add and delete external documents
- Determine whether a particular roseitem view exists on the state diagram
- Render the diagram in Windows metafile or enhanced Windows metafile formats to a file or the clipboard

The StateDiagram class corresponds to activity and statechart diagrams in the Rose user interface.

StateDiagram Class Properties

The following table summarizes the StateDiagram Class properties.

Table 146 StateDiagram Class Properties Summary

Property	Description
Element Properties	Inherits all Element Class properties
Diagram Properties	Inherits all Diagram Class properties
IsActivityDiagram	Indicates if the diagram is an activity diagram
Parent	State machine that contains this state diagram

StateDiagram.IsActivityDiagram Property

Description

This property indicates if the diagram is an activity diagram.

Note: *This property is read-only.*

Syntax

```
isActivityDiagram = objStateDiagram.IsActivityDiagram
```

Property Type

Boolean

StateDiagram.Parent Property

Description

Specifies the state machine that contains this state diagram.

Note: *This property is read-only.*

Syntax

```
StateDiagram.Parent
```

Property Type

StateMachine

StateDiagram Class Methods

The following table summarizes the StateDiagram Class methods.

Table 147 StateDiagram Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
Diagram Methods	Inherits all Diagram Class methods
AddActivityView	Adds an activity view
AddDecisionView	Adds a decision view
AddStateView	Adds a state view
AddSwimLaneView	Adds a swimlane view
AddSynchronizationView	Adds a synchronization view
GetActivityViews	Retrieves all activity views corresponding to the specified activity
GetDecisionViews	Retrieves all decision views corresponding to the specified decision
GetDiagramActivityViews	Retrieves all activity views displayed on the state diagram
GetDiagramDecisionViews	Retrieves all decision views displayed on the state diagram
GetDiagramStateVertexViews	Retrieves all state vertex views (activity views, state views, decision views, synchronization views) displayed on the state diagram
GetDiagramSwimLaneViews	Retrieves all swimlane views displayed on the state diagram
GetDiagramSynchronizationViews	Retrieves all synchronization views displayed on the state diagram
GetSelectedActivities	Retrieves the activities represented by the currently selected activity views

Method	Description
GetSelectedActivityViews	Retrieves the currently selected activity views
GetSelectedDecisionViews	Retrieves the currently selected decision views
GetSelectedDecisions	Retrieves the decisions represented by the currently selected decision views
GetSelectedStates	Retrieves the currently selected states
GetSelectedStateVertices	Retrieves the state vertices (activities, states, decisions, synchronizations) represented by the currently selected state vertex views
GetSelectedStateViews	Retrieves the currently selected state views
GetSelectedSwimLaneViews	Retrieves the currently selected swimlane views
GetSelectedSynchronizations	Retrieves the synchronizations represented by the currently selected synchronization views
GetSelectedSynchronizationViews	Retrieves the currently selected synchronization views
GetSelectedTransitions	Retrieves the currently selected transitions
GetStateVertexViews	Retrieves all state vertex views corresponding to a specified state vertex (activity, state, decision, synchronization)
GetStateView	Retrieves a given state's state view
GetStateViews	Retrieves all state views
GetSwimLaneViews	Retrieves all swimlane views corresponding to the specified swimlane

Method	Description
GetSynchronizationViews	Retrieves all synchronization views corresponding to the specified synchronization
RemoveActivityView	Removes the activity view
RemoveDecisionView	Removes the decision view
RemoveStateView	Removes the state view
RemoveSwimLaneView	Removes the swimlane view
RemoveSynchronizationView	Removes the synchronization view

StateDiagram.AddActivityView Method

Description

This method adds an activity view, corresponding to the specified activity, to the state diagram.

Syntax

```
Set theActivityView = myStateDiagram.AddActivityView
       (theActivity)
```

Element	Description
theActivityView As ActivityView	Returns the activity view being added to the state diagram
myStateDiagram As StateDiagram	The state diagram to which to add the activity view
theActivity As Activity	The activity to be represented by the newly added activity view

StateDiagram.AddDecisionView Method

Description

This method adds a decision view, corresponding to the specified decision, to the state diagram.

Syntax

```
Set theDecisionView = myStateDiagram.AddDecisionView
      (theDecision)
```

Element	Description
theDecisionView As DecisionView	Returns the decision view being added to the state diagram
myStateDiagram As StateDiagram	The state diagram to which to add the decision view
theDecision As Decision	The decision to be represented by the newly added decision view

StateDiagram.AddStateView Method

Description

This method adds a state view object to a state diagram.

Syntax

```
Set theStateView = theStateDiagram.AddStateView (theState)
```

Element	Description
theStateView As StateView	Returns the state view being added to the state diagram
theStateDiagram As StateDiagram	The state diagram to which the state view is being added
theState As State	The state represented by the state view object

StateDiagram.AddSwimLaneView Method

Description

This method adds a swimlane view, corresponding to the specified swimlane, to the state diagram.

Syntax

```
Set theSwimLaneView = myStateDiagram.AddSwimLaneView
       (theSwimLane)
```

Element	Description
theSwimLaneView As SwimLaneView	Returns the swimlane view being added to the state diagram
myStateDiagram As StateDiagram	The state diagram to which to add the swimlane view
theSwimLane As SwimLane	The swimlane to be represented by the newly added swimlane view

StateDiagram.AddSynchronizationView Method

Description

This method adds a synchronization view, corresponding to the specified synchronization, to the state diagram.

Syntax

```
Set theSyncItemView = myStateDiagram.AddSynchronizationView
      (theSyncItem, isHorizontal)
```

Element	Description
theSyncItemView As SyncItemView	Returns the synchronization view being added to the state diagram
myStateDiagram As StateDiagram	The state diagram to which to add the synchronization view
theSyncItem As SyncItem	The synchronization to be represented by the newly added synchronization view
isHorizontal As Boolean	Indicator for whether the synchronization view is horizontal Set this to True to display the synchronization view horizontally on the state diagram Set this to False to display the synchronization view vertically on the state diagram

StateDiagram.GetActivityViews Method

Description

This method retrieves all activity views, corresponding to the specified activity, from the state diagram. To retrieve all activity views from the state diagram, regardless of activity, use the StateDiagram.GetDiagramActivityViews method.

Syntax

```
Set theActivityViewCollection =
      myStateDiagram.GetActivityViews (theActivity)
```

Element	Description
theActivityViewCollection As ActivityViewCollection	Returns the collection of activity views, corresponding to the specified activity, from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the activity views
theActivity As Activity	Activity for which to retrieve activity views from the state diagram

StateDiagram.GetDecisionViews Method

Description

This method retrieves all decision views, corresponding to the specified decision, from the state diagram. To retrieve all decision views from the state diagram, regardless of decision, use the StateDiagram.GetDiagramDecisionViews method.

Syntax

```
Set theDecisionViewCollection =
      myStateDiagram.GetDecisionViews (theDecision)
```

Element	Description
theDecisionViewCollection As DecisionViewCollection	Returns the collection of decision views, corresponding to the specified decision, from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the decision views
theDecision As Decision	Decision for which to retrieve decision views from the state diagram

StateDiagram.GetDiagramActivityViews Method

Description

This method retrieves all activity views from the state diagram, regardless of activity. To retrieve all activity views from the state diagram for a particular activity, use the StateDiagram.GetActivityViews method.

Syntax

```
Set theActivityViewCollection =
      myStateDiagram.GetDiagramActivityViews ()
```

Element	Description
theActivityViewCollection As ActivityViewCollection	Returns the collection of all activity views from the state diagram, regardless of activity
myStateDiagram As StateDiagram	State diagram from which to retrieve the decision views

StateDiagram.GetDiagramDecisionViews Method

Description

This method retrieves all decision views from the state diagram, regardless of decision. To retrieve all decision views from the state diagram for a particular decision, use the StateDiagram.GetDecisionViews method.

Syntax

```
Set theDecisionViewCollection =
      myStateDiagram.GetDiagramDecisionViews ()
```

Element	Description
theDecisionViewCollection As DecisionViewCollection	Returns the collection of all decision views from the state diagram, regardless of decision
myStateDiagram As StateDiagram	State diagram from which to retrieve the decision views

StateDiagram.GetDiagramStateVertexViews Method

Description

This method retrieves all state vertex views from the state diagram, regardless of state vertex (activity, state, decision, synchronization). To retrieve all state vertex views from the state diagram for a particular state vertex, use the StateDiagram.GetStateVertexViews method.

Syntax

```
Set theItemViewCollection =
      myStateDiagram.GetDiagramStateVertexViews ()
```

Element	Description
theItemViewCollection As ItemViewCollection	Returns the collection of all state vertex views from the state diagram, regardless of state vertex
myStateDiagram As StateDiagram	State diagram from which to retrieve the state vertex views

StateDiagram.GetDiagramSwimLaneViews Method

Description

This method retrieves all swimlane views from the state diagram, regardless of swimlane. To retrieve all swimlane views from the state diagram for a particular swimlane, use the StateDiagram.GetSwimLaneViews method.

Syntax

```
Set theSwimLaneViewCollection =
      myStateDiagram.GetDiagramSwimLaneViews ()
```

Element	Description
theSwimLaneViewCollection As SwimLaneViewCollection	Returns the collection of all swimlane views from the state diagram, regardless of swimlane
myStateDiagram As StateDiagram	State diagram from which to retrieve the swimlane views

StateDiagram.GetDiagramSynchronizationViews Method

Description

This method retrieves all synchronization views from the state diagram, regardless of synchronization. To retrieve all synchronization views from the state diagram for a particular synchronization, use the StateDiagram.GetSynchronizationViews method.

Syntax

```
Set theSyncItemViewCollection =
      myStateDiagram.GetDiagramSynchronizationViews ()
```

Element	Description
theSyncItemViewCollection As SyncItemViewCollection	Returns the collection of all synchronization views from the state diagram, regardless of synchronization
myStateDiagram As StateDiagram	State diagram from which to retrieve the synchronization views

StateDiagram.GetSelectedActivities Method

Description

This method retrieves the activities represented by the currently selected activity views from the state diagram.

Syntax

```
Set theActivityCollection =
      myStateDiagram.GetSelectedActivities ()
```

Element	Description
theActivityCollection As ActivityCollection	Returns the collection of all activities represented by the currently selected activity views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the activities

StateDiagram.GetSelectedActivityViews Method

Description

This method retrieves the currently selected activity views from the state diagram.

Syntax

```
Set theActivityViewCollection =
      myStateDiagram.GetSelectedActivityViews ()
```

Element	Description
theActivityViewCollection As ActivityViewCollection	Returns the collection of all currently selected activity views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the selected activity views

StateDiagram.GetSelectedDecisionViews Method

Description

This method retrieves the currently selected decision views from the state diagram.

Syntax

```
Set theDecisionViewCollection =
      myStateDiagram.GetSelectedDecisionViews ()
```

Element	Description
theDecisionViewCollection As DecisionViewCollection	Returns the collection of all currently selected decision views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the selected decision views

StateDiagram.GetSelectedDecisions Method

Description

This method retrieves the decisions represented by the currently selected decision views from the state diagram.

Syntax

```
Set theDecisionCollection =
      myStateDiagram.GetSelectedDecisions ()
```

Element	Description
theDecisionCollection As DecisionCollection	Returns the collection of all decisions represented by the currently selected decision views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the decisions

StateDiagram.GetSelectedStates Method

Description

This method retrieves a collection containing the states that are currently selected in the specified state diagram.

Syntax

```
Set theStates = theStateDiagram.GetSelectedStates ()
```

Element	Description
theStates As StateCollection	Returns the collection containing the currently selected states
theStateDiagram As StateDiagram	State diagram from which the state collection is being retrieved

StateDiagram.GetSelectedStateVertices Method

Description

This method retrieves the state vertices (activities, states, decisions, synchronizations) represented by the currently selected state vertex views from the state diagram.

Syntax

```
Set theStateVertexCollection =
      myStateDiagram.GetSelectedStateVertices ()
```

Element	Description
theStateVertexCollection As StateVertexCollection	Returns the collection of all state vertices (activities, states, decisions, synchronizations) represented by the currently selected state vertex views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the state vertices

StateDiagram.GetSelectedStateViews Method

Description

This method retrieves a collection containing the state view objects that are currently selected in the specified state diagram.

Syntax

```
Set theStateViews = theStateDiagram.GetSelectedStateViews()
```

Element	Description
theStateViews As StateViewCollection	Returns the collection containing the currently selected state view objects
theStateDiagram As StateDiagram	State diagram from which the state view collection is being retrieved

StateDiagram.GetSelectedSwimLaneViews Method

Description

This method retrieves the currently selected swimlane views from the state diagram.

Syntax

```
Set theSwimLaneViewCollection =
      myStateDiagram.GetSelectedSwimLaneViews ()
```

Element	Description
theSwimLaneViewCollection As SwimLaneViewCollection	Returns the collection of all currently selected swimlane views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the selected swimlane views

StateDiagram.GetSelectedSynchronizations Method

Description

This method retrieves the synchronizations represented by the currently selected synchronization views from the state diagram.

Syntax

```
Set theSyncItemCollection =
      myStateDiagram.GetSelectedSynchronizations ()
```

Element	Description
theSyncItemCollection As SyncItemCollection	Returns the collection of all synchronizations represented by the currently selected synchronization views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the synchronizations

StateDiagram.GetSelectedSynchronizationViews Method

Description

This method retrieves the currently selected synchronization views from the state diagram.

Syntax

```
Set theSyncItemViewCollection =
      myStateDiagram.GetSelectedSynchronizationViews ()
```

Element	Description
theSyncItemViewCollection As SyncItemViewCollection	Returns the collection of all currently selected synchronization views from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the selected synchronization views

StateDiagram.GetSelectedTransitions Method

Description

This method retrieves a collection containing the transitions that are currently selected in the specified state diagram.

Syntax

```
Set theTransitions = theStateDiagram.GetSelectedTransitions
       ()
```

Element	Description
theTransitions As TransitionCollection	Returns the collection containing the currently selected transitions
theStateDiagram As StateDiagram	State diagram from which the transition collection is being retrieved

StateDiagram.GetStateVertexViews Method

Description

This method retrieves all state vertex views, corresponding to the specified state vertex (activity, state, decision, synchronization), from the state diagram. To retrieve all state vertex views from the state diagram, regardless of state vertex, use the StateDiagram.GetDiagramStateVertexViews method.

Syntax

```
Set theItemViewCollection =
      myStateDiagram.GetStateVertexViews (theStateVertex)
```

Element	Description
theItemViewCollection As ItemViewCollection	Returns the collection of state vertex views, corresponding to the specified state vertex (activity, state, decision, synchronization), from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the state vertex views
theStateVertex As StateVertex	State vertex for which to retrieve state vertex views from the state diagram

StateDiagram.GetStateView Method

Description

This method retrieves a state view from a state diagram.

Syntax

`Set` *`theStateView`* = *`theStateDiagram`*.**`GetStateView`** (*`theState`*)

Element	Description
theStateView As StateView	Returns the state view that represents the specified state
theStateDiagram As StateDiagram	State diagram from which the state view is being retrieved
theState As State	State whose state view object is being retrieved

StateDiagram.GetStateViews Method

Description

This method retrieves a collection containing all state view objects belonging to the specified state diagram.

Syntax

`Set` *`theStateViews`* = *`theStateDiagram`*.**`GetStateViews`** ()

Element	Description
theStateViews As StateViewCollection	Returns the collection containing the state view objects belonging to the specified state diagram
theStateDiagram As StateDiagram	State diagram whose state views are being retrieved

StateDiagram.GetSwimLaneViews Method

Description

This method retrieves all swimlane views, corresponding to the specified swimlane, from the state diagram. To retrieve all swimlane views from the state diagram, regardless of swimlane, use the StateDiagram.GetDiagramSwimLaneViews method.

Syntax

```
Set theSwimLaneViewCollection =
      myStateDiagram.GetSwimLaneViews (theSwimLane)
```

Element	**Description**
theSwimLaneViewCollection As SwimLaneViewCollection	Returns the collection of swimlane views, corresponding to the specified swimlane, from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the swimlane views
theSwimLane As SwimLane	State vertex for which to retrieve swimlane views from the state diagram

StateDiagram.GetSynchronizationViews Method

Description

This method retrieves all synchronization views, corresponding to the specified synchronization, from the state diagram. To retrieve all synchronization views from the state diagram, regardless of synchronization, use the StateDiagram.GetDiagramSynchronizationViews method.

Syntax

```
Set theSyncItemViewCollection =
      myStateDiagram.GetSynchronizationViews (theSyncItem)
```

Element	Description
theSyncItemViewCollection As SyncItemViewCollection	Returns the collection of synchronization views, corresponding to the specified synchronization, from the state diagram
myStateDiagram As StateDiagram	State diagram from which to retrieve the synchronization views
theSyncItem As SyncItem	State vertex for which to retrieve synchronization views from the state diagram

StateDiagram.RemoveActivityView Method

Description

This method deletes the activity view from the state diagram. Note that this method does not remove the activity from the model. To delete the activity from the model, use the StateMachine.DeleteStateVertex method.

Syntax

```
isActivityViewRemoved = myStateDiagram.RemoveActivityView
      (theActivityView)
```

Element	Description
isActivityViewRemoved As Boolean	Returns True if the activity view is successfully deleted from the state diagram
myStateDiagram As StateDiagram	State diagram from which to delete the activity view
theActivityView As ActivityView	Activity view to be deleted from the state diagram

StateDiagram.RemoveDecisionView Method

Description

This method deletes the decision view from the state diagram. Note that this method does not remove the decision from the model. To delete the decision from the model, use the StateMachine.DeleteStateVertex method.

Syntax

```
isDecisionViewRemoved = myStateDiagram.RemoveDecisionView
       (theDecisionView)
```

Element	**Description**
isDecisionViewRemoved As Boolean	Returns True if the decision view is successfully deleted from the state diagram
myStateDiagram As StateDiagram	State diagram from which to delete the decision view
theDecisionView As DecisionView	Decision view to be deleted from the state diagram

StateDiagram.RemoveStateView Method

Description

This method removes a state view object from a state diagram.

Syntax

IsRemoved = *theStateDiagram*.**RemoveStateView** (*theStateView*)

Element	Description
IsRemoved As Boolean	Returns a value of True when the state view is successfully removed from the diagram
theStateDiagram As StateDiagram	State diagram from which to remove the state view
theStateView As StateView	State view being removed from the diagram

StateDiagram.RemoveSwimLaneView Method

Description

This method deletes the swimlane view from the state diagram. Note that this method does not remove the swimlane from the model. To delete the swimlane from the model, use the StateMachine.DeleteSwimLane method.

Syntax

```
isSwimLaneViewRemoved = myStateDiagram.RemoveSwimLaneView
        (theSwimLaneView)
```

Element	Description
isSwimLaneViewRemoved As Boolean	Returns True if the swimlane view is successfully deleted from the state diagram
myStateDiagram As StateDiagram	State diagram from which to delete the swimlane view
theSwimLaneView As SwimLaneView	Swimlane view to be deleted from the state diagram

StateDiagram.RemoveSynchronizationView Method

Description

This method deletes the synchronization view from the state diagram. Note that this method does not remove the synchronization from the model. To delete the synchronization from the model, use the StateMachine.DeleteStateVertex method.

Syntax

```
isSynchronizationViewRemoved =
      myStateDiagram.RemoveSynchronizationView
      (theSyncItemView)
```

Element	Description
isSynchronizationViewRemoved As Boolean	Returns True if the synchronization view is successfully deleted from the state diagram
myStateDiagram As StateDiagram	State diagram from which to delete the synchronization view
theSyncItemView As SyncItemView	Synchronization view to be deleted from the state diagram

StateMachine Class

The StateMachine class is an abstract class that exposes Rose's state machine functionality in the extensibility interface. With the properties and methods of the StateMachine class, you can:

- Retrieve information about state machines, such as name, documentation, and stereotypes
- Retrieve objects associated with a particular state machine, including application, model, parent class, state machine owner, and transitions
- Create and retrieve tool and property settings for state machines
- Open specification sheets for state machines
- Add, delete, and retrieve activities, decisions, external documents, states, and synchronizations

- Add and retrieve activity diagrams and statechart diagrams
- Add and delete swimlanes
- Relocate states

A class can have zero or one state machine; a state machine can belong to only one class.

The StateMachine class corresponds to state/activity models in the Rose user interface.

StateMachine Class Properties

The following table summarizes the StateMachine class properties.

Table 148 StateMachine Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
AbstractStates	Specifies the collection of abstract states (activities, states) belonging to the state machine
Activities	Specifies the collection of activities belonging to the state machine
Decisions	Specifies the collection of decisions belonging to the state machine
Diagrams	Specifies the collection of state diagrams (statechart, activity) belonging to the state machine
Documentation	Specifies the state machine's documentation
ExternalDocuments	Specifies the collection of external documents belonging to the state machine
LocalizedStereotype	Specifies the state machine's localized stereotype
ParentClass	The class to which the state machine belongs
StateMachineOwner	Specifies the state machine owner (model element) to which the state machine belongs
States	The collection of states belonging to the state machine

Property	Description
StateVertices	Specifies the collection of state vertices (activities, states, decisions, synchronizations) belonging to the state machine
Stereotype	Specifies the state machine's stereotype
SwimLanes	Specifies the collection of swimlanes belonging to the state machine
Synchronizations	Specifies the collection of synchronizations belonging to the state machine

StateMachine.AbstractStates Property

Description

This property specifies the collection of abstract states (activities, states) belonging to the state machine.

Note: *This property is read-only.*

Syntax

```
Set myAbstractStateCollection =
    myStateMachine.AbstractStates
```

Property Type

AbstractStateCollection

StateMachine.Activities Property

Description

This property specifies the collection of activities belonging to the state machine.

Note: *This property is read-only.*

Syntax

`Set` *`myActivityCollection`* `=` *`myStateMachine`*`.`**`Activities`**

Property Type

ActivityCollection

StateMachine.Decisions Property

Description

This property specifies the collection of decisions belonging to the state machine.

Note: *This property is read-only.*

Syntax

`Set` *`myDecisionCollection`* `=` *`myStateMachine`*`.`**`Decisions`**

Property Type

DecisionCollection

StateMachine.Diagrams Property

Note: *This method replaces the StateMachine class method, Diagram, used in Rational Rose version 98i.*

Description

This property specifies the collection of state diagrams (statechart, activity) belonging to the state machine.

Note: *This property is read-only.*

Syntax

```
Set myStateDiagramCollection = myStateMachine.Diagrams
```

Property Type

StateDiagramCollection

StateMachine.Documentation Property

Description

This property specifies the state machine’s documentation.

Syntax

```
theDocumentation = myStateMachine.Documentation
```

```
myStateMachine.Documentation = "Some text about the state
      machine that displays in the documentation section
      of the state machine specification."
```

Property Type

String

StateMachine.ExternalDocuments Property

Description

This property specifies the collection of external documents belonging to the state machine.

Note: *This property is read-only.*

Syntax

```
Set myExternalDocumentCollection =
      myStateMachine.ExternalDocuments
```

Property Type

ExternalDocumentCollection

StateMachine.LocalizedStereotype Property

Description

This property specifies the state machine’s localized stereotype.

Syntax

```
theLocalizedStereotype =
      myStateMachine.LocalizedStereotype
```

```
myStateMachine.LocalizedStereotype = "Some text indicating
      the localized version of the stereotype"
```

Property Type

String

StateMachine.ParentClass Property

Description

Specifies the class to which the state machine belongs. A state machine can belong to only one class.

Note: *This property is read-only.*

Syntax

StateMachine.**ParentClass**

Property Type

Class

StateMachine.StateMachineOwner Property

Description

This property specifies the state machine owner (model element) to which the state machine belongs.

Note: *This property is read-only.*

Syntax

Set *myStateMachineOwner* = *myStateMachine*.**StateMachineOwner**

Property Type

StateMachineOwner

StateMachine.States Property

Description

Specifies the collection of states belonging to the state machine.

Note: *This property is read-only.*

Syntax

```
StateMachine.States
```

Property Type

StateCollection

StateMachine.StateVertices Property

Description

This property specifies the collection of state vertices (activities, states, decisions, synchronizations) belonging to the state machine.

Note: *This property is read-only.*

Syntax

```
Set myStateVertexCollection = myStateMachine.StateVertices
```

Property Type

StateVertexCollection

StateMachine.Stereotype Property

Description

This property specifies the state machine's stereotype.

Syntax

theStereotype = *myStateMachine*.**Stereotype**

myStateMachine.**Stereotype** = "Interface"

Property Type

String

StateMachine.SwimLanes Property

Description

This property specifies the collection of swimlanes belonging to the state machine.

Note: *This property is read-only.*

Syntax

Set *mySwimLaneCollection* = *myStateMachine*.**SwimLanes**

Property Type

SwimLaneCollection

StateMachine.Synchronizations Property

Description

This property specifies the collection of synchronizations belonging to the state machine.

Note: *This property is read-only.*

Syntax

`Set` *`mySyncItemCollection`* = *`myStateMachine`*.**`Synchronizations`**

Property Type

SyncItemCollection

StateMachine Class Methods

The following table summarizes the StateMachine Class methods.

Table 149 StateMachine Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class Methods
AddActivity	Adds an activity to the state machine
AddActivityDiagram	Adds an activity diagram to the state machine
AddDecision	Adds a decision to the state machine
AddExternalDocument	Adds an external document to the state machine
AddState	Adds a state to the state machine
AddStateChartDiagram	Adds a statechart diagram to the state machine
AddSwimLane	Adds a swimlane to the state machine
AddSynchronization	Adds a synchronization to the state machine
DeleteExternalDocument	Deletes an external document from the state machine
DeleteState	Deletes a state from the state machine
DeleteStateVertex	Deletes a state vertex (activity, state, decision, synchronization) from the state machine
DeleteSwimLane	Deletes a swimlane from the state machine
GetAllAbstractStates	Retrieves all abstract states (activities, states) from the state machine and all of its nested state machines
GetAllActivities	Retrieves all activities from the state machine and all of its nested state machines
GetAllDecisions	Retrieves all decisions from the state machine and all of its nested state machines

Method	Description
GetAllDiagrams	Retrieves all diagrams (activity, statechart) from the state machine and its nested state machines and all of its nested state machines
GetAllStates	Retrieves all states belonging to the state machine and all of its nested state machines
GetAllStateVertices	Retrieves all state vertices (activities, states, decisions, synchronizations) from the state machine and all of its nested state machines
GetAllSynchronizations	Retrieves all synchronizations from the state machine and all of its nested state machines
GetAllTransitions	Retrieves all transitions belonging to the state machine and all of its nested state machines
GetTransitions	Retrieves all first-level transitions belonging to the state machine
OpenCustomSpecification	If a custom specification exists, opens the custom specification window for the state machine
OpenSpecification	Opens the Rational Rose default specification window for the state machine
RelocateState	Relocates a state in the state machine

StateMachine.AddActivity Method

Description

This method creates a new activity and adds it to the state machine.

Syntax

Set *theActivity* = *myStateMachine*.**AddActivity** (*theName*)

Element	Description
theActivity As Activity	Returns the newly created activity and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the activity
theName As String	Name to give to the newly created activity

StateMachine.AddActivityDiagram Method

Description

This method creates a new activity diagram and adds it to the state machine.

Syntax

Set *theStateDiagram* = *myStateMachine*.**AddActivityDiagram** (*theName*)

Element	Description
theStateDiagram As StateDiagram	Returns the newly created activity diagram and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the activity diagram
theName As String	Name to give to the newly created activity diagram

StateMachine.AddDecision Method

Description

This method creates a new decision and adds it to the state machine.

Syntax

`Set` *`theDecision`* `=` *`myStateMachine`*`.`**`AddDecision`** `(`*`theName`*`)`

Element	Description
theDecision As Decision	Returns the newly created decision and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the decision
theName As String	Name to give to the newly created decision

StateMachine.AddExternalDocument Method

Description

This method adds an external document to the state machine.

Syntax

```
Set theExternalDocument =
      myStateMachine.AddExternalDocument (docName, docType)
```

Element	Description
theExternalDocument As ExternalDocument	Returns the external document and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the external document
docName As String	Fully qualified name of the external document For a file, this must include the entire path name (for example, “C:\My Documents\My File Name”). For a URL, this must include the entire address (for example, “http://myCompany.com/myHomePage/mySubPage/”).
docType As Integer	Indicates the type of external document: 1 = file 2 = URL

StateMachine.AddState Method

Description

This method creates a new state and adds it to the state machine.

Syntax

```
Set theState = theStateMachine.AddState (theName)
```

Element	Description
theState As State	Returns the state being added to the state
theStateMachine As State	The state machine to which the state is being added
theName As String	Name of the state to add

StateMachine.AddStateChartDiagram Method

Description

This method creates a new statechart diagram and adds it to the state machine.

Syntax

```
Set theStateDiagram = myStateMachine.AddStateChartDiagram
       (theName)
```

Element	Description
theStateDiagram As StateDiagram	Returns the newly created statechart diagram and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the statechart diagram
theName As String	Name to give to the newly created statechart diagram

StateMachine.AddSwimLane Method

Description

This method creates a new swimlane and adds it to the state machine.

Syntax

```
Set theSwimLane = myStateMachine.AddSwimLane (theName)
```

Element	Description
theSwimLane As SwimLane	Returns the newly created swimlane and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the swimlane
theName As String	Name to give to the newly created swimlane

StateMachine.AddSynchronization Method

Description

This method creates a new synchronization and adds it to the state machine.

Syntax

```
Set theSyncItem = myStateMachine.AddSynchronization
        (theName)
```

Element	Description
theSyncItem As SyncItem	Returns the newly created synchronization and adds it to the state machine
myStateMachine As StateMachine	State machine to which to add the synchronization
theName As String	Name to give to the newly created synchronization

StateMachine.DeleteExternalDocument Method

Description

This method deletes an external document from the state machine.

Syntax

```
isExternalDocumentDeleted =
      myStateMachine.DeleteExternalDocument
      (theExternalDocument)
```

Element	Description
isExternalDocumentDeleted As Boolean	Returns a value of True, if the external document is successfully deleted from the state machine
myStateMachine As StateMachine	State machine from which to delete the external document
theExternalDocument As ExternalDocument	External Document to delete from the state machine

StateMachine.DeleteState Method

Description

This method deletes a state from the state machine.

Syntax

```
IsDeleted = theStateMachine.DeleteState (theState)
```

Element	Description
IsDeleted As Boolean	Returns a value of True when the state is successfully deleted
theStateMachine As StateMachine	State machine from which the state is being deleted
theState as State	State being deleted from the state machine

StateMachine.DeleteStateVertex Method

Description

This method deletes a state vertex (activity, state, decision, synchronization) from the state machine.

Syntax

```
isStateVertexDeleted = myStateMachine.DeleteStateVertex
      (theStateVertex)
```

Element	Description
isStateVertexDeleted As Boolean	Returns a value of True, if the state vertex (activity, state, decision, synchronization) is successfully deleted from the state machine
myStateMachine As StateMachine	State machine from which to delete the state vertex
theStateVertex As StateVertex	State vertex to delete from the state machine

StateMachine.DeleteSwimLane Method

Description

This method deletes a swimlane from the state machine.

Syntax

```
isSwimLaneDeleted = myStateMachine.DeleteSwimLane
      (theSwimLane)
```

Element	Description
isSwimLaneDeleted As Boolean	Returns a value of True, if the swimlane is successfully deleted from the state machine
myStateMachine As StateMachine	State machine from which to delete the swimlane
theSwimLane As SwimLane	Swimlane to delete from the state machine

StateMachine.GetAllAbstractStates Method

Description

This method retrieves all abstract states (activities, states) from the state machine and all of its nested state machines.

Syntax

```
Set theAbstractStateCollection =
      myStateMachine.GetAllAbstractStates ()
```

Element	Description
theAbstractStateCollection As AbstractStateCollection	Returns a collection of all abstract states (activities, states) belonging to the state machine and all of its nested state machines
myStateMachine As StateMachine	State machine from which to retrieve the abstract states

StateMachine.GetAllActivities Method

Description

This method retrieves all activities from the state machine and all of its nested state machines.

Syntax

```
Set theActivityCollection = myStateMachine.GetAllActivities
      ()
```

Element	Description
theActivityCollection As ActivityCollection	Returns a collection of all activities belonging to the state machine and all of its nested state machines
myStateMachine As StateMachine	State machine from which to retrieve the activities

StateMachine.GetAllDecisions Method

Description

This method retrieves all decisions from the state machine and all of its nested state machines.

Syntax

```
Set theDecisionCollection = myStateMachine.GetAllDecisions
       ()
```

Element	Description
theDecisionCollection As DecisionCollection	Returns a collection of all decisions belonging to the state machine and all of its nested state machines
myStateMachine As StateMachine	State machine from which to retrieve the decisions

StateMachine.GetAllDiagrams Method

Description

This method retrieves all diagrams (activity, statechart) from the state machine and all of its nested state machines.

Syntax

```
Set theStateDiagramCollection =
       myStateMachine.GetAllDiagrams ()
```

Element	Description
theStateDiagramCollection As StateDiagramCollection	Returns a collection of all diagrams (activity, statechart) belonging to the state machine and all of its nested state machines
myStateMachine As StateMachine	State machine from which to retrieve the diagrams

StateMachine.GetAllStates Method

Description

This method retrieves the states belonging to the state machine and all of its nested machines

Syntax

```
Set theStates = theStateMachine.GetAllStates ()
```

Element	Description
theStates As StateCollection	Returns a state collection that contains the states belonging to the given state machine and all if its nested state machines
theStateMachine As State Machine	State machine whose states are being retrieved

StateMachine.GetAllStateVertices Method

Description

This method retrieves all state vertices (activities, states, decisions, synchronizations) from the state machine and all of its nested state machines.

Syntax

```
Set theStateVertexCollection =
      myStateMachine.GetAllStateVertices ()
```

Element	Description
theStateVertexCollection As StateVertexCollection	Returns a collection of all state vertices (activities, states, decisions, synchronizations) belonging to the state machine and all of its nested state machines
myStateMachine As StateMachine	State machine from which to retrieve the state vertices

StateMachine.GetAllSynchronizations Method

Description

This method retrieves all synchronizations from the state machine and all of its nested state machines.

Syntax

```
Set theSyncItemCollection =
      myStateMachine.GetAllSynchronizations ()
```

Element	Description
theSyncItemCollection As SyncItemCollection	Returns a collection of all synchronizations belonging to the state machine and all of its nested state machines
myStateMachine As StateMachine	State machine from which to retrieve the synchronizations

StateMachine.GetAllTransitions Method

Description

This method retrieves all transitions belonging to the state machine and all of its nested state machines. To retrieve only first-level transitions belonging to the state machine, use the StateMachine.GetTransitions method.

Syntax

```
Set theTransitions = theStateMachine.GetAllTransitions ()
```

Element	Description
theTransitions As TransitionCollection	Returns a transition collection that contains all transitions belonging to the given state machine and all of its nested state machines
theStateMachine As StateMachine	State machine whose transitions are being retrieved

StateMachine.GetTransitions Method

Description

This method retrieves all first-level transitions belonging to the state machine. To retrieve all transitions belonging to the state machine and all of its nested state machines, use the StateMachine.GetAllTranistions method.

Syntax

```
Set theTransitions = theStateMachine.GetTransitions ()
```

Element	Description
theTransitions As TransitionCollection	Returns a transition collection that contains the first-level transitions belonging to the given state machine
theStateMachine As StateMachine	State machine whose transitions are being retrieved

StateMachine.OpenCustomSpecification Method

Description

If you have defined a custom specification window in your add-in, this method opens your add-in's custom specification window for the specified state machine. To open the Rational Rose default specification window for a state machine, use the StateMachine.OpenSpecification method.

Syntax

```
isCustomSpecificationOpen =
      myStateMachine.OpenCustomSpecification ()
```

Element	Description
isCustomSpecificationOpen As Boolean	Returns a value of True if the custom specification for the state machine is successfully opened
myStateMachine As StateMachine	State machine for which to open the custom specification

StateMachine.OpenSpecification Method

Description

This method opens the Rational Rose default specification window for the specified state machine. To open your add-in's custom specification window, use the StateMachine.OpenCustomSpecification method.

Syntax

```
isSpecificationOpen = myStateMachine.OpenSpecification ()
```

Element	Description
isSpecificationOpen As Boolean	Returns a value of True if the Rational Rose default specification for the state machine is successfully opened
myStateMachine As StateMachine	State machine for which to open the default specification

StateMachine.RelocateState Method

Description

This method relocates a state to the state machine. If the state was a substate, it becomes a top level state.

Syntax

```
IsRelocated = theStateMachine.RelocateState
       (theRelocatedState)
```

Element	Description
IsRelocated As Boolean	Returns a value of True when the state is successfully relocated
theStateMachine As StateMachine	State Machine to which the state is being relocated
theRelocatedState As State	State being relocated

StateMachineOwner Class

The StateMachineOwner class is an abstract class that exposes Rose's state machine owner functionality of model elements in the extensibility interface. With the properties and methods of the StateMachineOwner class, you can:

- Create, delete, and retrieve state machines
- Retrieve information about state machine owners, such as name, parent application, and parent model
- Create and retrieve tool and property settings for state machine owners

The StateMachineOwner class does not directly correspond to anything in the Rose user interface.

StateMachineOwner Class Properties

The following table describes the StateMachineOwner Class properties.

Table 150 StateMachineOwner Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
StateMachines	Specifies the collection of state machines belonging to the state machine owner

StateMachineOwner.StateMachines Property

Description

This property specifies the collection of all state machines belonging to a state machine owner (model element).

Note: *This property is read-only.*

Syntax

```
Set myStateMachineCollection =
      myStateMachineOwner.StateMachines
```

Property Type

StateMachineCollection

StateMachineOwner Class Methods

The following table describes the StateMachineOwner Class methods.

Table 151 StateMachineOwner Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
CreateStateMachine	Creates a state machine
DeleteStateMachine	Deletes a state machine
GetParentItem	Retrieves the parent Rose item of a state machine owner

StateMachineOwner.CreateStateMachine Method

Description

This method creates a state machine and adds it to the specified state machine owner (model element). For another way to add a state machine, see the AbstractState.AddStateMachine method.

Syntax

```
Set theStateMachine =
      myStateMachineOwner.CreateStateMachine
      (theStateMachineName)
```

Element	Description
theStateMachine As StateMachine	Returns the state machine that has been created and added to the specified state machine owner (model element)
myStateMachineOwner As StateMachineOwner	State machine owner (model element) to which to add the newly created state machine
theStateMachineName As String	Name to be given to the newly created state machine

StateMachineOwner.DeleteStateMachine Method

Description

This method deletes a state machine from a state machine owner (model element). For other ways to delete a state machine, see the Class.DeleteStateMachine and AbstractState.DeleteStateMachine methods.

Syntax

```
isStateMachineDeleted =
      myStateMachineOwner.DeleteStateMachine
      (theStateMachine)
```

Element	Description
isStateMachineDeleted As Boolean	Returns a value of True if the state machine is successfully deleted
myStateMachineOwner As StateMachineOwner	State machine owner (model element) from which to delete the state machine
theStateMachine As StateMachine	State machine to be deleted

StateMachineOwner.GetParentItem Method

Description

This method retrieves the parent Rose item of a state machine owner.

Syntax

```
Set objRoseItem = objStateMachineOwner.GetParentItem ()
```

Element	Description
objRoseItem As RoseItem	Returns the parent Rose item
objStateMachineOwner As StateMachineOwner	State machine owner for which to retrieve the parent Rose item

StateVertex Class

The StateVertex class is an abstract class that exposes Rose's state vertex functionality (activities, states, decisions, synchronizations) in the extensibility interface. With the properties and methods of the StateVertex class, you can:

- Retrieve the objects associated with state vertices, such as the parent state vertex, state machine of the parent state vertex, outgoing transitions, and associated swimlanes
- Add and delete transitions
- Retrieve information about the state vertex such as name, parent application, and parent model, documentation, stereotypes, and external documents
- Create and retrieve tool and property settings for state vertices
- Add and delete external documents
- Open specification sheets for state vertices

The StateVertex class does not directly correspond to anything in the Rose user interface. Through inheritance, however, activities, states, decisions, and synchronizations are state vertices.

StateVertex Class Properties

The following table describes the StateVertex Class properties.

Table 152 StateVertex Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
Parent	Specifies the parent state vertex
ParentStateMachine	Specifies the state machine belonging to the parent state vertex
Transitions	Specifies all outgoing transitions of the state vertex

StateVertex.Parent Property

Description

This property specifies the parent state vertex (activity, state, decision, synchronization).

Note: *This property is read-only.*

Syntax

`Set` *`myStateVertex`* = *`myStateVertex`*.**`Parent`**

Property Type

StateVertex

StateVertex.ParentStateMachine Property

Description

This property specifies the state machine belonging to the parent state vertex (activity, state, decision, synchronization).

Note: *This property is read-only.*

Syntax

`Set` *`myStateMachine`* = *`myStateVertex`*.**`ParentStateMachine`**

Property Type

StateMachine

StateVertex.Transitions Property

Description

This property specifies all outgoing transitions of the state vertex (activity, state, decision, synchronization).

Note: *This property is read-only.*

Syntax

`Set` *`myTransitionCollection`* = *`myStateVertex`*.**`Transitions`**

Property Type

TransitionCollection

StateVertex Class Methods

The following table describes the StateVertex Class methods.

Table 153 StateVertex Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
AddTransition	Adds a transition
AddTransitionToVertex	Adds a transition between activities, states, decisions, or synchronizations
DeleteTransition	Deletes a transition
GetSwimLanes	Retrieves all swimlanes associated with the state vertex

StateVertex.AddTransition Method

Note: *This method has been superceded by the StateVertex.AddTransitionToVertex method.*

Description

This method adds a transition to a state vertex (activity, state, decision, synchronization).

Syntax

```
Set theTransition = myStateVertex.AddTransition (onEvent,
      targetState)
```

Element	Description
theTransition As Transition	Creates and adds a transition to the specified state vertex
myStateVertex As StateVertex	State vertex to which to add the transition
onEvent As String	Description of the event that triggers the transition
targetState As State	State that is the target of the transition

StateVertex. AddTransitionToVertex Method

Description

This method adds a transition from one state vertex (activity, state, decision, synchronization) to another state vertex.

Syntax

```
Set objTransition = objStateVertex.AddTransitionToVertex
       (strOnEvent, objTarget)
```

Element	Description
objTransition As Transition	Creates and adds a transition
objStateVertex As StateVertex	State vertex (activity, state, decision, synchronization) to which to add the transition
strOnEvent As String	Description of the event that triggers the transition
objTarget As StateVertex	State vertex (activity, state, decision, synchronization) that is the target of the transition

See Also

StateVertex.DeleteTransition Method

StateVertex.DeleteTransition Method

Description

This method deletes a transition from a state vertex (activity, state, decision, synchronization).

Syntax

```
isTransitionDeleted = myStateVertex.DeleteTransition
      (theTransition)
```

Element	Description
isTransitionDeleted As Boolean	Returns a value of True if the transition is successfully deleted from the specified state vertex
myStateVertex As StateVertex	State vertex from which to delete the transition
theTransition As Transition	Transition to delete from the state vertex

StateVertex.GetSwimLanes Method

Description

This method retrieves all swimlanes in which the state vertex's view is displayed.

Syntax

```
Set theSwimLaneCollection = myStateVertex.GetSwimLanes ()
```

Element	Description
theSwimLaneCollection As SwimLaneCollection	Returns the collection of swimlanes associated with the specified state vertex
myStateVertex As StateVertex	State vertex (activity, state, decision, synchronization) from which to retrieve the swimlanes

StateView Class

The state view contains the state diagrams belonging to a model. Its properties and methods allow you to control all aspects of the state view, including its display characteristics, its relationships to other model views, the diagrams that belong to the view, etc.

StateView Class Properties

The following table summarizes the StateView Class properties.

Table 154 StateView Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element Class properties
RoseItemView Properties	Inherits all RoseItemView Class properties

StateView Class Methods

The following table summarizes the StateView Class methods.

Table 155 StateView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
RoseItemView Methods	Inherits all RoseItemView class methods
GetState	Retrieves the current state of the state view

StateView.GetState Method

Description

This method retrieves the state that is represented by the specified state view object.

Syntax

```
Set theState = theStateView.GetState ()
```

Element	Description
theState As State	Returns the state represented by the specified state view object
theStateView As StateView	State view object whose state is being retrieved

Subsystem Class

A subsystem is a collection of logically related modules. (The subsystem/module relationship is analagous to the category/class relationship).

The subsystem class exposes properties and methods that allow you to define and manipulate subsystems and their characteristics.

Subsystem Class Properties

The following table summarizes the Subsystem Class properties.

Table 156 Subsystem Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
ControllableUnit	Inherits all ControllableUnit properties
Package Properties	Inherits all Package properties
ModuleDiagrams	Collection that contains the module diagrams assigned to the subsystem

Property	Description
Modules	Module collection that contains the modules belonging to the subsystem
ParentSubSystem	Subsystem that contains the subsystem
Subsystems	Subsystem collection that contains all child subsystems of the subsystem

Subsystem.ModuleDiagrams Property

Description

Contains the module diagrams belonging to the subsystem.

Note: *This property is read-only.*

Syntax

Subsystem.**ModuleDiagrams**

Property Type

ModuleDiagramCollection

Subsystem.Modules Property

Description

Contains the modules belonging to the subsystem.

Note: *This property is read-only.*

Syntax

Subsystem.**Modules**

Property Type

ModuleCollection

Subsystem.ParentSubsystem Property

Description

Identifies the subsystem object that contains the subsystem.

If the subsystem is the root subsystem, then the value of ParentSubsystem is set to *Nothing.*

Note: *You can also use the TopLevel method to check for this condition. This property is read-only.*

Syntax

`Subsystem`.**`ParentSubsystem`**

Property Type

Subsystem

Subsystem.Subsystems Property

Description

Contains the subsystems belonging to the subsystem.

Note: *This property is read-only.*

Syntax

`Subsystem`.**`Subsystems`**

Property Type

SubsystemCollection

Subsystem Class Methods

The following table summarizes the Subsystem Class methods.

Table 157 Subsystem Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
ControllableUnit Methods	Inherits all ControllableUnit methods
Package Methods	Inherits all Package methods
AddModuleDiagram	Adds a module diagram to the subsystem
AddModule	Adds a module to the subsystem
AddSubsystem	Adds a subsystem to the subsystem
DeleteModule	Deletes a module from the subsystem
DeleteSubsystem	Deletes a subsystem from the subsystem
GetAllModules	Retrieves the collection that contains the modules belonging to the subsystem and those belonging to its children
GetAllSubsystems	Retrieves the collection that contains all child subsystems of the subsystem and those belonging to its children
GetAssignedCategories	Retrieves the collection that contains the categories assigned to the subsystem
GetAssignedClasses	Retrieves the collection that contains the classes assigned to the subsystem
GetSubsystemDependencies	Retrieves subsystem dependencies
GetVisibilityRelations	Retrieves the module visibility (dependency) relationships of a subsystem
GetVisibleSubsystems	Retrieves visible subsystems
RelocateModuleDiagram	Relocates a module diagram to/from a subsystem

Method	Description
RelocateModule	Relocates a module to/from a subsystem
RelocateSubSystem	Relocates a subsystem to/from a subsystem
TopLevel	Indicates whether this is the root subsystem

Subsystem.AddModule Method

Description

This method creates a new module in a subsystem and returns it in the specified object.

Syntax

`Set theModule = theSubsystem.AddModule (theName)`

Element	Description
theModule As Module	Returns the newly created module object
theSubsystem As Subsystem	Subsystem to which new module is being added
theName As String	Name of the module to be created

Subsystem.AddModuleDiagram Method

Description

This method creates a new module diagram in a subsystem and returns it in the specified object.

Syntax

```
Set theModuleDiagram = theSubsystem.AddModuleDiagram
       (theName)
```

Element	Description
theModuleDiagram As ModuleDiagram	Returns the newly created module diagram object
theSubsystem As Subsystem	Subsystem to which new module diagram is being added
theName As String	Name of the module diagram to be created

Subsystem.AddSubsystem Method

Description

This method creates a new subsystem in a model and returns it in the specified subsystem object.

Syntax

```
Set theSubsystem = theObject.AddSubsystem (theName)
```

Element	Description
theSubsystem As Subsystem	Returns the newly created subsystem
theObject As Subsystem	Instance of the subsystem being created
theName As String	Name of the subsystem being created

Subsystem.DeleteModule Method

Description

This method deletes a module from a subsystem.

Syntax

IsDeleted = *theSubsystem*.**DeleteModule** (*theModule*)

Element	Description
IsDeleted As Boolean	Returns a value of True when the module is successfully deleted
theSubsystem As Subsystem	Subsystem from which to delete the module
theModule As Module	Module being deleted

Subsystem.DeleteSubsystem Method

Description

This subroutine deletes a subsystem from a subsystem.

Syntax

IsDeleted = *theSubsystem*.**DeleteSubsystem** (*theSubsystem*)

Element	Description
IsDeleted As Boolean	Returns a value of True when the subsystem is successfully deleted
theSubsystem As Subsystem	Subsystem from which to delete the subsystem
theSubsystem As Subsystem	Subsystem being deleted

Subsystem.GetAllModules Method

Description

This method retrieves all modules belonging to a subsystem.

Syntax

```
Set theModules = theObject.GetAllModules ()
```

Element	Description
theModules As ModuleCollection	Returns all modules belonging to the subsystem
theObject As Subsystem	Subsystem whose modules are being retrieved

Subsystem.GetAllSubsystems Method

Description

This method retrieves all subsystems belonging to a subsystem.

Syntax

```
Set theSubsystems = theObject.GetAllSubsystems ()
```

Element	Description
theSubsystems As SubsystemCollection	Returns all subsystems belonging to the subsystem
theObject As Subsystem	Subsystem whose subsystems are being retrieved

Subsystem.GetAssignedCategories Method

Description

This method retrieves the categories assigned to a subsystem.

Syntax

`Set` *`theCategories`* `=` *`theObject`*`.`**`GetAssignedCategories`** `()`

Element	Description
theCategories As CategoryCollection	Returns the categories assigned to the subsystem
theObject As Subsystem	Subsystem whose categories are being retrieved

Subsystem.GetAssignedClasses Method

Description

This method retrieves the classes assigned to a subsystem.

Syntax

`Set` *`theClasses`* `=` *`theObject`*`.`**`GetAssignedClasses`** `()`

Element	Description
theClasses As ClassCollection	Returns the classes assigned to the subsystem
theObject As Subsystem	Subsystem whose classes are being retrieved

Subsystem.GetSubsystemDependencies Method

Description

This method retrieves the subsystem dependency collection from a subsystem and returns it in the specified object.

Syntax

```
Set theSubDependencies =
      theClientSubsys.GetSubsystemDependencies
      (theSupplierSubsys)
```

Element	Description
theSubDependencies As ModuleVisibilityRelationshipCollection	Returns the subsystem dependency collection from the subsystem
theClientSubsys As Subsystem	Subsystem from which the collection is being retrieved
theSupplierSubsys As Subsystem	Supplier subsystem of the dependency

Subsystem.GetVisibilityRelations Method

Description

This method retrieves the module visibility (dependency) relationships between the specified subsystem and its supplier subsystems and modules. For an example of how to use this method, see the sample code in the next section.

Syntax

```
Set theModVisRels = theSubsystem.GetVisibilityRelations ()
```

Element	Description
theModVisRels As ModuleVisibilityRelationshipCollection	Returns the subsystem's module visibility relationships
theSubsystem As Subsystem	Subsystem from which the collection is being retrieved

Sample Code for GetVisibilityRelations

The following sample Rational Rose Script gets the module visibility relationships for a subsystem. The script then iterates through the collection and prints the name of the subsystem or module at the other end of the relationship.

```
Sub Main
   Dim theSubsystems As SubsystemCollection
   Dim theSubsystem As Subsystem
   Dim theDependencies As
     ModuleVisibilityRelationshipCollection
   Dim thedependency As ModuleVisibilityRelationship
   Dim theModule As Module

   viewport.open
   Set theSubsystems = RoseApp.CurrentModel.GetAllSubSystems ()
   Set theSubsystem = theSubsystems.GetFirst ("NewPackage")
   Set theDependencies = theSubsystem.GetVisibilityRelations ()
   total = theDependencies.count
   For i = 1 To total
      Set thedependency = theDependencies.GetAt (i)
      Set theSubsystem = thedependency.SupplierSubsystem
      Set theModule = thedependency.SupplierModule
      Print "The subsystem for dependency ";i;" is:"
      Print theSubsystem.Name
      If theModule Is Not Nothing Then
         Print "The module for dependency ";i;" is:"
         Print theModule.Name
      Else
         Print "There are NO modules for this dependency."
      End If
      Print "*****"
      Print
   Next i
End Sub
```

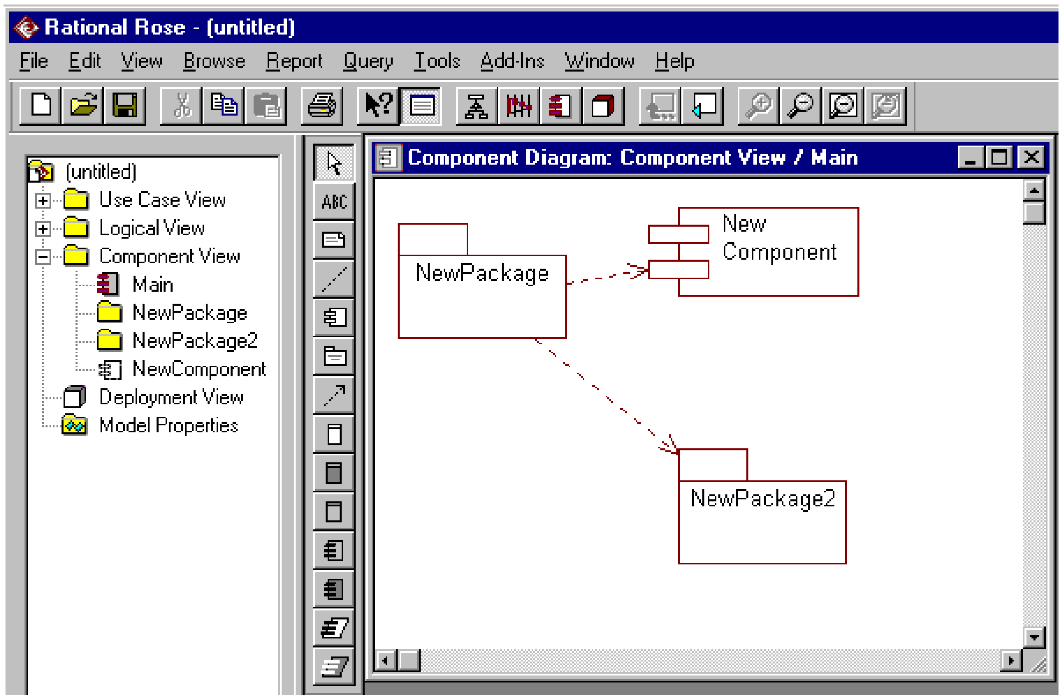

Figure 4 Example: Retrieving Module Visibility Relationships

Output

```
The subsystem for dependency 1 is:
Component View
The module for dependency 1 is:
NewComponent
*****


The subsystem for dependency 2 is:
NewPackage2
There are NO modules for this dependency.
*****
```

Subsystem.GetVisibleSubsystems Method

Description

This method retrieves the visible subsystems collection from a subsystem and returns it in the specified object.

Syntax

```
Set theVisSubsystems = theObject.GetVisibleSubsystems ()
```

Element	Description
theVisSubsystems As SubsystemCollection	Returns the visible subsystem collection from the subsystem
theObject As Subsystem	Subsystem from which the collection is being retrieved

Subsystem.RelocateModule Method

Description

This subroutine relocates a module in a subsystem.

Syntax

```
theObject.RelocateModule theModule
```

Element	Description
theObject As Subsystem	Subsystem that contains the module being relocated
theModule As Module	Module being relocated

Subsystem.RelocateModuleDiagram Method

Description

This subroutine relocates a module diagram in a subsystem.

Syntax

theObject.**RelocateModuleDiagram** *theModuleDiagram*

Element	Description
theObject As Subsystem	Subsystem that contains the module diagram being relocated
theModuleDiagram As ModuleDiagram	Module diagram being relocated

Subsystem.RelocateSubSystem Method

Description

This subroutine relocates a subsystem in a model.

Syntax

theObject.**RelocateSubsystem** *theSubsystem*

Element	Description
theObject As Model	Model that contains the subsystem being relocated
theSubsystem As Subsystem	Subsystem being relocated

Subsystem.TopLevel Method

Description

This method determines whether the specified object is the root subsystem.

Syntax

```
IsTopLevel = theObject.TopLevel ()
```

Element	**Description**
IsTopLevel As Boolean	Returns a value of True if the specified object is the root category
theObject As Subsystem	Subsystem object being tested as root subsystem

SubsystemView Class

Subsystems contain modules, as well as other subsystems. The subsystem view is the visual representation of a subsystem, and is what appears on a diagram in the model.

The subsystem view class inherits the RoseItemView properties and methods that determine the size and placement of the subsystem view. It also allows you to retrieve the subsystem object itself from the subsystem view.

SubsystemView Class Properties

The following table summarizes the SubsystemView Class properties.

Table 158 SubsystemView Class Properties Summary

Property	**Description**
Element	Inherits all Element class properties
RoseItemView Properties	Inherits all RoseItemView class properties

SubsystemView Class Methods

The following table summarizes the SubsystemView Class methods.

Table 159 SubsystemView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject methods
Element	Inherits all Element class methods
RoseItemView	Inherits all RoseItemView class methods
GetSubsystem	Retrieves the subsystem object represented by the subsystem view

SubsystemView.GetSubsystem Method

Description

This method retrieves the Subsystem represented by the subsystem view.

Syntax

`Set` *`theSubsystem`* `=` *`theSubsystemView`*`.`**`GetSubsystem`** `()`

Element	**Description**
theSubsystem As Subsystem	Returns the Subsystem represented by the SubsystemView. Note that the REI return class is currently called Module, not Subsystem
theSubsystemView As SubsystemView	Instance of the SubsystemView whose corresponding Subsystem (module) is being retrieved

SwimLane Class

The SwimLane class is an abstract class that exposes Rose's swimlane functionality in the extensibility interface. With the properties and methods of the SwimLane class, you can:

- Retrieve information about swimlanes, such as name, application, model, documentation, stereotypes, and external documents
- Create and retrieve tool and property settings for swimlanes
- Add and delete external documents
- Open specification sheets for swimlanes
- Retrieve all objects associated with a particular swimlane, including states, activities, decisions, and synchronizations

The SwimLane class corresponds to swimlanes in the Rose user interface.

SwimLane Class Properties

The following table describes the SwimLane Class properties.

Table 160 SwimLane Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties

SwimLane Class Methods

The following table describes the SwimLane Class methods.

Table 161 SwimLane Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
GetStateVertices	Returns a collection which contains all state vertices (states, activities, decisions, synchronizations) assigned to the swimlane

SwimLane.GetStateVertices Method

Description

This method retrieves a collection which contains all state vertices (activities, states, decisions, synchronizations) assigned to the swimlane

Syntax

```
Set theStateVertexCollection = mySwimLane.GetStateVertices
       ()
```

Element	Description
theStateVertexCollection As StateVertexCollection	Returns the state vertices (activities, states, decisions, synchronizations) assigned to the swimlane
mySwimLane As SwimLane	Swimlane from which to retrieve the state vertices

SwimLaneView Class

The SwimLaneView class is an abstract class that exposes Rose's swimlane view functionality of activity diagrams in the extensibility interface. With the properties and methods of the SwimLaneView class, you can:

- Retrieve information about the swimlane view, such as name, parent application, parent model
- Retrieve objects associated with the swimlane view, such as the diagram it is on, any parent or child views, line vertices, and the swimlane object represented by the swimlane view
- Retrieve physical information about the swimlane view such as position, height, width, fill color, line color, font
- Create and retrieve tool and property settings for swimlane views

The SwimLaneView class corresponds to swimlanes on diagrams in the Rose user interface.

SwimLaneView Class Properties

The following table describes the SwimLaneView Class properties.

Table 162 SwimLaneView Class Properties Summary

Property	Description
Element Properties	Inherits all Element class properties
RoseItemView Properties	Inherits all RoseItemView class properties

SwimLaneView Class Methods

The following table describes the SwimLaneView Class methods.

Table 163 SwimLaneView Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItemView Methods	Inherits all RoseItemView class methods

SyncItem Class

The SyncItem class is an abstract class that exposes Rose's synchronization functionality in the extensibility interface. With the properties and methods of the SyncItem class, you can:

- Retrieve information about synchronizations, such as name, parent application, parent model, documentation, stereotypes, and external documents
- Retrieve objects associated with synchronizations, such as parent state vertex, parent state machine, transitions, and swimlanes
- Create and retrieve tool and property settings for synchronizations
- Add and delete external documents
- Open specification sheets for synchronizations
- Add and delete transitions

The SyncItem class corresponds to synchronizations in the Rose user interface.

SyncItem Class Properties

The following table describes the SyncItem Class properties.

Table 164   SyncItem Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItem Properties	Inherits all RoseItem class properties
StateVertex Properties	Inherits all StateVertex class properties

SyncItem Class Methods

The following table describes the SyncItem Class methods.

Table 165   SyncItem Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItem Methods	Inherits all RoseItem class methods
StateVertex Methods	Inherits all StateVertex class methods

SyncItemView Class

The SyncItemView class is an abstract class that exposes Rose's synchronization view functionality of activity diagrams in the extensibility interface. With the properties and methods of the SyncItemView class, you can:

- Retrieve information about synchronization views, such as name, parent application, parent model
- Retrieve objects associated with synchronization views, such as the diagram it is on, any parent or child views, line vertices, and the synchronization object represented by the synchronization view
- Retrieve physical information about the synchronization view such as position, height, width, fill color, line color, font, whether the synchronization view is horizontal or vertical

- Create and retrieve tool and property settings for synchronization views

The SyncItemView class corresponds to synchronizations on diagrams in the Rose user interface.

SyncItemView Class Properties

The following table describes the SyncItemView Class properties.

Table 166 SyncItemView Class Properties Summary

Property	**Description**
Element Properties	Inherits all Element class properties
RoseItemView Properties	Inherits all RoseItemView class properties
Horizontal	Indicates whether the synchronization view is horizontal

SyncItemView.Horizontal Property

Description

This property Indicates whether the synchronization view is horizontal. If this property is True, the synchronization view is displayed horizontally on the diagram. If this property is False, the synchronization view is displayed vertically on the diagram.

Syntax

isHorizontal = *mySyncItemView*.**Horizontal**

mySyncItemView.**Horizontal** = True

Property Type

Boolean

SyncItemView Class Methods

The following table describes the SyncItemView Class methods.

Table 167 SyncItemView Class Methods Summary

Method	**Description**
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element class methods
RoseItemView Methods	Inherits all RoseItemView class methods
GetSynchronization	Retrieves the synchronization represented by the synchronization view

SyncItemView.GetSynchronization Method

Description

This method retrieves the synchronization represented by the synchronization view.

Syntax

```
Set theSyncItem = mySyncItemView.GetSynchronization ()
```

Element	**Description**
theSyncItem As SyncItem	Returns synchronization represented by the synchronization view
mySyncItemView As SyncItemView	Synchronization view from which to retrieve the synchronization

Transition Class

The Transition class provides a means for tracking an object through a state change; that is the point at which it is no longer in its original state and has not yet reached its target state.

Transition Class Properties

The following table summarizes the Transition Class properties.

Table 168 Transition Class Properties Summary

Property	Description
Element Properties	Inherits all Element Class properties
RoseItem Properties	Inherits all RoseItem Class properties
Relation	Inherits all Relation Class properties

Transition Class Methods

The following table summarizes the Transition Class methods.

Table 169 Transition Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element Class methods
RoseItem Methods	Inherits all Element Class methods
Relation	Inherits all Relation Class methods
GetSendAction	Retrieves the message to send when the transition occurs
GetSourceState	Retrieves the transition's initial state
GetSourceStateVertex	Retrieves the transition's source state vertex (activity, state, decision, synchronization)
GetTargetState	Retrieves the transition's target state
GetTargetStateVertex	Retrieves the transition's target state vertex (activity, state, decision, synchronization)

Method	Description
GetTriggerAction	Retrieves the action to perform when the transition's trigger event occurs
GetTriggerEvent	Retrieves the event that triggers the transition
RedirectTo	Redirects the transition to a new target state

Transition.GetSendAction Method

Description

This method retrieves the message to send when the transition occurs.

Syntax

`Set` *`theAction`* `=` *`theTransition`*`.`**`GetSendAction`** `()`

Element	Description
theAction As Action	Returns the message to send when the transition occurs
theTransition As Transition	The transition whose send action is being retrieved

Transition.GetSourceState Method

Note: *This method has been superceded by the Transition.GetSourceStateVertex method.*

Description

This method retrieves the source (initial) state of the specified transition.

Syntax

```
Set theSource = theTransition.GetSourceState ()
```

Element	Description
theSource As State	Returns the source state of the specified transition
theTransition As Transition	The transition whose source state is being retrieved

Transition. GetSourceStateVertex Method

Description

This method retrieves the source state vertex (activity, state, decision, synchronization) of the specified transition.

Syntax

```
Set objStateVertex = objTransition.GetSourceStateVertex ( )
```

Element	Description
objStateVertex As StateVertex	Returns the state vertex (activity, state, decision, or synchronization) that is the source for the transition
objTransition As Transition	Transition whose source state vertex is being retrieved

See also

Transition.GetTargetStateVertex Method

Transition.GetTargetState Method

Note: *This method has been superceded by the Transition.GetTargetStateVertex method.*

Description

This method retrieves the target state of the specified transition.

Syntax

```
Set theTarget = theTransition.GetTargetState ()
```

Element	Description
theTarget As State	Returns the target state of the specified transition
theTransition As Transition	The transition whose target state is being retrieved

Transition. GetTargetStateVertex Method

Description

This method retrieves the target state vertex (activity, state, decision, synchronization) of the specified transition.

Syntax

```
Set objStateVertex = objTransition.GetTargetVertexState ( )
```

Element	Description
objStateVertex As StateVertex	Returns the state vertex (activity, state, decision, or synchronization) that is the target of the transition
objTransition As Transition	Transition whose target state vertex is being retrieved

See also

Transition.GetSourceStateVertex Method

Transition.GetTriggerAction Method

Description

This method retrieves the action to perform when the transition's trigger event occurs.

Syntax

`Set` *`theAction`* `=` *`theTransition`*`.`**`GetTriggerAction`** `()`

Element	**Description**
theAction As Action	Returns the action to perform when the transition's trigger event occurs
theTransition As Transition	The transition whose trigger action is being retrieved

Transition.GetTriggerEvent Method

Description

This method retrieves the event that triggers the specified transition.

Syntax

`Set` *`theTrigger`* `=` *`theTransition`*`.`**`GetTriggerEvent`** `()`

Element	**Description**
theTrigger As Event	Returns the event that triggers the specified transition
theTransition As Transition	The transition whose trigger event is being retrieved

Transition.RedirectTo Method

Description

This method redirects the transition to a new target.

Syntax

IsRedirected = *theTransition*.**RedirectTo** (*NewTargetState*)

Element	Description
IsRedirected As Boolean	Returns a value of True when the transition is successfully redirected
theTransition As Transition	The transition whose target state is being changed (redirected)
NewTargetState As State	State to which the transition is to be redirected

UseCase Class

The UseCase class exposes properties and methods that allow you to define and manipulate the sets of class diagrams and scenario diagrams that comprise a model's use cases.

UseCase Class Properties

The following table summarizes the UseCase Class properties.

Table 170 UseCase Class Properties Summary

Property	Description
Element Properties	Inherits all Element properties
RoseItem Properties	Inherits all RoseItem properties
Abstract	Indicates that the use case is an abstract class
ClassDiagrams	Collection that contains the class diagrams belonging to the use case
ParentCategory	Category that contains the use case

Property	Description
Rank	Specifies the rank of the use case
ScenarioDiagrams	Collection that contains the scenario diagrams belonging to the use case
StateMachine	Specifies the state machine belonging to the use case

UseCase.Abstract Property

Description

This property indicates whether the UseCase is an abstract class.

Syntax

UseCase.**Abstract**

Property Type

Boolean

UseCase.ClassDiagrams Property

Description

Specifies the collection of class diagrams belonging to the use case

Note: *This property is read-only.*

Syntax

UseCase.**ClassDiagram**

Property Type

ClassDiagramCollection

UseCase.ParentCategory Property

Description

If the UseCase is the root category, then the value of the parent category will be set to *Nothing.*

You can check its value by using the TopLevel method.

Note: *This property is read-only.*

Syntax

`UseCase``.`**`ParentCategory`**

Property Type

Category

UseCase.Rank Property

Description

Specifies the rank of the use case.

Syntax

`UseCase``.`**`Rank`**

Property Type

String

UseCase.ScenarioDiagrams Property

Description

Specifies the collection of scenario diagrams belonging to the use case.

Note: *This property is read-only.*

Syntax

`UseCase``.`**`ScenarioDiagrams`**

Property Type

ScenarioDiagramCollection

UseCase.StateMachine Property

Description

This property specifies the state machine that belongs to the use case. A state machine defines all of the state information, including states, transitions, and state diagrams, defined for a given use case.

A use case can have zero or one state machine.

If the use case does not have a state machine, this property returns a **Nothing** or **Null** value. To create a state machine, please see the UseCase's CreateStateMachine method. For an example of how to use StateMachine and CreateStateMachine, see Sample Code to Check for and Create a State Machine.

Note: *This property is read-only.*

Syntax

`theStateMachine` `=` *`theUseCase`*`.`**`StateMachine`**

Property Type

StateMachine

Sample Code to Check for and Create a State Machine

The following sample Rational Rose Script checks to see if a use case has a state machine. If not, the script then creates the state machine.

```
Sub Main
   Dim theStateMachine As StateMachine
   Dim theUseCase As UseCase
   Dim theUseCases As UseCaseCollection

   Set theUseCases = RoseApp.CurrentModel.GetAllUseCases ( )
   Set theUseCase = theUseCases.Get... 'Get the particular use
                                       'case

   'Get the use case's state machine
   Set theStateMachine = theUseCase.StateMachine
   If theStateMachine Is Nothing Then
      'The state machine does not exist.  Therefore, create it
      theUseCase.CreateStateMachine
   Else
      Print "The state machine exists"
   End If
End Sub
```

UseCase Class Methods

The following table summarizes the UseCase Class methods.

Table 171 UseCase Class Methods Summary

Method	Description
RoseObject Methods	Inherits all RoseObject class methods
Element Methods	Inherits all Element methods
RoseItem Methods	Inherits all RoseItem methods
AddAssociation	Adds an association to a use case
AddClassDiagram	Adds a class diagram to a use case
AddInheritRel	Adds an inherit relation to a use case
AddScenarioDiagram	Adds a scenario diagram to a use case
AddUseCaseDiagram	Adds a use case diagram to a use case
CreateStateMachine	Creates a state machine for a use case
DeleteAssociation	Deletes an association from a use case
DeleteClassDiagram	Deletes a class diagram from a use case
DeleteInheritRel	Deletes an inherit relation from a use case
DeleteScenarioDiagram	Deletes a scenario diagram from a use case
GetAssociations	Retrieves the associations belonging to the use case
GetInheritRelations	Retrieves the collection of inherit relations belonging to the use case
GetRoles	Retrieves the collection of roles belonging to the use case
GetSuperUseCases	Retrieves the super use cases belonging to the use case

UseCase.AddAssociation Method

Description

This method adds an association to a use case and returns it in the specified object.

Syntax

```
Set theAssociation = theUseCase.AddAssociation
      (theSupplierRoleName, theSupplierName)
```

Element	Description
theAssociation As Association	Returns the association being added to the use case
theUseCase As UseCase	Use case to which the association is being added
theSupplierRoleName As String	Name of the supplier role in the association
theSupplierName As String	Name of the class, use case, or actor to which to attach the association ***Note:*** *If this name is not unique, you must use the qualified name (for example, Logical View::package_name::supplier_name)*

UseCase.AddClassDiagram Method

Description

This method creates a new class diagram and adds it to a use case.

Syntax

Set *theClassDiagram* = *theObject*.**AddClassDiagram** (*theName*)

Element	**Description**
theClassDiagram As ClassDiagram	Returns the class diagram being added to the use case
theObject As UseCase	UseCase to which the diagram is being added
theName As String	The name of the class diagram to be added

UseCase.AddInheritRel Method

Description

This method adds an inherit relation to a use case.

Syntax

```
Set theInheritRel = theObject.AddInheritRel
       (theName,theParentName)
```

Element	**Description**
theInheritRel As InheritRelation	Returns the inherit relation being added to the use case
theObject As UseCase	Use case to which the relationship is being added
theName As String	Name of the relationship being added to the use case
theParentName As String	Name of the parent use case from which this use case inherits its properties and methods

UseCase.AddScenarioDiagram Method

Description

This method adds a scenario diagram to a use case.

Syntax

```
Set theScenarioDiagram = theObject.AddScenarioDiagram
      (theName, theType)
```

Element	Description
theScenarioDiagram As Scenario Diagram	Returns the scenario diagram being added to the use case
theObject As UseCase	UseCase to which the scenario diagram is being added
theName As String	Name of the scenario diagram being added
theType As Integer	Type of scenario diagram being added Valid values are: 1 = Sequence Diagram 2 = Collaboration Diagram

UseCase.AddUseCaseDiagram Method

Description

This method adds a use case diagram to a use case.

Syntax

```
Set theUseCaseDiagram = theUseCase.AddUseCaseDiagram
      (theName)
```

Element	Description
theUseCaseDiagram As ClassDiagram	Returns the use case diagram being added to the use case ***Note:*** *The REI implements use case diagrams as class diagrams. Therefore, to determine whether a particular class diagram is actually a use case diagram, use ClassDiagram's IsUseCaseDiagram method.*
theUseCase As UseCase	UseCase to which the use case diagram is being added
theName As String	Name of the use case diagram being added

UseCase.CreateStateMachine Method

Description

This method creates a state machine for the specified use case. Use this to create a state machine after receiving a **Nothing** or **Null** value from the UseCase's StateMachine property. For an example of how to use StateMachine and CreateStateMachine, see Sample Code to Check for and Create a State Machine. In the UseCase.StateMachine property section of this reference guide.

Syntax

theUseCase.**CreateStateMachine**

Element	Description
theUseCase As UseCase	Use case for which the state machine is being created

UseCase.DeleteAssociation Method

Description

This method deletes an association from a use case.

Syntax

Deleted = *theObject*.**DeleteAssociation** (*theAssociation*)

Element	Description
Deleted As Boolean	Returns a value of True when the association is deleted
theObject As UseCase	Use case from which the association is being deleted
theAssociation As Association	Instance of the association being deleted (The association must belong to the specified use case.)

UseCase.DeleteClassDiagram Method

Description

This method deletes a class diagram from a use case.

Syntax

deleted = *theObject*.**DeleteClassDiagram** (*theClassDiagram*)

Element	Description
deleted As Boolean	Returns a value of True when the class diagram is deleted
theObject As UseCase	Use case from which the class diagram is being deleted
theClassDiagram As ClassDiagram	Instance of the class diagram being deleted

UseCase.DeleteInheritRel Method

Description

This method deletes an InheritRelation from a use case.

Syntax

Deleted = *theObject*.**DeleteInheritRel** (*theInheritRel*)

Element	Description
Deleted As Boolean	Returns a value of True when the InheritRelation is deleted from the use case
theObject As UseCase	Use case from which the relationship is being deleted
theInheritRel As InheritRelation	InheritRelation being deleted from the use case

UseCase.DeleteScenarioDiagram Method

Description

This method deletes a scenario diagram from a use case.

Syntax

```
Deleted = theObject.DeleteScenarioDiagram
      (theScenarioDiagram)
```

Element	Description
Deleted As Boolean	Returns a value of True when the scenario diagram is deleted
theObject As UseCase	Instance of the use case from which the scenario diagram is being deleted
theScenarioDiagram As ScenarioDiagram	Instance of the scenario diagram being deleted

UseCase.GetAssociations Method

Description

This method retrieves an association collection from a use case and returns it in the specified object.

Syntax

```
Set theAssociations = theObject.GetAssociations ()
```

Element	Description
theAssociations As AssociationCollection	Returns the association collection from the use case
theObject As UseCase	Use case from which the collection is being retrieved

UseCase.GetInheritRelations Method

Description

This method retrieves an inherit relation collection from a use case and returns it in the specified object.

Syntax

Set *theInheritRelations* = *theObject*.**GetInheritRelations** ()

Element	**Description**
theInheritRelations As InheritRelationCollection	Returns the InheritRelation collection from the use case
theObject As UseCase	Use case from which the collection is being retrieved

UseCase.GetRoles Method

Description

This method retrieves a role collection from a use case and returns it in the specified object.

Syntax

Set *theRoles* = *theObject*.**GetRoles** ()

Element	**Description**
theRoles As RoleCollection	Returns the role collection from the class
theObject As UseCase	UseCase from which the collection is being retrieved

UseCase.GetSuperUseCases Method

Description

This method retrieves a super use case collection from a use case and returns it in the specified object.

Syntax

`Set` *`theSuperUseCases`* `=` *`theObject`*`.`**`GetSuperUseCases`** `()`

Element	**Description**
theSuperUseCases As UseCaseCollection	Returns the super use case collection from the use case
theObject As UseCase	Use case from which the collection is being retrieved

Appendix A

REI Inheritance Diagrams

Introduction

This Appendix is a series of inheritance diagrams that show the inheritance relationships between each exposed class in the Rational Rose Extensibility Model. Figure 5 shows the RoseBase Class and all the classes that inherit from it. Figure 2 shows the RoseObject Class and the first few levels that inherit from it. Subsequent inheritance diagrams display logical portions of the inheritance trees that inherit from RoseObject.

RoseBase Inheritance Diagram

Figure 5 shows the classes that inherit from the RoseBase Class in the Rational Rose Extensibility Interface.

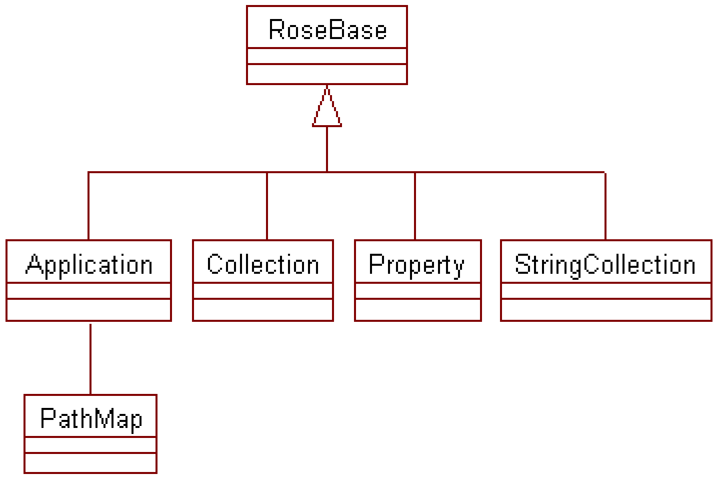

Figure 5 RoseBase Inheritance Diagram

RoseObject Inheritance Diagram

Figure 6 shows the first few levels of classes that inherit from the RoseObject class in the Rational Rose Extensibility Interface model. The inheritance diagrams following this diagram continue down the RoseObject inheritance tree in logical groups.

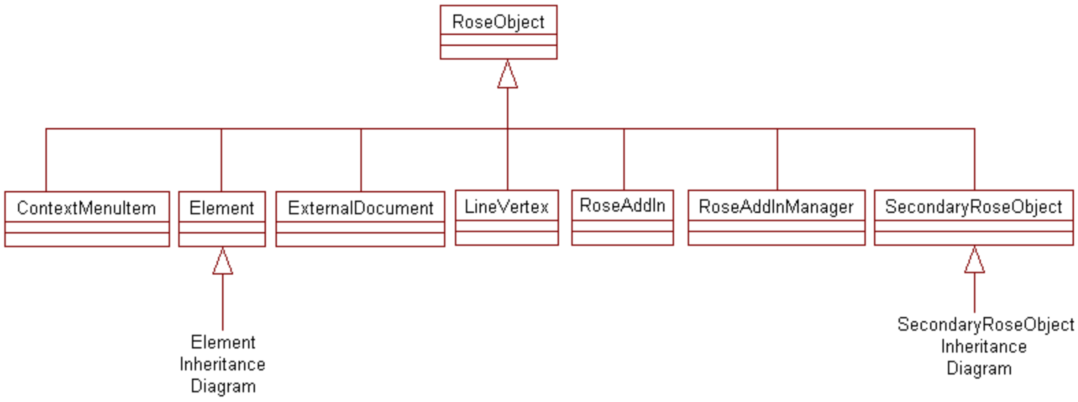

Figure 6 RoseObject Inheritance Diagram

Element Inheritance Diagram

Figure 7 shows the first few levels of classes that inherit from the Element class in the Rational Rose Extensibility Interface.

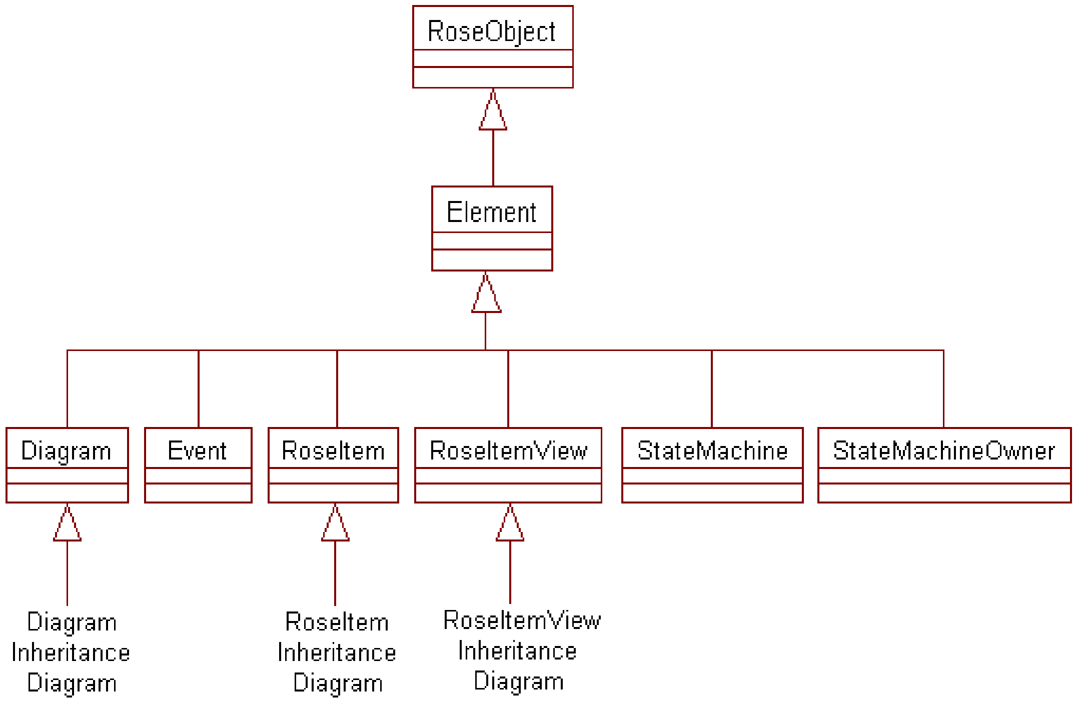
RoseObject
Element
Diagram
Event
RoseItem
RoseItemView
StateMachine
StateMachineOwner
Diagram Inheritance Diagram
RoseItem Inheritance Diagram
RoseItemView Inheritance Diagram

Figure 7 Element Inheritance Diagram

Diagram Inheritance Diagram

Figure 8 shows the classes that inherit from the Diagram class in the Rational Rose Extensibility Interface.

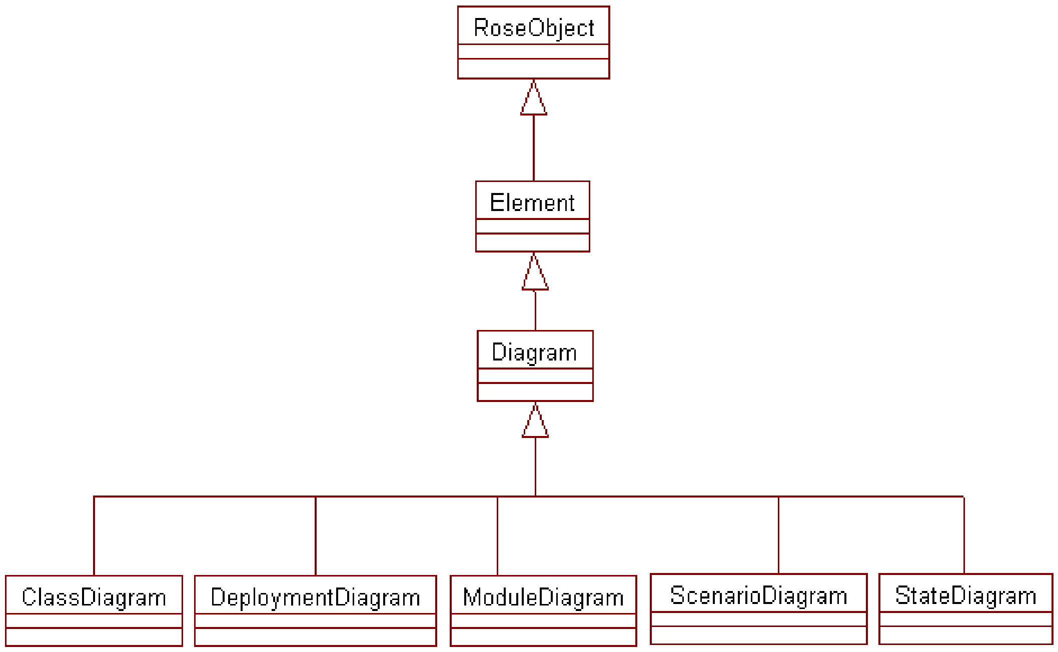

Figure 8   Diagram Inheritance Diagram

RoseItem Inheritance Diagram

Figure 9 shows the first few levels of classes that inherit from the RoseItem class in the Rational Rose Extensibility Interface.

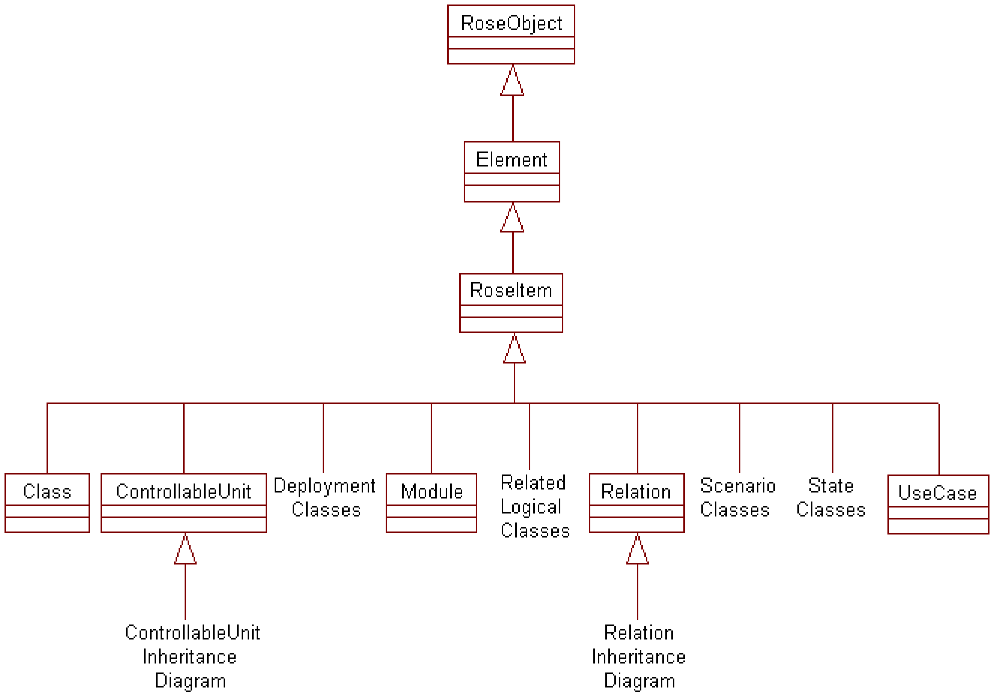

Figure 9 RoseItem Inheritance Diagram

ControllableUnit Inheritance Diagram

Figure 10 shows the classes that inherit from the ControllableUnit class in the Rational Rose Extensibility Interface.

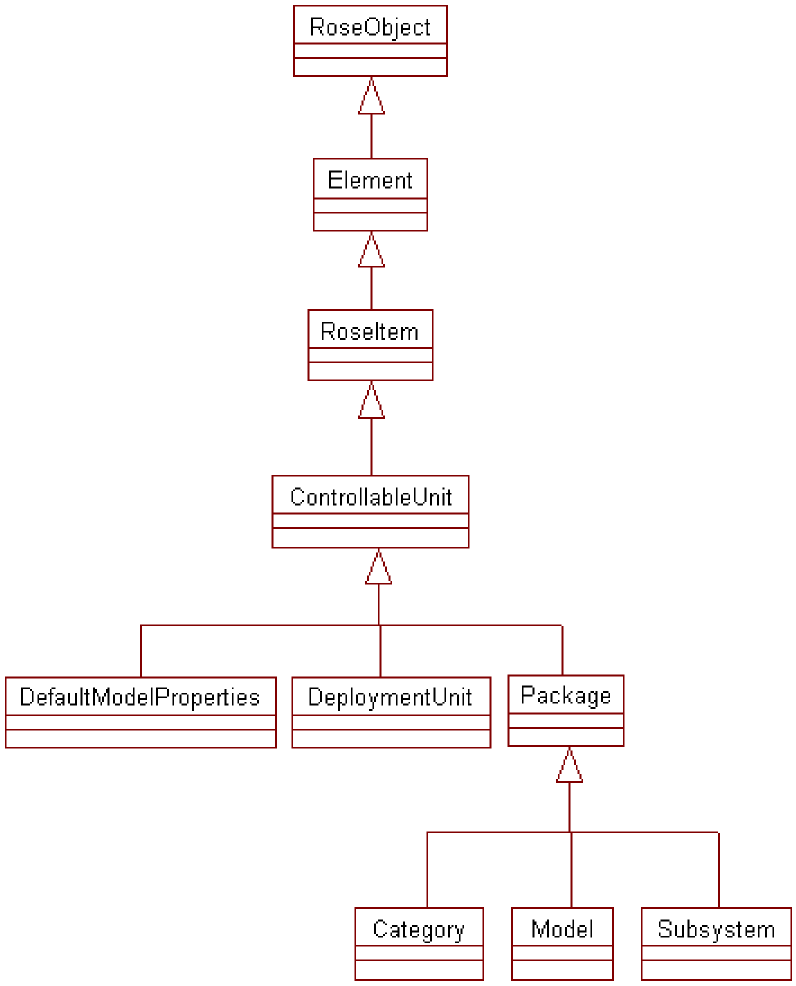

Figure 10 ControllableUnit Inheritance Diagram

Deployment Classes

Figure 11 shows the Deployment classes that inherit from the RoseItem class in the Rational Rose Extensibility Interface.

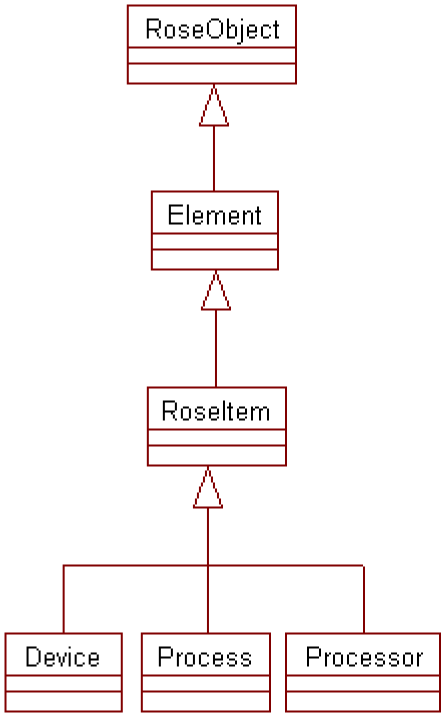

Figure 11 Deployment Classes

Related Logical Classes

Figure 12 shows the Related Logical classes that inherit from the RoseItem class in the Rational Rose Extensibility Interface.

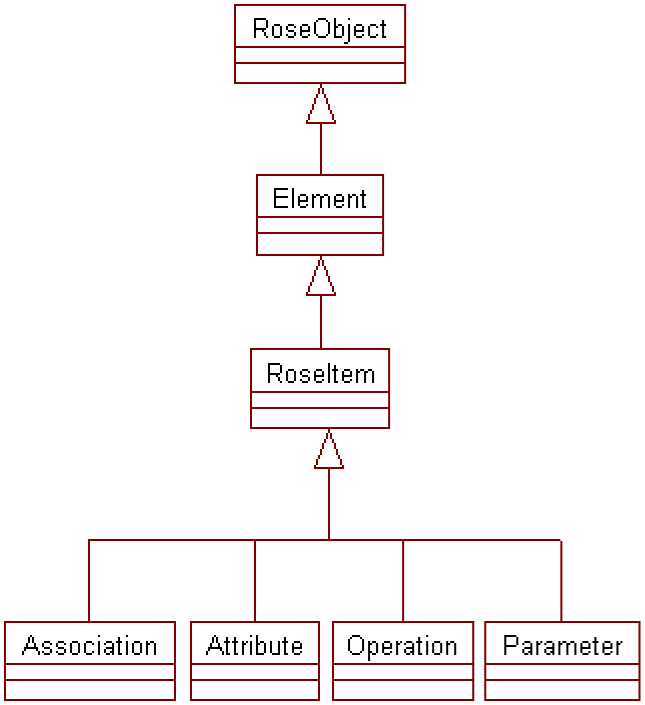

Figure 12   Related Logical Classes

Relation Inheritance Diagram

Figure 13 shows the first few levels of classes that inherit from the Relation class in the Rational Rose Extensibility Interface.

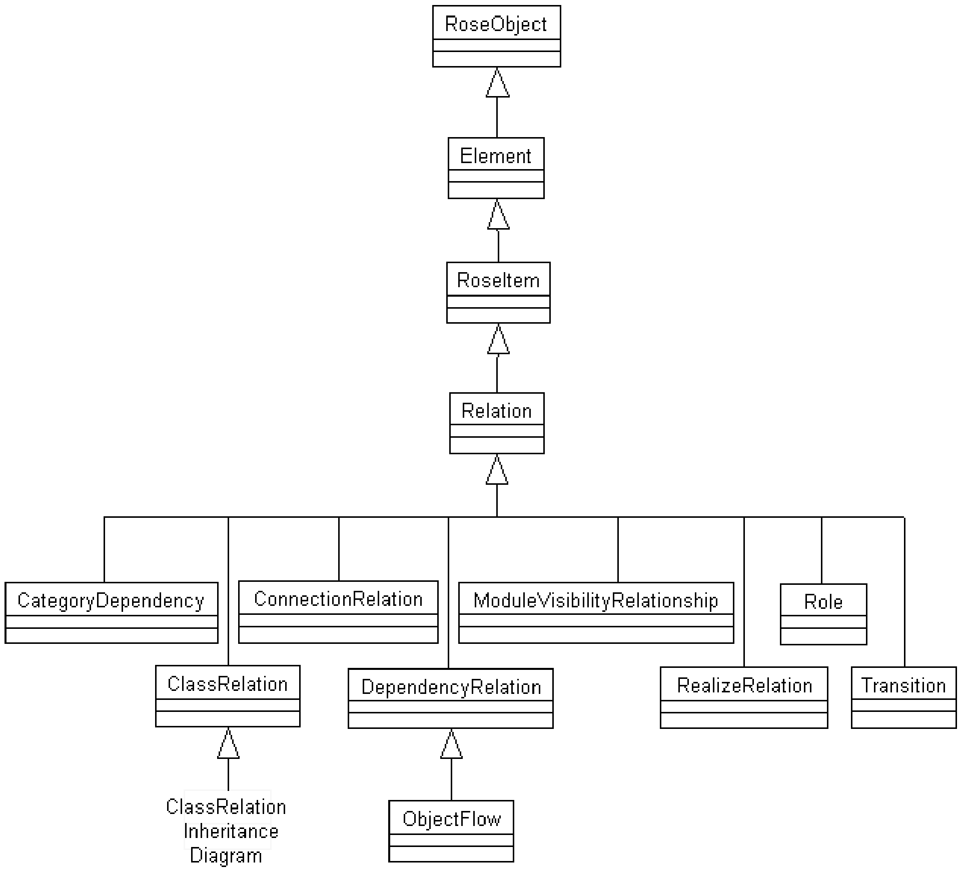

Figure 13 Relation Inheritance Diagram

ClassRelation Inheritance Diagram

Figure 14 shows the classes that inherit from the ClassRelation class in the Rational Rose Extensibility Interface.

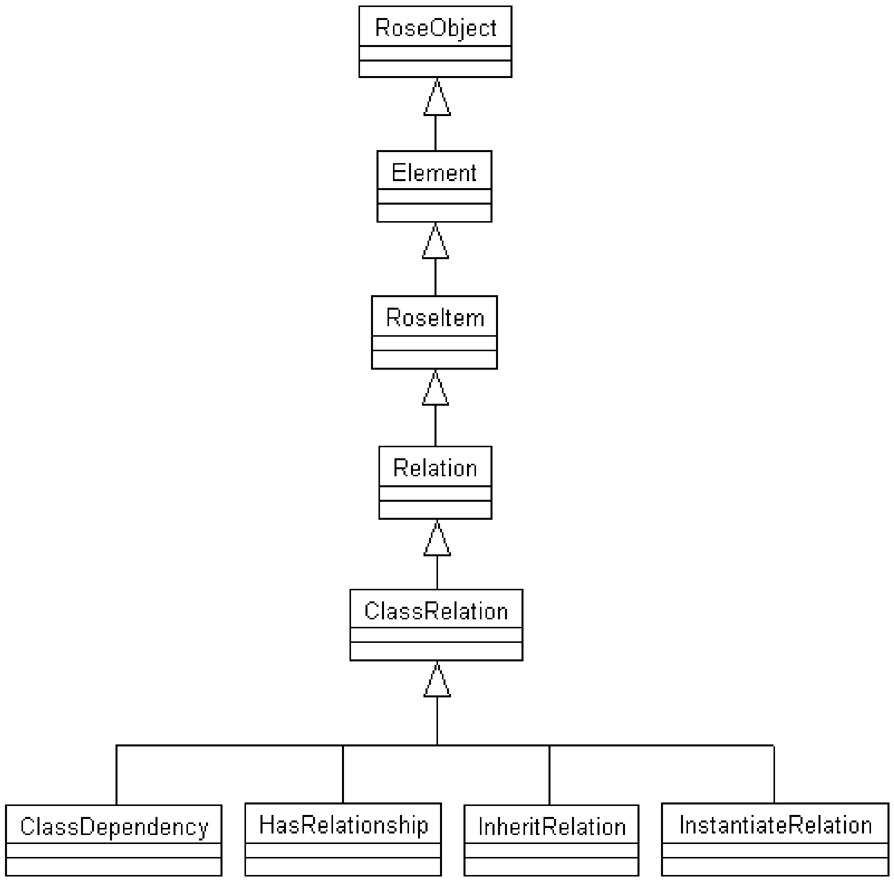

Figure 14 ClassRelation Inheritance Diagram

Scenario Classes

Figure 15 shows the Scenario classes that inherit from the RoseItem class in the Rational Rose Extensibility Interface.

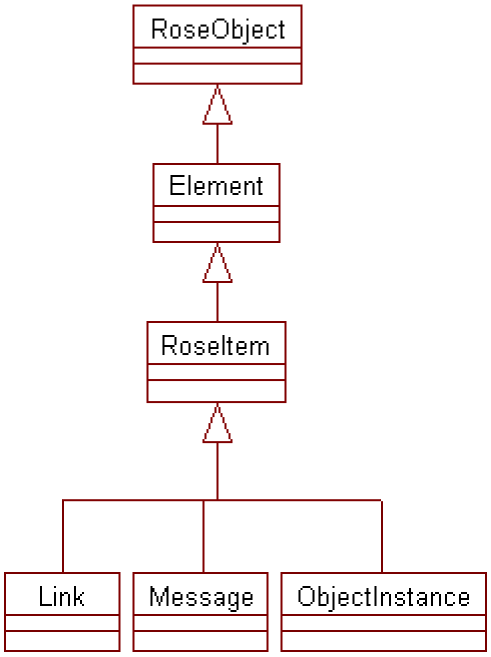

Figure 15   Scenario Classes

State Classes

Figure 16 shows the State classes that inherit from the RoseItem class in the Rational Rose Extensibility Interface.

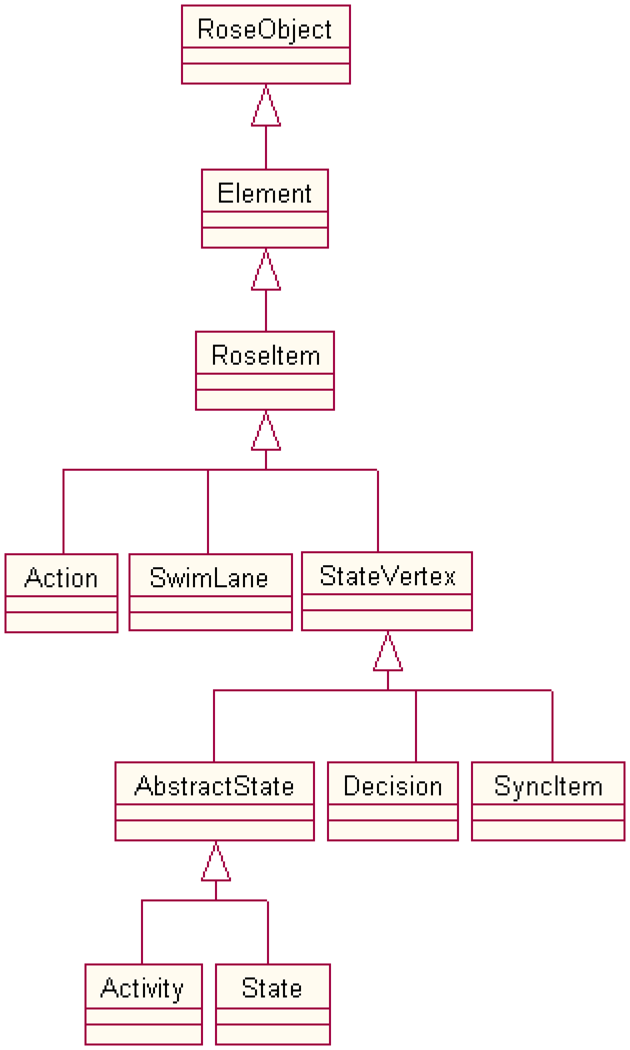

Figure 16 State Classes

RoseItemView Inheritance Diagram

Figure 17 shows the classes that inherit from the RoseItemView class in the Rational Rose Extensibility Interface.

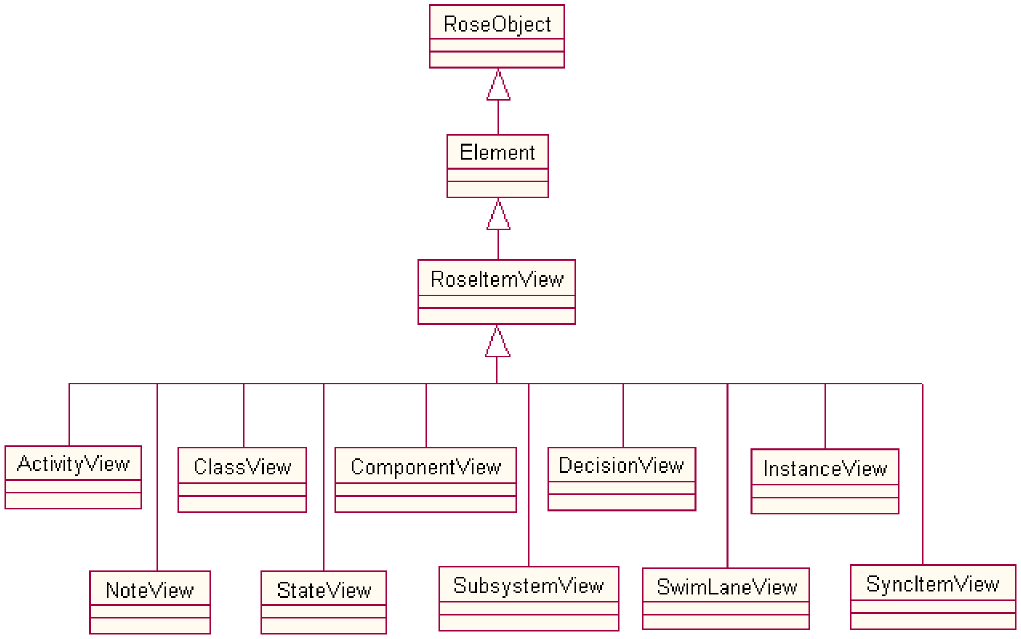

Figure 17 RoseItemView Inheritance Diagram

SecondaryRoseObject Inheritance Diagram

Figure 18 shows the classes that inherit from the SecondaryRoseObject class in the Rational Rose Extensibility Interface.

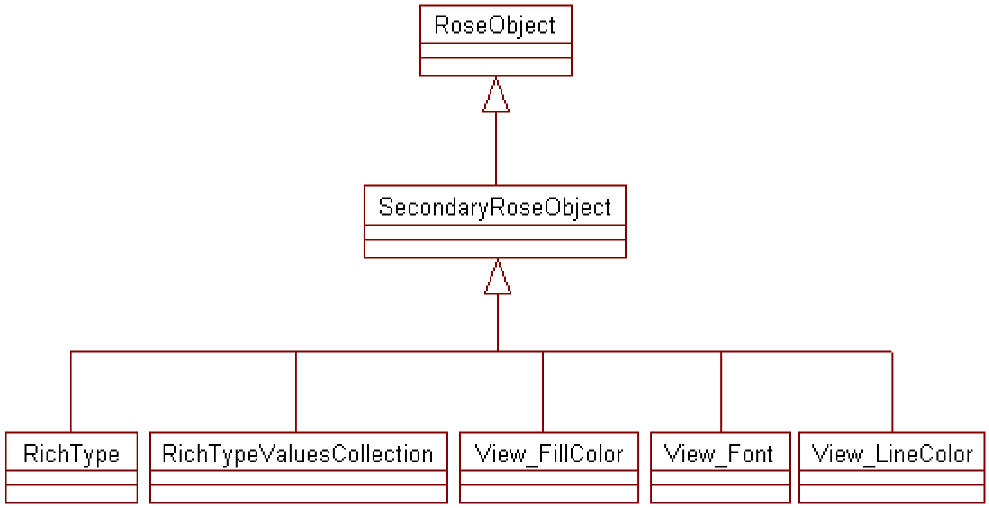

Figure 18 SecondaryRoseObject Inheritance Diagram

Appendix B

REI Events

Introduction

This appendix is provided to give additional information for customers wanting to explore the use of events and add-ins. However, events and creation of add-ins are not directly supported by Rational Technical Support. Additional support for events and add-ins is available through the Rational Unified Solutions Partner Program and Rational University.

For more information on the Rational Unified Solutions Partner Program see:

http://www.rational.com/corpinfo/partners/

For training on Rose's REI and add-ins see the "Extending Rational Rose" course from Rational University:

http://www.rational.com/university/description/

OnActivate Event

Description

The **OnActivate** Event is generated:

- every time Rose is started. It is fired to all add-ins whose “Active” registry setting is set to Yes.
- whenever a user activates an add-in through the Add-In Manager in the GUI (**Add-Ins > Add-In Manager**) or REI (RoseAddIn.Activate). In this case, this event is only fired to the individual add-in or add-ins that are activated.

Responding to this event gives your add-in the opportunity to execute any initializations, including shortcut menu item setup.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

Syntax

`OnActivate(``objRoseApp`**`)`**

Element	**Description**
objRoseApp As Application	Rose application in which add-ins are being activated

No return value is necessary.

See Also

OnDeactivate Event

OnEnableContextMenuItems Event

For more information, see page 575.

OnAppInit Event

Description

The **OnAppInit** event is generated whenever a new copy of the Rose application is started. It is fired to all active add-ins that are registered for it.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

Syntax

`OnAppInit(``*objRoseApp*`)`

Element	**Description**
objRoseApp As Application	Rose application which is being started

No return value is necessary.

OnBrowseBody Event

Description

The **OnBrowseBody** Event is generated whenever a user browses a body file in Rose. In other words, any time the user presses `CTRL-K` or clicks the **Browse Code** menu item, Rose fires the **OnBrowseBody** event to all active add-ins that are registered for this event. It should be used to trigger the add-in to display the source module containing the class body. Only language add-ins should register for this event.

For more information, see page 575.

Rose fires the **OnBrowseBody** event to the language add-in of the selection set, if that language add-in supports this event. If more than one language is detected in the selection set, Rose displays an error message indicating that Rose does not support simultaneous browse code operations of multiple languages.

Associated Rose error messages that your user may see:

```
Multiple languages have been detected in the selection set.
Rational Rose dose not support simultaneous browse code
operations of multiple languages.

Browse code cannot be performed. The language add-in
(<language>) is not currently active.

Browse code cannot be performed. Nothing is selected.

Browse code cannot be performed. There is no assigned language
in the selection set.
```

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

Syntax

OnBrowseBody(*objRoseApp*)

Element	Description
objRoseApp As Application	Rose application in which the user browsed a body file

No return value is necessary.

See Also

OnBrowseHeader Event

OnGenerateCode Event

For more information, see page 575.

OnBrowseHeader Event

Description

The **OnBrowseHeader** Event is generated whenever a user browses a header file in Rose. In other words, any time the user presses **CTRL-H** or clicks the **Browse Header** menu item, Rose fires the **OnBrowseHeader** event to all active add-ins that are registered for this event. It should be used to trigger your add-in to display the source module containing the class header. Only language add-ins should register for this event.

Rose fires the **OnBrowseHeader** event to the language add-in of the selection set, if that language add-in supports this event. If more than one language is detected in the selection set, Rose displays an error message indicating that Rose does not support simultaneous browse header operations of multiple languages.

Associated Rose error messages that your user may see:

```
Multiple languages have been detected in the selection set.
Rational Rose dose not support simultaneous browse header
operations of multiple languages.

Browse header cannot be performed. The language add-in
(<language>) is not currently active.

Browse header cannot be performed. Nothing is selected.

Browse header cannot be performed. There is no assigned
language in the selection set.
```

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

For more information, see page 575.

Syntax

```
OnBrowseHeader(objRoseApp)
```

Element	Description
objRoseApp As Application	Rose application in which the user browsed a header file

No return value is necessary.

See Also

OnBrowseBody Event

OnGenerateCode Event

OnCancelModel Event

Description

The **OnCancelModel** event is generated whenever a user clicks the Cancel button on the Save Changes dialog, thus returning to the changed Rose model without saving or aborting the changes. This event is sent to all add-ins, language and non-language, registered for the event.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE Server that includes a method with the correct signature for this event.

For more information, see page 575.

Syntax

`OnCancelModel``(`*`objRoseApp, objUnit`*`)`

Element	**Description**
objRoseApp As Application	Rose application in which the closing of a changed model is being canceled
objUnit As ControllableUnit	The controllable unit whose changes are being canceled

No return value is necessary.

See Also

OnCloseModel Event

OnNewModel Event

OnOpenModel Event

OnSaveModel Event

OnCloseModel Event

Description

The **OnCloseModel** event is generated whenever a user closes a model in Rose. This event is sent to all add-ins, language and non-language, registered for the event.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

For more information, see page 575.

Syntax

`OnCloseModel(objRoseApp)`

Element	Description
objRoseApp As Application	Rose application in which a model is being closed

No return value is necessary.

See Also

OnCancelModel Event

OnNewModel Event

OnOpenModel Event

OnSaveModel Event

OnDeactivate Event

Description

The **OnDeactivate** Event is generated:

- every time Rose is shut down. It is fired to all active add-ins.
- whenever a user deactivates an add-in through the Add-In Manager in the GUI (**Add-Ins > Add-In Manager**) or REI (RoseAddIn.Deactivate). In this case the event is only fired to the individual add-in or add-ins that are deactivated.

Note: *When an add-in is deactivated via the Add-In Manager dialog box, the add-in remains deactivated until the user activates it again via the Add-In Manager dialog box.*

Responding to this event gives your add-in the opportunity to perform any "clean-up" before your add-in's COM server exits. For example, your add-in could respond to this event by freeing up references using Set myObj = Nothing.

For more information, see page 575.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

Syntax

`OnDeactivate(`*`objRoseApp`***`)`**

Element	**Description**
objRoseApp As Application	Rose application in which add-ins are being deactivated

No return value is necessary.

See Also

OnActivate Event

OnDeletedModelElement Event

Description

The **OnDeletedModelElement** event is generated whenever a user deletes a model element in Rose.

- If the deleted model element has an assigned language, Rose sends the **OnDeletedModelElement** event to that language add-in and all non-language add-ins registered for this event.
- If the deleted model element has no assigned language, Rose sends the **OnDeletedModelElement** event to all non-language add-ins registered for this event.

Warning: *Do not delete model elements when you receive the* ***OnDeletedModelElement*** *event. Otherwise you will be in an infinite loop and Rose will look like it has stopped functioning.*

For more information, see page 575.

Warning: *Do not display a dialog in response to this event. This keeps Rose hanging until your user dismisses your dialog. This is especially bad if a user deletes several model elements at once—users would have to dismiss your dialog box several times.*

Warning: *Return control to Rose as quickly as possible. One way to do this is to spawn another process then return control to Rose.*

Filtering of the **OnDeletedModelElement** event can take place quickly on the add-in side by checking the *objItem* parameter. For example, if your add-in is interested only in use cases, the add-in can check to see if *objItem* is a use case. If it is not, simply return.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE Server that includes a method with the correct signature for this event.

Syntax

`OnDeletedModelElement``(`*`objRoseApp`*`, `*`objItem`*`)`

Element	**Description**
objRoseApp As Application	Rose application in which model elements are being deleted
objItem As RoseItem	Model element being deleted

No return value is necessary.

See Also

OnModifiedModelElement Event

OnNewModelElement Event

For more information, see page 575.

OnEnableContextMenuItems Event

Description

The **OnEnableContextMenuItems** event is generated when shortcut menu items are enabled in Rose. When the user right-clicks to display the shortcut menu, Rose fires this event to each active add-in registered for it. In response to this event, your add-in should conditionalize (disable/enable) its shortcut menu items, if desired.

If you are not going to conditionalize your menu items, that is, they will always be enabled, you do not need to use this event.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE Server that includes a method with the correct signature for this event.

Syntax

`OnEnableContextMenuItems(`*`objRoseApp, intItemType`***`) As BOOLEAN`**

Element	**Description**
objRoseApp As Application	Rose application in which the shortcut menu is being enabled
intItemType As Integer	Indicates the type of shortcut menu items being enabled. See *ContextMenuItemType Enum Values* for a list of these types

Return Type

Return TRUE or FALSE. For this event it does not matter which value you choose to return.

For more information, see page 575.

See Also

OnActivate Event

OnSelectedContextMenuItem Event

Creating Events for Shortcut Menus in the *Rational Rose 2000e Extensibility User Guide*

OnGenerateCode Event

Description

The **OnGenerateCode** event is generated whenever a user generates code in Rose. In other words, any time the user presses CTRL-G or clicks any **Generate Code** menu item, Rose fires the **OnGenerateCode** event to all active add-ins that are registered for this event. It should be used to trigger the add-in to generate code for the applicable class, classes, component or components. Only language add-ins should register for this event.

Rose fires the **OnGenerateCode** event to the language add-in of the selection set, if that language add-in supports this event. If more than one language is detected in the selection set, Rose displays an error message indicating that Rose does not support simultaneous code generation of multiple languages.

Associated Rose error messages that your user may see:

```
Multiple languages have been detected in the selection set.
Rational Rose dose not support simultaneous code generation of
multiple languages.

Code generation cannot be performed. The language add-in
(<language>) is not currently active.

Code generation cannot be performed. Nothing is selected.

Code generation cannot be performed. There is no assigned
language in the selection set.
```

For more information, see page 575.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

Syntax

`OnGenerateCode(``objRoseApp)`

Element	**Description**
objRoseApp As Application	Rose application in which code is being generated

No return value is necessary.

See Also

OnBrowseBody Event

OnBrowseHeader Event

OnModifiedModelElement Event

Description

The **OnModifiedModelElement** event is generated in certain cases when a user modifies a model element in Rose.

Assigned Language Changed:

- Rose sends the **OnModifiedModelElement** event, with the reason indicating a language change, to the language add-in corresponding to the *old* language, providing the language add-in is registered for **OnModifiedModelElement**. If the *old* language is "Analysis" (that is, there is no assigned language), Rose sends this event to all non-language add-ins that are registered for it.

For more information, see page 575.

- Rose also sends the **OnModifiedModelElement** event, with the reason indicating a language change, to the language add-in corresponding to the *new* language, providing the language add-in is registered for **OnModifiedModelElement**. If the *new* language is "Analysis", Rose sends this event to all non-language add-ins that are registered for it.
- If the language of the item did *not* change from or to "Analysis", Rose sends the **OnModifiedModelElement** event with the reason indicating a language change, to all non-language add-ins that are registered for it.

Name Changed:

- Rose sends the **OnModifiedModelElement** event, with the reason indicating a name change, to all non-language add-ins registered for the event.

Warning: *Do not modify model elements when you receive the* ***OnModifiedModelElement*** *event. Otherwise you will be in an infinite loop and Rose will look like it has stopped functioning.*

Warning: *Do not display a dialog in response to this event. This keeps Rose hanging until your user dismisses your dialog. This is especially bad if a user modifies several model elements at once—users would have to dismiss your dialog several times.*

Warning: *Return control to Rose as quickly as possible. One way to do this is to spawn another process then return control to Rose.*

Filtering of the **OnModifiedModelElement** event can take place quickly on the add-in side by checking the *objItem* and *intReason* parameters. For example, if your add-in is interested only in use case name changes, the add-in can check to see if *objItem* is a use case. If it is not, simply return. If *objItem* is a use case, but the value of the *intReason* parameter is not 1 (for Name Changed), simply return.

We recommend that you only use this event as a way to log changes during a Rose session. Then, when the user saves their model, have your add-in read your log and make changes (for example, code generation). Why wait to execute your add-in's functionality until the

For more information, see page 575.

user saves their model? Because saving the model is the point at which the user actually commits to the changes they made during their Rose session. If your add-in takes action sooner and the user changes their mind by closing the model without saving changes, your add-in has to undo everything it did based on those abandoned changes.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE Server that includes a method with the correct signature for this event.

Syntax

`OnModifiedModelElement(`*`objRoseApp`*`, `*`objItem`*`, `*`intReason`*`)`**

Element	**Description**
objRoseApp As Application	Rose application in which model elements are being modified
objItem As RoseItem	Modified model element
intReason As Integer	Integer value indicating what modification was made to the model element. Valid values are: 1 = Name Changed 2 = Language Changed

No return value is necessary.

See Also

OnDeletedModelElement Event

OnNewModelElement Event

For more information, see page 575.

OnNewModel Event

Description

The **OnNewModel** event is generated whenever a user creates a new model in Rose, either via the GUI (click **File > New** or the Create New Model toolbar button) or REI (Application.NewModel). This event is fired to all active add-ins that are registered for it.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

Syntax

`OnNewModel(`*`objRoseApp`*`)`

Element	**Description**
objRoseApp As Application	Rose application in which models are being created

No return value is necessary.

See Also

OnCancelModel Event

OnCloseModel Event

OnOpenModel Event

OnSaveModel Event

For more information, see page 575.

OnNewModelElement Event

Description

The **OnNewModelElement** event is generated whenever a user creates a new model element in Rose (including pasting a cut or copied model element, or undoing a deleted model element).

- If the new model element has an assigned language, Rose sends the **OnNewModelElement** event to that language add-in and all non-language add-ins registered for this event.
- If the new model element has no assigned language, Rose sends the **OnNewModelElement** event to all non-language add-ins registered for this event.

This event is also sent when changing the assigned language of a model element. The **OnNewModelElement** event is sent to the language add-in corresponding to the *new* language. This notifies the language add-in that it needs to add this model element to its generated code.

Warning: *Do not create new model elements when you receive the* ***OnNewModelElement*** *event. Otherwise you will be in an infinite loop and Rose will look like it has stopped functioning.*

Warning: *Do not display a dialog in response to this event. This keeps Rose hanging until your user dismisses your dialog box. This is especially bad if a user adds several model elements at once (for example, by undoing a delete of several model elements)—users would have to dismiss your dialog box several times.*

Warning: *Return control to Rose as quickly as possible. One way to do this is to spawn another process then return control to Rose.*

Filtering of the **OnNewModelElement** event can take place quickly on the add-in side by checking the *objItem* parameter. For example, if your add-in is interested only in use cases, the add-in can check to see if *objItem* is a use case. If it is not, simply return.

For more information, see page 575.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE Server that includes a method with the correct signature for this event.

Syntax

`OnNewModelElement``(`*`objRoseApp`*`, `*`objItem`*`)`

Element	**Description**
objRoseApp As Application	Rose application in which new model elements are being created
objItem As RoseItem	Newly created model element

No return value is necessary.

See Also

OnDeletedModelElement Event

OnModifiedModelElement Event

OnOpenModel Event

Description

The **OnOpenModel** event is generated whenever a user opens a model in Rose. This event is sent to all add-ins, language and non-language, registered for the event.

Registry and Server Requirements

An add-in needs to add registry entries to register for this event. The registry entry indicates whether the event is an Interface event or a Script event.

For more information, see page 575.

Syntax

`OnOpenModel(`*`objRoseApp`*`)`

Element	Description
objRoseApp As Application	Rose application in which a model is being opened

No return value is necessary.

See Also

OnCancelModel Event

OnCloseModel Event

OnNewModel Event

OnSaveModel Event

OnPropertySpecOpen Event

Description

The **OnPropertySpecOpen** event is generated whenever a user opens a specification in Rose directly from the browser or diagram. If the selected item has an assigned language and its language add-in is registered for this event, Rose fires **OnPropertySpecOpen** every time a specification is opened. In other words, it does not matter if your user double-clicks, selects an **Open Specification** menu option, presses F4 or presses CTRL-B, Rose fires the **OnPropertySpecOpen** event in each of those cases.

This event applies only to language add-ins. Otherwise Rose could display multiple specification dialogs at once.

Note: *If different types of model elements are selected, F4 and CTRL-B do not cause Rose to fire the* ***OnPropertySpecOpen*** *event. Otherwise, your user could have several different specification dialogs to deal with at the same time (one for each type of selected model element).*

For more information, see page 575.

Also note, if you are already displaying the standard Rose specification for a model element and your user displays the specification for a model element from that specification, Rose does not fire the **OnPropertySpecOpen** event. For example, if you have designed and implemented a custom specification for attributes and your user displays the standard Rose specification for a class, then displays the specification for one of the attributes on the class specification, Rose displays the standard attribute specification, not your custom attribute specification.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE Server that includes a method with the correct signature for this event.

Syntax

`OnPropertySpecOpen(``objRoseApp, objItem`**`) AS BOOLEAN`**

Element	**Description**
objRoseApp As Application	Rose application in which a specification is being opened
objItem As RoseItem	Model element for which a specification is being opened

Return Type

To display your custom specification for the specified model element, return TRUE.

If you do not have a custom specification for the specified model element, return FALSE to display the standard Rose specification.

See Also

Shortcut Menu Design Considerations in the *Rational Rose 2000e Extensibility User Guide*

For more information, see page 575.

OnSaveModel Event

Description

The **OnSaveModel** event is generated whenever a user saves a model in Rose. This event is sent to all add-ins, language and non-language, registered for the event.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE server that includes a method with the correct signature for this event.

Syntax

`OnSaveModel(``objRoseApp, objUnit, blnIgnoreSubUnits)`

Element	**Description**
objRoseApp As Application	Rose application in which a model is being saved
objUnit As ControllableUnit	The controllable unit whose changes are being saved
blnIgnoreSubUnits As Boolean	Flag to indicate whether or not the user saved changes in subunits of *objUnit* ■ If the user chose to save changes in subunits of the specified *objUnit*, Rose sets *blnIgnoreSubUnits* to FALSE. Therefore, in addition to any processing of *objUnit*, your add-in also needs to process any subunits of *objUnit*. ■ If the user chose not to save changes (ignore them) in subunits of the specified *objUnit*, Rose sets *blnIgnoreSubUnits* to TRUE. Your add-in can then ignore subunits of *objUnit* and needs to deal only with *objUnit*.

No return value is necessary.

For more information, see page 575.

See Also

OnCancelModel Event

OnCloseModel Event

OnNewModel Event

OnOpenModel Event

OnSelectedContextMenuItem Event

Description

The **OnSelectedContextMenuItem** event is generated whenever a user selects a menu item from a shortcut menu in Rose. In response to this event, your add-in should execute the appropriate action corresponding to the selected shortcut menu item.

Registry and Server Requirements

You do not need to add registry entries to register for this event. To receive this event, however, you must provide an OLE server that includes a method with the correct signature for this event.

Syntax

```
OnSelectedContextMenuItem(objRoseApp, strInternalName) AS
      BOOLEAN
```

Element	Description
objRoseApp As Application	Rose application in which a shortcut menu item is being selected
strInternalName As String	The internal name defined for the shortcut menu item by the RoseAddIn.AddContextMenuItem method. The internal name is your mapping from the selected shortcut menu item to your add-in's corresponding functionality

For more information, see page 575.

Return Type

Return TRUE or FALSE. For this event it does not matter which value you choose to return.

See Also

OnEnableContextMenuItems Event

Creating Events for Shortcut Menus in the *Rational Rose 2000e Extensibility User Guide*

For more information, see page 575.

Index

A

Abstract 82, 544
abstract states
    actions 8, 10
    activities 10
    decisions 3, 11
    do actions 6, 12
    entry actions 6, 13
    exit actions 7, 13
    retrieving 482
    retrieving all 499
    state machines 7, 9, 14, 482, 499
    states 3, 11
    subactivities 2
    synchronizations 4, 12
    user-defined events 8, 9, 14
abstract, determining if or setting 544
AbstractState Class 1
AbstractState Class Methods 5
AbstractState Class Properties 2
AbstractStates 482
access adornments, visibility of 139
Action Class 15
Action Class Methods 16
Action Class Properties 15
actions
    abstract states 8, 10
    adding 6, 7
    arguments 15
actions, *continued*
    deleting 8
    retrieving 10, 12, 13, 236, 539
    send 539
    targets 16
    transitions 539, 542
    triggers 542
Activate 207, 395
activating add-ins event 576
Active 84
active
    add-ins 399
    diagram 213
Activities 483
activities
    abstract states 2, 10
    activity views 19, 462
    adding 492
    deleting 498
    retrieving 2, 19, 482, 483, 487, 510, 533, 540, 541
    retrieving all 10, 499, 501
    retrieving selected 466, 469
    state diagrams 466
    state machines 483, 492, 499
Activity Class 17
Activity Class Methods 18
Activity Class Properties 17

activity diagrams
    adding 492
    determining if 455
    retrieving all 500
    state machines 492
activity views
    activities 19, 462
    adding 458
    removing 476
    retrieving 462, 464, 465, 472
    retrieving selected 467
    state diagrams 458, 462, 464, 467, 476
ActivityView Class 18
ActivityView Class Methods 19
ActivityView Class Properties 18
Add 152
AddActivity 492
AddActivityDiagram 492
AddActivityView 458
AddAssignedModule 92
AddAssociation 92, 124, 549
AddAttachmentToView 329
AddAttribute 93
AddCategory 65, 124
AddCategoryDependency 65
AddClass 66, 125
AddClassDependency 93
AddClassDiagram 66, 550
AddCollection 152
AddComponentView 317
AddContextMenuItem 395
AddDataModelDiagram 67
AddDecision 493
AddDecisionView 459
AddDefaultProperty 179
AddDevice 192, 286
AddDeviceConnection 199, 364
AddDoAction 6
AddEntry 352
AddEntryAction 6
AddExitAction 7
AddExternalDocument 208, 409, 494
AddHas 94
add-in manager
    add-ins 402
    applications 21
    events
        disabling 403
        enabling 404
    retrieving 21
add-in names, retrieving 391
AddInheritRel 94, 550
AddInManager 21
AddIns 402
add-ins 387
    activating 395
    activating event 576
    add-in manager 402
    checking if active 399
    checking if language add-in 400
    company name 388
    contacting support 575
    context menu items
        adding 395
        retrieving 398
    copyright 388
    deactivating 397
    deactivating event 582
    display name 399
    event handler 389
    executing scripts 397
    fundamental types 389
    help file paths 390
    installation directory 390
    menu file path 391
    name 391
    property file path 392
    Rational Unified Solutions Partner Program 575
    registry settings
        creating 401
        retrieving 400
    retrieving 402

add-ins, *continued*
  root registry paths 392
  server name 393
  shortcut menu items
    adding 395
    retrieving 398
  support 575
  technical support 575
  tool names 393
  training 575
  versions 394
AddInstance 442
AddInstanceView 443
AddInstantiateRel 95
AddKey 386
AddLink 335
AddMessageTo 264
AddModule 520
AddModuleDiagram 521
AddNestedClass 95
AddNoteView 209
AddOperation 96
AddParameter 97, 345
AddProcess 364
AddProcessor 193, 286
AddProcessorConnection 199, 365
AddRealizeRel 99, 309
AddRelationView 209
AddScenarioDiagram 68, 551
AddState 452, 495
AddStateChartDiagram 495
AddStateMachine 7
AddStateView 459
AddSubsystem 521
AddSubsystemView 318
AddSwimLane 496
AddSwimLaneView 460
AddSynchronization 496
AddSynchronizationView 461
AddTransition 512
AddTransitionToVertex 513
AddUseCase 68, 125
AddUseCaseDiagram 552
AddUserDefinedEvent 8
AddVisibilityRelationship 310
Aggregate 378
aggregation 244
  determining or setting 378
AllClasses 83
Application 218
Application Class 20
Application Class Methods 27
Application Class Properties 20
application paths
  applications 22
  retrieving 22
application window 20
ApplicationPath 22
applications
  add-in manager 21
  application paths 22
  command line 22
  current model 23
  determining if initialized 23
  error log, writing 43
  exiting 30
  external documents, opening 37
  height 23
  left 24
  models 23, 42, 43
    creating 36
    opening 38, 40
    opening as template 39
  objects, retrieving 32
  pathmaps 24
    adding entries 352
    deleting entries 352
  paths 22, 24
  product names 25
  profile strings
    retrieving 33
    writing 44
  retrieving 218
  retrieving, licensed 32

applications, *continued*
- Rose.ini path 34
- saving 42
- saving as 43
- scripts
  - creating (RoseScript only) 36
  - loading (RoseScript only) 35
  - opening (RoseScript only) 41
- top 25
- URLs, opening 41
- versions 26
- visible 26
- width 27

applications (Rose), initializing event 577
Arguments 15, 235
arguments
- actions 15
- events 235
- retrieving or setting 15, 235

AssignAssociation 265
assigned language
- classes 106
- retrieving 106

AssignedLanguage 306
associate items, retrieving 379
associate roles
- classes 107
- retrieving 107

AssociateItem 379
Association 379
Association Class 45
Association Class Methods 49
Association Class Properties 45
Associations 59
associations 258
- adding 92, 549
- adding to diagram 124
- assigning to links 265
- categories 47
- class diagrams 124
- classes 92, 100, 110
- constraints 46

associations, *continued*
- corresponding roles, retrieving 50
- deleting 100, 553
- derived 46
- line vertices 253
- link class names, setting 52
- link classes 46, 114
- links 266
- name directional role, checking for 51
- name directions
  - clearing 49
  - retrieving roles 51
  - setting 52
- other roles, retrieving 50
- packages 47
- relocating 75
- removing from
  - diagrams 131
  - links 267
- retrieving 59, 110, 114, 266, 379, 555
- retrieving from diagrams 126
- role 1 47
- role 2 48
- roles 48, 379
- use cases 549, 553, 555

asynchronous 271
attachments, adding to note views 329
Attribute Class 53
Attribute Class Methods 57
Attribute Class Properties 53
Attributes 82
attributes
- adding 93
- adding as role keys 386
- classes 93, 100
- containment 54
- deleting 100
- deleting as role keys 385
- derived 54
- displaying or hiding in diagrams 136

attributes, *continued*
- export control 55
- friend 384
- initial value 56
- keys 383
- parent class 56
- retrieving 82, 140, 383
- roles 383, 384
- setting as static 56
- static 56
- suppressing display in diagrams 138
- type 57

automatic resize, setting 136
AutomaticResize 136

B

balking 271
blue
- fill color 417
- font 419
- lines 424

Body 305
body, browsing event 577
bold 420
Booch 281, 326
BoochNotation 326
browsing body event 577
browsing header event 579
ByReference 54, 245, 381
ByValue 54, 245, 381

C

canceling model changes event 580
CanTypeCast 435
Caption 158
captions, retrieving or setting 158
Cardinality 82, 379
cardinality
- client
  - class dependencies 119
  - has relationships 244
- determining 247
- retrieving 82, 119, 121, 244, 379
- roles 379
- setting 82, 119, 121, 244, 247, 379
- supplier
  - class dependencies 121
  - has relationships 247

Categories 59
categories 347
- adding 65
- adding to diagrams 124
- associations 47, 59, 75
- category dependencies
  - adding 65
  - deleting 69
  - retrieving 74
- checking if root 348
- child categories 69, 72
- class diagrams 60, 66, 70, 76, 122
- classes 60, 66, 70, 72, 76
  - retrieving all 292
- deleting 69
- determining if global 61
- external documents 238
- global 61
- parent category 61
- relocating 75
- removing from diagrams 131
- retrieving 47, 59, 122, 238, 288, 292, 524, 545
  - all 72
  - by unique id 288
  - from diagrams 126, 128
  - parent 61, 87
  - selected 296
- root 78, 281
- scenario diagrams 62, 68, 71, 77
- setting as global 61

categories, *continued*
subcategories 59
subsystems 73, 74, 77, 524
top level 78
use cases 62, 68, 71, 73, 545
Category Class 58
Category Class Methods 63
Category Class Properties 58
category dependencies
adding 65
client categories, retrieving 79
deleting 69
retrieving 74
supplier categories, retrieving 80
CategoryDependency Class 78
CategoryDependency Class Methods 79
CategoryDependency Class Properties 78
center of item views 416
Characteristics 156, 197, 360
characteristics
of connections 156
of processors 360
retrieving or setting 156, 360
checked 160, 268
Class 380
Class Class 80
Class Class Methods 89
Class Class Properties 81
class dependencies 133
adding 93
cardinality 119, 121
classes 93, 101, 111
creating 93
deleting 101
export control 120
friendship 121
retrieving 111
class diagrams
adding 66, 550
associations 124
categories 124
classes 125
packages 124
use cases 125
categories 122
checking for use case diagrams 130
deleting 70, 554
packages 122
relocating 76
removing
associations 131
categories 131
classes 132
packages 131
use cases 132
retrieving 60, 544
associations 126
categories 126
class views 127
classes 127, 129
OLE object (RoseScript only) 128
packages 126
selected categories 128
selected packages 128
use cases 129
use cases 543, 544, 550, 554
class kind, retrieving or setting 83
class relations 374
retrieving client classes 134
retrieving supplier classes 134
class views
access adornments 139
retrieving attributes 140
retrieving automatic resize setting 136
retrieving from diagrams 127
ClassDependency Class 119
ClassDependency Class Methods 121
ClassDependency Class Properties 119

ClassDiagram Class 122
ClassDiagram Class Methods 123
ClassDiagram Class Properties 122
ClassDiagrams 60, 544
Classes 60
classes
  abstract 82
  adding 66
  adding to diagrams 125
  assigned language 106
  associate roles 107
  associations 92, 100, 110, 114
  attributes 56, 82, 93, 100
  cardinality 82
  characteristics 53
  checking if class 337
  class dependencies 93, 101, 111
  class kind 83
  class relations 134
  client classes 112
  concurrency 84
  deleting 70
  determining if abstract 82
  determining if fundamental type 86
  export control 85
  fundamental type 86
  get all 72
  has relationships 94, 101, 113
  inherit relations 94, 102, 113
  instantiated relations 95, 102, 114
  link 46, 117
  modules 92, 107, 118, 312
  nested 95, 103, 115, 118
  object instances 333
  objects 437, 438
  operations 86, 96, 103, 341
  parameters 87, 97, 104
  parent category 87
  parent class 88
  parent package 87
  persistence 88
  properties 82
classes, *continued*
  realize relations 99, 105, 115, 372, 373
  relocating 76
  removing from diagrams 132
  retrieving 56, 60, 116, 117, 289, 341, 380, 486, 524
    all 292
    by unique id 289
    names 226, 386
    selected 296
  retrieving from
    diagrams 127, 129
    object instances 336
  retrieving nested 106
  retrieving parent 88
  roles 116, 378, 380
  setting as abstract 82
  setting as fundamental type 86
  space algorithms, retrieving 88
  state machines 89, 99, 105, 486
  subclasses 116
  subsystems 524
  superclasses 117
ClassKind 83
ClassName 333
ClassRelation Class 133
ClassRelation Class Methods 133
ClassRelation Class Properties 133
ClassView Class 135
ClassView Class Methods 139
ClassView Class Properties 135
ClearRoleForNameDirection 49
ClientCardinality 119, 244
ClientRelKind 142
ClientRelType 143
clients
  cardinality of has relationships 244
  relations 375, 376
  retrieving 375
  retrieving classes 112

Clipboard
- diagrams 216
- rendering icons 233, 243

CloneDefaultPropertySet 180
closing model event 581
code generating event 586
code generation 252, 369
Collaboration Diagram 446, 551
Collection Class Methods 146
Collection Class Methods, User-Defined 151
Collection Class Properties 145
Collection Classes 144
collection count, retrieving 145
collections, objects 147, 148, 149, 150, 151, 152, 153
colors
- determining or setting 416
- fills 417, 418, 419
- fonts 419, 421, 422
- lines 424, 425

command line
- applications 22
- retrieving 22

CommandLine 22
company names
- add-ins 388
- retrieving 388

CompanyName 388
CompileScriptFile 29
component diagrams
- adding component views 317
- adding subsystem views 318
- removing component views 321
- removing subsystem views 322
- retrieving component views 314
- retrieving subsystem views 315

Component View 282
component views
- adding 317
- components 155
- modules 155

component views, *continued*
- removing 321
- retrieving 282, 314

components
- realize relations 372, 373
- retrieving 155

ComponentView Class 153
ComponentView Class Methods 154
ComponentView Class Properties 154
ComponentViews 314
Concurrency 84, 339
concurrency, determining or setting 84, 339
configuration management 165, 173, 203
connection relations
- See connections

ConnectionRelation Class 155
ConnectionRelation Class Methods 157
ConnectionRelation Class Properties 156
Connections 198, 361
connections
- characteristics 156
- devices
  - adding 364
  - creating 199
  - removing 201
- overview 155
- processors
  - adding 365
  - creating 199
  - removing 201
- retrieving 198, 361
- supplier type 157

Const 349
constant
- determining if or setting 349
- parameters 349

Constraints 46, 380

constraints
- associations 46
- retrieving or setting 46, 380
- roles 380

Containment 54, 245, 381
containment 244
- has relationships 245
- retrieving 54, 245, 381
- roles 381
- setting 54, 245, 381

context menu items
- caption 158
- enabling event 585
- internal name 159
- menu id 159
- menu state 160
- selecting event 596

context menus 158, 159, 160, 268, 395
context modules, retrieving 323
context subsystems, retrieving 324
ContextMenuItem Class 158
ContextMenuItem Class Methods 160
ContextMenuItem Class Properties 158
ContextMenuItemType 161
ContextModule 323
ContextSubsystem 324
Control 165
controllable units
- checking if
  - controlled 167
  - loaded 167
  - locked 168
  - modifiable 168
  - modified 169
- controlling 165
- file names 165
- loading 169, 301
- locking 170
- refreshing 171
- retrieving file names 166
- saving 172
- saving as 172

controllable units, *continued*
- setting modifiability 170
- subunits
  - retrieving all 165
  - retrieving immediate 166
- uncontrolling 173
- unloading 173
- unlocking 174
- updating 171

ControllableUnit Class 162
ControllableUnit Class Methods 163
ControllableUnit Class Properties 163
coordinates
- item views 432
- retrieving or setting 416

Copyright 388
copyrights, retrieving 388
Count 145
CreateDefaultPropertySet 181
CreateMessage 444
CreateObject 445
CreateProperty 221
CreateStateMachine 99, 507, 553
creating new model elements event 591
creating new models event 590
current model, applications 23
currently active diagram 213
CurrentModel 23
Cyclic 362

D

data model diagrams
- adding 67
- determining if 130

data types
- retrieving 57, 369
- setting 57

Deactivate 397
deactivating add-ins event 582
debugging 159
Decision Class 174

Decision Class Methods 175
Decision Class Properties 175
decision views
  adding 459
  decisions 463, 468
  removing 477
  retrieving 463, 464, 465, 472
  retrieving selected 467
  state diagrams 459, 463, 464, 467, 477
Decisions 483
decisions
  abstract states 3, 11
  adding 493
  decision views 463, 468
  deleting 498
  retrieving 3, 483, 487, 510, 533, 540, 541
  retrieving all 11, 500, 501
  retrieving selected 468, 469
  state diagrams 468
  state machines 483, 493, 500
DecisionView Class 175
DecisionView Class Methods 176
DecisionView Class Properties 176
Declarations 303
declarations, retrieving or setting 303
default languages, determining or setting 279
default properties
  adding 179
  checking tool visibility 188
  cloning 180
  creating 181
  deleting 182
  deleting sets 183
  finding 184
  retrieving 222
    set names 186, 225
    sets 185
    tool names 187
    value 224
default properties, *continued*
  setting tool visibility 188
DefaultLanguage 279
DefaultModelProperties Class 177
DefaultModelProperties Class Methods 178
DefaultModelProperties Class Properties 177
DefaultProperties 279
DeleteAction 8
DeleteAssociation 100, 553
DeleteAttribute 100
DeleteCategory 69
DeleteCategoryDependency 69
DeleteClass 70
DeleteClassDependency 101
DeleteClassDiagram 70, 554
DeleteDefaultProperty 182
DeleteDefaultPropertySet 183
DeleteDevice 287
DeleteEntry 352
DeleteExternalDocument 210, 410, 497
DeleteHas 101
DeleteInheritRel 102, 554
DeleteInstance 445
DeleteInstantiateRel 102
DeleteKey 385
DeleteLink 336
DeleteMessage 265
DeleteModule 522
DeleteNestedClass 103
DeleteOperation 103
DeleteParameter 104, 346
DeleteProcess 365
DeleteProcessor 287
DeleteRealizeRel 105, 310
DeleteScenarioDiagram 71, 555
DeleteState 453, 497
DeleteStateMachine 9, 105, 508
DeleteStateVertex 498
DeleteSubsystem 522
DeleteSwimLane 498

DeleteTransition 514
DeleteUseCase 71
DeleteUserDefinedEvent 9
DeleteVisibilityRelationship 311
deleting model elements event 583
dependencies
  category 78
  package 78
  retrieving 311, 312, 525
DependencyRelation Class 189
DependencyRelation Class Methods 190
DependencyRelation Class Properties 189
deployment diagrams 190
  deployment units 195, 280
  devices
    adding 192
    removing 194
    retrieving 193
  processors
    adding 193
    removing 195
    retrieving 194
  retrieving 280
deployment units, retrieving 280
DeploymentDiagram 280
DeploymentDiagram Class 190
DeploymentDiagram Class Methods 191
DeploymentDiagram Class Properties 190
DeploymentUnit 280
DeploymentUnit Class 195
DeploymentUnit Class Methods 196
DeploymentUnit Class Properties 196
Derived 46, 54
derived, determining or setting 46, 54
device characteristics, retrieving or setting 197
Device Class 196
Device Class Methods 198
Device Class Properties 197
device connections, adding 364
devices 190
  adding 192, 286
  characteristics 197
  connections
    creating 199
    removing 201
  deleting 287
  processors 366, 367
  removing 194
  removing connections 367
  retrieving 293
    connected 200, 366
    from diagram 193
Diagram Class 202
Diagram Class Methods 205
Diagram Class Properties 202
Diagrams 484
diagrams
  activating 207
  active
    checking if 213
    retrieving 291
  activity 492
  adding 66, 492, 495
  adding data model 67
  arranging 214
  class 202, 544
  component views 153
  deployment 190, 202
  determining if data model 130
  displaying
    attributes 136
    operations 137
    operations signature 137
  documentation 202
  external documents 203
    adding 208
    deleting 210
  hiding
    attributes 136
    operations 137
    operations signature 137

diagrams, *continued*
- invalidating 213
- item views 204, 214, 425
  - retrieving 212
  - selecting 433
- items 203
  - checking for 210
  - retrieving selected 212
- module 517
- note views
  - adding 209
  - removing 215
  - retrieving 211
- redrawing 213
- refreshing 213
- relation views, adding 209
- removing item views 214
- rendering 215
  - enhanced 216
  - to clipboard 216, 217
- retrieving 238, 290, 425, 517, 544, 546
  - active 291
  - components 155
  - modules 155
- retrieving all 500
- retrieving associated model element 211
- retrieving from note views 330
- retrieving selected 297
- scenario 202, 440, 546
- sizing icons 136
- state machines 500
- statechart 495
- suppressing
  - attribute display 138
  - operation display 138
- switching focus to 207
- updating 217
- visibility 204
- zoom factor 205

directories 390

disabled 160, 268

DisableEvents 403

displaying stereotypes 415

do actions
- abstract states 6, 12
- adding 6
- retrieving 12

Documentation 202, 405, 484

documentation
- generation 252
- items 405
- retrieving or setting 202, 405, 484
- state machines 484

E

Element Class 218

Element Class Methods 219

Element Class Properties 218

elements
- application 218
- classes, retrieving name 226
- default properties
  - retrieving 222
  - retrieving set names 225
  - retrieving value 224
- models 219
- name 219
- properties
  - checking if default value 230
  - checking if overridden 231
  - creating 221
  - overriding 232
  - restoring default value 229
  - retrieving 223, 228
  - retrieving active set 224
  - retrieving all 223
  - retrieving value 226
  - specifying current set 234
- qualified name, retrieving 227
- tools, retrieving names 228
- unique ids, retrieving 229

enabled 160, 268
EnableEvents 404
enabling shortcut (context) menu items event 585
entry actions
  abstract states 6, 13
  adding 6
  retrieving 13
entry behavior, operations 342
enumerations
  ClientRelKind 142
  ClientRelType 143
  ContextMenuItemType 161
  MenuState 268
  NotationTypes 326
  RoseAddInEventTypes 401
error log 20
  applications 43
  writing messages to 43
Event Class 234
Event Class Methods 236
Event Class Properties 235
event handlers
  add-ins 389
  retrieving 389
  server name 393
EventHandler 389
events (activities, states)
  actions, retrieving 236
  adding 8
  arguments 235
  deleting 9
  guard conditions 235
  names 236
  retrieving 14, 453
  states 453
  transitions 542
  triggers, retrieving 542
  user-defined 453
events (COM)
  activating add-ins 576
  browsing body 577
  browsing header 579
  canceling model changes 580
  closing models 581
  creating new model elements 591
  creating new models 590
  deactivating add-ins 582
  deleting model elements 583
  disabling 401, 403
  enabling 401, 404
  enabling shortcut (context) menu items 585
  generating code 586
  initializing Rose applications 577
  modifying model elements 587
  OnActivate 395
  opening models 592
  opening specifications 593
  saving models 595
  selecting shortcut (context) menu items 596
events (object instances) 440
Exceptions 339
exceptions, retrieving or setting 339
ExecuteScript 30, 397
Executive 362
Exists 147, 210
Exit 30
exit actions
  abstract states 7, 13
  adding 7
  retrieving 13
exit behavior, operations 342
export control
  attributes 55
  class dependencies 120
  classes 85
  has relationships 246
  inherit relations 249
  operations 340

export control, *continued*
- retrieving 55, 85, 120, 246, 249, 340, 382
- roles 382
- setting 55, 85, 120, 246, 249, 340, 382

ExportControl 55, 85, 120, 246, 249, 340, 382
Extending Rational Rose course 575
external documents
- adding 409, 494
- adding to diagrams 208
- applications 37
- categories 238
- deleting 410, 497
- deleting from diagrams 210
- diagrams 203
- items 406
- opening 37, 242
- packages 238
- paths 239
- retrieving 203, 406, 485
- retrieving parent 239
- retrieving parent diagrams 238
- retrieving, selected 297
- state machines 485, 494, 497
- URLs 240
  - checking for 241
  - opening 41

ExternalDocument Class 237
ExternalDocument Class Methods 240
ExternalDocument Class Properties 237
ExternalDocuments 203, 406, 485

F

face name, font 420
file names
- associating with controllable units 165
- retrieving for controllable units 166

fill color
- blue 417
- green 418
- item views 418, 419, 433
- red 418
- retrieving or setting 417, 418, 419
- transparent 419

FillColor.Blue 417
FillColor.Green 418
FillColor.Red 418
FillColor.Transparent 419
Final 451
FindCategories 288
FindCategoryWithID 288
FindClasses 289
FindClassWithID 289
FindDefaultProperty 184, 222
FindDiagramWithID 290
FindFirst 147
finding a particular object in a collection 145
FindItems 290
FindItemWithID 291
FindNext 148
FindProperty 223
focus 291
Font.Blue 419
Font.Bold 420
Font.FaceName 420
Font.Green 421
Font.Italic 421
Font.Red 422
Font.Size 422
Font.StrikeThrough 422
Font.Underline 423
fonts 421
- blue 419
- bold 420
- colors, retrieving or setting 419, 421, 422
- face names, retrieving or setting 420

fonts, *continued*
formatting, retrieving or setting 420, 421, 422, 423
green 421
item views 419, 420, 421, 422, 423
red 422
size, retrieving or setting 422
strikethrough 422
underline 423
FreeScript 31
Frequency 270
frequency, determining or setting 270
Friend 384
friend
attributes 384
operations 384
roles 384
friendship, determining or setting 121, 250, 384
FriendshipRequired 250
fundamental types
add-ins 389
retrieving 389
FundamentalType 86
FundamentalTypes 389

G

generating code event 586
Generic 305
GetAction 236
GetActions 10
GetActiveDiagram 291
GetActivity 19
GetActivityViews 462
GetActualPath 353
GetActualPathWithContext 353
GetAllAbstractStates 499
GetAllActivities 499
GetAllCategories 72, 292
GetAllClasses 72, 292
GetAllDecisions 500
GetAllDependencies 311
GetAllDevices 293
GetAllDiagrams 500
GetAllModules 293, 523
GetAllNestedClasses 106
GetAllProcessors 294
GetAllProperties 223
GetAllStates 501
GetAllStateVertices 501
GetAllSubActivities 10
GetAllSubDecisions 11
GetAllSubStates 11
GetAllSubSynchronizations 12
GetAllSubsystems 294, 523
GetAllSubUnitItems 165
GetAllSynchronizations 502
GetAllTransitions 502
GetAllUseCases 73, 295
GetAssignedCategories 524
GetAssignedClasses 312, 524
GetAssignedLanguage 106
GetAssignedModules 107
GetAssignedSubsystem 73
GetAssociateRoles 107, 108
GetAssociation 266
GetAssociations 110, 126, 555
GetAt 149
GetAttachedNotes 428
GetCategories 126
GetCategoryDependencies 74
GetClass 336
GetClassDependencies 111
GetClasses 127
GetClassName 386
GetClassView 127
GetClient 375
GetClients 112, 142, 143
GetComponent 155
GetConnectedDevices 200, 366
GetConnectedProcessors 200, 366
GetContextCategory 79
GetContextClass 134, 372

GetContextComponent 372
GetContextMenuItems 398
GetCorrespondingRole 50
GetCurrentPropertySetName 224
GetDecisionViews 463
GetDefaultHeight 429
GetDefaultPropertySet 185
GetDefaultPropertyValue 224
GetDefaultSetNames 186, 225
GetDefaultWidth 429
GetDependencies 312
GetDevices 193
GetDiagramActivityViews 464
GetDiagramDecisionViews 464
GetDiagramLink 330
GetDiagramStateVertexViews 465
GetDiagramSwimLaneViews 465
GetDiagramSynchronizationViews 466
GetDiagramType 446
GetDisplayedAttributes 140
GetDisplayName 399
GetDoActions 12
GetEntryActions 13
GetExitActions 13
GetFileName 166
GetFirst 149
GetHasRelations 113
GetIconIndex 225, 241
GetInheritRelations 113, 556
GetInstance 252
GetInstantiateRelations 114
GetLicensedApplication 32
GetLink 272
GetLinkAssociation 114
GetMessages 266, 446
GetMinHeight 430
GetMinWidth 430
GetModules 319
GetNestedClasses 115
GetNoteViews 211
GetNoteViewType 330
GetObject 32, 128, 150, 295, 355, 370, 437
GetObjects 447
GetOperation 273
GetOtherRole 50
GetParentContext 211
GetParentItem 508
GetProcessors 194
GetProfileString 33
GetPropertyClassName 226
GetPropertyValue 226
GetQualifiedName 227
GetRealizeRelations 115, 313
GetReceiverObject 273
GetRoleForNameDirection 51
GetRoles 108, 116, 556
GetRoseIniPath 34
GetRoseItem 410
GetSelectedActivities 466
GetSelectedActivityViews 467
GetSelectedCategories 128, 296
GetSelectedClasses 129, 296
GetSelectedDecisions 468
GetSelectedDecisionViews 467
GetSelectedDiagrams 297
GetSelectedExternalDocuments 297
GetSelectedItems 212, 298
GetSelectedLinks 447
GetSelectedMessages 448
GetSelectedModules 299, 319
GetSelectedObjects 448
GetSelectedStates 468
GetSelectedStateVertices 469
GetSelectedStateViews 469
GetSelectedSubsystems 299, 320
GetSelectedSwimLaneViews 470
GetSelectedSynchronizations 470
GetSelectedSynchronizationViews 471
GetSelectedTransitions 471
GetSelectedUseCases 300
GetSendAction 539
GetSenderObject 274

GetSequenceInformation 275
GetSourceState 540
GetSourceStateVertex 540
GetState 516
GetStateMachines 14
GetStateVertexViews 472
GetStateVertices 533
GetStateView 473
GetStateViews 473
GetSubClasses 116
GetSubsystem 531
GetSubsystemDependencies 313, 525
GetSubsystems 320
GetSubUnitItems 166
GetSuperclasses 117
GetSuperUseCases 557
GetSupplier 376
GetSupplierCategory 80
GetSupplierClass 134, 373
GetSupplierComponent 373
GetSwimLanes 514
GetSwimLaneViews 474
GetSynchronization 537
GetSynchronizationViews 475
GetTargetState 541
GetTargetStateVertex 541
GetToolNames 228
GetToolNamesForClass 187
GetToolProperties 228
GetTransitions 503
GetTriggerAction 542
GetTriggerEvent 542
GetUniqueId 229
GetUseCases 129
GetUserDefinedEvents 14, 453
GetViewFrom 212
GetVirtualPath 355
GetVirtualPathWithContext 356
GetVisibilityRelations 525
GetVisibleSubsystems 528
GetWithUniqueID 150
GetXPosition 254
GetYPosition 255
Global 61
green
  fill color 418
  font 421
  line color 424
guard conditions, retrieving or setting 235
GuardCondition 235
Guarded 84, 339

H

hardware 196, 359
has relationships 133, 374
  adding 94
  classes 94, 101, 113
  client cardinality 244
  containment 245
  deleting 101
  export control 246
  retrieving 113
  static 247
  supplier cardinality 247
HasAssignedSubsystem 74
HasClient 376
HasEntry 357
HasItem 431
HasParentView 431
HasRelationship Class 244
HasRelationship Class Methods 247
HasRelationship Class Properties 244
HasSupplier 377
headers, browsing event 579
Height 23, 413
height
  applications 23
  item views 413, 429, 430
  retrieving or setting 23, 413
help file paths
  add-ins 390
  retrieving 390

HelpFilePath 390
History 450
history, retrieving or setting 450
Horizontal 536
horizontal
  determining if 536
  setting 536
  synchronization views 536
horizontal coordinates (x), item views 416

I

icon indexes, retrieving 225, 241
icons 136
  rendering 233, 243
  retrieving index 225
IdentifyClass 437
ImplementationAccess 55, 85, 120, 246, 249, 340, 382
Import 300
importing packages and subsystems 300
index, of objects 151
indexes, retrieving for icons 241
IndexOf 151
inherit relations 133, 374
  adding 94, 550
  classes 94, 102, 113
  deleting 102, 554
  export control 249
  friendship 250
  retrieving 113, 556
  use cases 550, 554, 556
  virtual 250
InheritProperty 229
InheritRelation Class 248
InheritRelation Class Methods 250
InheritRelation Class Properties 248
Initial 451
initial values
  parameters 349
  retrieving or setting 56, 349
initializing Rose applications event 577
InitValue 56, 349
installation directories
  add-ins 390
  retrieving 390
InstallDirectory 390
instance views
  object instances 440
  retrieving 252, 440
  scenario diagrams 440
InstanceView Class 251
InstanceView Methods 251
InstanceView Properties 251
InstanceViews 440
instantiated relations
  adding 95
  classes 95, 102, 114
  deleting 102
  retrieving 114
InstantiatedClass 83
InstantiatedUtility 83
InstantiateRelation Class 252
InstantiateRelation Class Methods 253
InstantiateRelation Class Properties 252
internal names, retrieving or setting 159
InternalName 159
Invalidate 213, 428
invariants
  assumed by operations 342
  satisfied by operations 342
InvolvesFriendship 121
IsActive 213, 399
IsActivityDiagram 455
IsALinkClass 117
IsClass 337, 438
IsControlled 167
IsDataModelingDiagram 130
IsDefaultProperty 230
IsInitialized 23
IsLanguageAddIn 400
IsLoaded 167
IsLocked 168

IsMessageToSelf 276
IsModifiable 168
IsModified 169
IsNestedClass 118
IsOperation 276
IsOverriddenProperty 231
IsRootPackage 348
IsSelected 432
IsToolVisible 188
IsURL 241
IsUseCaseDiagram 130
italic 421
Item 423
item collections 203
item views 416
    center 416
    checking if selected 432
    diagrams 204, 425
    displaying stereotypes 415
    fill color 417, 418, 419
        supporting 433
    fonts 419, 420, 421, 422, 423
    height 413
        retrieving ideal 429
        retrieving minimum 430
    horizontal coordinate (x) 416
    items 423
        checking for 431
    line color, supporting 434
    line vertices 414
    lines 424, 425
    name 414
    parent item views 426
        checking for 431
    points, checking if in item views 432
    redrawing 428
    removing 214
    retrieving 426
    retrieving attached notes 428
    retrieving from diagrams 204, 212
    selecting 433
    subitem views 426
item views, *continued*
    vertical coordinate (y) 416
    width 415
        retrieving ideal 429
        retrieving minimum 430
Items 203
items
    clients 375
    diagrams 203
    documentation 405
    external documents 406
        adding 409
        deleting 410
    item views 423, 431
    localized stereotype 406
    retrieving 290, 375, 376, 379, 410, 423
        by unique id 291
        from diagrams 203
        selected 298
        selected from diagrams 212
    specifications, opening 411
    state machine owners 407, 508
    stereotypes 407
    suppliers 376
ItemViews 204
iterating through collections 145

K

Keys 383
keys
    adding 386
    attributes 383
    deleting 385
    retrieving 383
    roles 383
kinds
    of classes 83
    of relationships 142
    of states 451

L

languages
- add-ins 400
- default 279
- modules 306
- retrieving 306
- setting 306

language-specific issues, operations 343
Layout 214
Left 24
left
- applications 24
- retrieving or setting 24

licensing 32
line colors
- item views 434
- retrieving or setting 424, 425

line vertices 253
- retrieving 414
- sample script 256
- x position, retrieving 254
- y position, retrieving 255

LineColor.Blue 424
LineColor.Green 424
LineColor.Red 425
lines
- blue 424
- color 424, 425
- item views 424, 425
- red 425

LineVertex Class 253
LineVertex Class Methods 254
LineVertex Class Properties 253
LineVertices 414
Link Class 258
Link Class Methods 263
link class names, associations 52
Link Class Properties 258
link classes
- associations 114
- checking for 117

link classes, *continued*
- retrieving 46
- setting 46, 52

link roles, retrieving 259
link visibility 261, 262
LinkClass 46
LinkRole1 259
LinkRole1Shared 259
LinkRole1Visibility 261
LinkRole2 259
LinkRole2Shared 260
LinkRole2Visibility 262
Links 333
links
- associations
    - assigning 265
    - removing 267
    - retrieving 266
- messages
    - adding to 264
    - deleting 265
    - retrieving 266
- object instances 333
    - adding 335
    - deleting 336
- retrieving 272, 333, 447
- scenario diagrams 447

Load 169
LoadControlledUnits 301
LoadScript 35
LocalizedStereotype 406, 485
Lock 170
locking, controllable units 168, 170
logical class collections 58
Logical View 281
logical view, retrieving 281

M

Main 305
Manual 362
menu file path, retrieving 391

menu ids, retrieving 159
menu states, retrieving or setting 160
MenuFilePath 391
MenuID 159
MenuState 160, 268
Message Class 270
Message Class Methods 272
Message Class Properties 270
messages
    adding 264
    checking if
        an operation 276
        to itself 276
    concurrency 271
    creating 444
    frequency 270
    links
        deleting 265
        retrieving 266, 272
    one time only 270
    operations, retrieving 273
    periodically 270
    receiver object, retrieving 273
    retrieving 446, 448
    retrieving sequence numbers 275
    scenario diagrams 446, 448
    sender objects, retrieving 274
    sending 270
    synchronization 271
Meta 83
metafiles 215, 216, 217
Methods for All Collections 146
Methods for User-Defined Collections 151
Model 219
model changes, canceling event 580
Model Class 277
Model Class Methods 283
Model Class Properties 278
model elements
    creating new event 591
    deleting event 583
    modifying event 587
    retrieving 211
model properties, retrieving 279
models 20, 277
    applications 23, 36, 38, 39, 40, 42, 43
    categories
        retrieving 288, 292
        retrieving selected 296
    classes
        retrieving 289, 292
        retrieving name 226
        retrieving selected 296
    closing event 581
    component view 282
    controllable units
        checking for 167
        loading 169, 301
        unloading 173
    creating 36
    creating new event 590
    default languages, determining or setting 279
    default properties 279
        adding 179
        checking tool visibility 188
        cloning 180
        creating 181
        deleting 182
        deleting sets 183
        finding 184
        retrieving 222
            set names 186, 225
            sets 185
            tool names 187
            value 224
        setting tool visibility 188
    deployment diagrams 280

models, *continued*
- deployment units 280
- devices
  - adding 286
  - deleting 287
  - retrieving 293
- diagrams, retrieving active 291
- elements 219
- external documents, retrieving selected 297
- importing packages and subsystems 300
- items
  - retrieving 290, 291
  - retrieving selected 298
- logical view 281
- modules
  - retrieving 293
  - retrieving selected 299
- new 36
- notation 281
- objects, retrieving (RoseScript only) 295
- opening 38, 40
  - as template 39
- opening event 592
- packages
  - retrieving 288, 292
  - retrieving selected 296
- physical implementation 153
- processors
  - adding 286
  - deleting 287
  - retrieving 294
- properties 218
  - checking if
    - default value 230
    - overridden 231
  - creating 221
  - overriding 232
  - restoring default value 229

models, *continued*
- properties, *continued*
  - retrieving 223, 228
    - active set 224
    - all 223
    - value 226
  - specifying current set 234
- qualified name, retrieving 227
- resolving references 301
- retrieving 219
- retrieving current 23
- root categories 281
- root subsystem 282
- root use case categories 282
- saving 42
- saving as 43
- saving event 595
- subsystems
  - relocating 529
  - retrieving 294
  - retrieving selected 299
- tools, retrieving names 228
- unique ids, retrieving 229
- use case view 282
- use cases 283
  - retrieving 295
  - retrieving selected 300

Modifiable 170

modifying model elements event 587

Module Class 302

Module Class Methods 308

Module Class Properties 302

module diagrams
- adding 521
- adding modules 317
- adding subsystem views 318
- modules
  - retrieving 319
  - retrieving selected 319
- relocating 529
- removing component views 321

module diagrams, *continued*
removing subsystem views 322
retrieving 517
retrieving component views 314
retrieving subsystem views 315
subsystems 517, 521, 529
retrieving 320
retrieving, selected 320
module visibility relationships
adding 310
context modules 323
context subsystems 324
deleting 311
retrieving 311, 312, 525
sample script 526
supplier modules 324
supplier subsystems 325
ModuleDiagram Class 314
ModuleDiagram Class Methods 316
ModuleDiagram Class Properties 314
ModuleDiagrams 517
Modules 517
modules 153, 516
adding 520
adding to module diagrams 317
assigned language 306
assigning 92, 107
body 305
classes 92, 107, 118
classes, retrieving 312
declarations 303
deleting 522
dependencies, retrieving 311, 312
generic 305
main 305
module diagrams 314
other parts 303
package types 307
parts 305
paths 306
modules, *continued*
realize relations 372, 373
adding 309
deleting 310
retrieving 313
removing 118
retrieving 155, 293, 323, 324, 517
all 523
from module diagrams 319
selected 299, 319
specification 305
subsystem dependencies, retrieving 313
subsystem visibility relationships, retrieving 313
subsystems 517, 520, 522, 523
modules, relocating 528
retrieving 304
retrieving parent 315
subtypes 307
task types 307
two part 303
types 307
visibility relationships 322
adding 310
context modules 323
deleting 311
retrieving 311, 312
supplier modules 324
ModuleVisibilityRelationship Class 322
ModuleVisibilityRelationship Class Methods 325
ModuleVisibilityRelationship Class Properties 323
MultipleInstances 333
MyProcessor 358

N

Name 219, 236, 368, 391, 414
name direction
    associations 49
    clearing 49
    existence 51
    roles
        retrieving 51
        setting 52
NameIsDirectional 51
names
    add-ins 391
    events 236
    font 420
    item views 414
    product 25
    properties 368
    retrieving 219, 228, 236, 368, 391, 414
        add-in display name 399
        qualified 227
    server 393
    setting 219, 236, 368, 414
    tools 393
Navigable 383
navigable
    determining if or setting 383
    roles 383
NeedsRefreshing 171
nested classes
    adding 95
    checking for 118
    deleting 103
    retrieving 115
    retrieving all 106
new models, creating event 590
NewModel 36
NewScript 36
NonPreemptive 362
Normal 451
NormalClass 83
NotAClassKind 83
Notation 281
notation
    determining or setting 281
    enumerated types 326
NotationTypes 281, 326
note views
    adding attachments 329
    adding to diagrams 209
    boxed 330
    free floating 330
    removing 215
    retrieving attached 428
    retrieving diagrams 330
    retrieving from diagram 211
    text, retrieving 327
    types 330
notes
    boxed 330
    free floating 330
    note views, types 330
    retrieving attached 428
    text, retrieving 327
NoteView Class 327
NoteView Class Methods 328
NoteView Class Properties 327

O

object instances
    adding 442, 443
    classes
        checking if class 337
        retrieving 336
        retrieving names 333
    deleting 445
    instance views 440
    links
        adding 335
        deleting 336
        retrieving 333

object instances, *continued*
multiple, determining if or setting 333
persistent 334
removing 449
role 1 links 259
shared visibility 259
specified visibility 261
role 2 links 259
shared visibility 260
specified visibility 262
scenario diagrams 445, 449
setting class names 333
static 334
ObjectInstance Class 332
ObjectInstance Class Methods 334
ObjectInstance Class Properties 332
objects 46
adding to user-defined collections 152
adding user-defined 152
checking for existence 147
classes
checking for specific class (RoseAutomation only) 438
identifying (RoseAutomation only) 437
collections 147, 148, 149, 150, 151, 153
finding first in collections 147
finding next in collections 148
instances
adding links to 335
checking if class 337
deleting links from 336
retrieving classes 336
multiple instances 333
OLE interface object, retrieving (RoseScript only) 437
pathmaps, retrieving (RoseScript only) 355
properties 370
objects, *continued*
receivers 273
removing
all from collections 153
from collections 153
retrieving 32, 410, 447, 448
(RoseScript only) 295, 370
first from collections 149
from collections 149, 150
index 151
scenario diagrams 445, 447, 448
senders 274
typecasting (RoseScript only) 435, 439
unique ids 150
user-defined collections 152
OLE class name 393
OLE interface objects, retrieving (RoseScript only) 437
OLE object 389
OLE objects, retrieving (RoseScript only) 32, 128, 150
OMT 281, 326
OMTNotation 326
OnActivate 576
OnAppInit 577
OnBrowseBody 577
OnBrowseHeader 579
OnCancelModel 580
OnCloseModel 581
OnDeactivate 582
OnDeletedModelElement 583
OnEnableContextMenuItems 585
OnGenerateCode 586
OnModifiedModelElement 587
OnNewModel 590
OnNewModelElement 591
OnOpenModel 592
OnPropertySpecOpen 593
OnSaveModel 595
OnSelectedContextMenuItem 596
Open 242

OpenCustomSpecification 411, 503
OpenExternalDocument 37
opening models event 592
opening specifications event 593
OpenModel 38
OpenModelAsTemplate 39
OpenRoseModel 40
OpenScript 41
OpenSpecification 411, 504
OpenURL 41
Operation Class 337
Operation Class Methods 345
Operation Class Properties 338
Operations 86
operations
  absolute storage 344
  actions 343
  adding 96
  allowable 342
  as messages 276
  classes 96, 103
  classes, retrieving parent 341
  deleting 103
  displaying or hiding on diagrams 137
  entry behavior 342
  exceptions, retrieving 339
  exit behavior 342
  friend 384
  invariants 342
  language-specific issues 343
  legal order 342
  parameters 348
    adding 345
    deleting 346
    deleting all 346
    retrieving 341
  postconditions, retrieving 342
  preconditions, retrieving 342
  protocols 342
  qualifications 343
  relative storage 344
  retrieving 86, 273
operations, *continued*
  return types 343
  roles 384
  semantics 343
  size 344
  suppressing display on diagrams 138
  time 344
  virtual 344
operations signature, displaying or hiding on diagrams 137
other parts, retrieving 303
OtherPart 303
overridden properties, checking for 231
OverrideProperty 232

P

Package Class 347
Package Class Methods 347
Package Class Properties 347
package dependencies
  adding 65
  client packages, retrieving 79
  deleting 69
  retrieving 74
  supplier packages, retrieving 80
packages 58
  adding 65
  adding to diagrams 124
  associations 47, 59, 75
  checking if root 348
  class diagrams 60, 66, 70, 76, 122
  classes 60, 66, 70, 72, 76
    retrieving all 292
  deleting 69
  external documents 238
  global 61
  importing 300
  package dependencies 69, 74
    adding 65
  parent package 61

packages, *continued*
  relocating 75
  removing from diagrams 131
  retrieving 47, 59, 122, 288, 292, 524, 545
    all 72
    by unique id 288
    from diagrams 126, 128
    selected 296
  retrieving parent 61, 87
  root 78, 281
  scenario diagrams 62, 68, 71, 77
  subpackages 59, 69, 72
  subsystems 73, 74, 77, 524
  top level 78
  use cases 62, 68, 71, 73, 545
PackageType 307
Parameter Class 348
Parameter Class Properties 348
parameterized classes 97, 104
ParameterizedClass 83, 97, 104
ParameterizedUtililty 83
Parameters 87, 341
parameters
  adding 97
  adding to operations 345
  classes 97, 104
  constant 349
  deleting 104, 346
  deleting all from operations 346
  initial values 349
  operations 341
  retrieving 87, 341
Parent 455, 510
parent classes, retrieving 341, 486
parent diagrams, retrieving 425
parent item views 426, 431
parent items, retrieving for state machine owners 508
parent subsystems, retrieving 518
ParentCategory 47, 61, 87, 122, 238, 545
ParentClass 56, 88, 341, 486
ParentDiagram 238, 425
ParentItem 239
ParentStateMachine 510
ParentSubsystem 304, 315, 518
ParentView 426
Part 305
parts, determining or setting 305
Path 239, 306, 390
PathMap 24
PathMap Class 350
PathMap Class Methods 351
PathMap Class Properties 350
pathmaps
  actual paths
    retrieving 353
    retrieving within context 353
  applications 24
  entries
    adding 352
    checking for 357
    deleting 352
  objects, retrieving (RoseScript only) 355
  retrieving 24
  virtual paths
    retrieving 355
    retrieving within context 356
paths 390, 391, 392
  actual
    retrieving 353
    retrieving within context 353
  applications 22, 24
  external documents 239
  help files 390
  modules 306
  property files 392
  retrieving 22, 239, 306, 390, 391, 392
  root registry 392
  Rose.ini, retrieving 34
  setting 239, 306

paths, *continued*
- virtual
  - retrieving 355
  - retrieving within context 356

Persistence 88, 334
persistence
- classes 88
- determining or setting 88, 334
- object instances 334

physical implementation 302
PointInView 432
points 253
points, item views 432
position of object in collections 145
Postconditions 342
postconditions
- operations 342
- retrieving or setting 342

Preconditions 342
preconditions
- operations 342
- retrieving or setting 342

Preemptive 362
Priority 359
priority
- determining or setting 359
- processes 359

PrivateAccess 55, 85, 120, 246, 249, 340, 382
Process Class 358
Process Class Methods 359
Process Class Properties 358
Processes 361
processes
- adding 364
- deleting 365
- priority 359
- processors
  - associated 361
  - retrieving 358
- retrieving 361
- static 359

Processor Class 359
Processor Class Methods 363
Processor Class Properties 360
processors 190
- adding 193, 286
- characteristics 360
- connected devices, retrieving 366
- connected processors, retrieving 366
- connections
  - adding 365
  - creating 199
  - removing 201
- deleting 287
- device connections
  - adding 364
  - removing 367
- processes 361
  - adding 364
  - associated 358
  - deleting 365
- processor connections, removing 367
- removing 195
- retrieving 294, 358
  - connected processors 200
  - from diagrams 194
- retrieving connections 361
- scheduling 362

product names
- applications 25
- retrieving 25

ProductName 25
profile strings
- applications 33, 44
- retrieving 33
- writing 44

program files 153

properties 218
- adding 179
- checking
  - if default value 230
  - if overridden 231
  - tool visibility 188
- cloning 180
- creating 181, 221
- default 279
- deleting 182
- deleting sets 183
- finding 184
- item views 416
- names, retrieving or setting 368
- objects, retrieving 370
- overriding 232
- restoring default value 229
- retrieving 223, 228, 279
  - active set 224
  - all 223
  - set names 186, 225
  - sets 185
  - tool names 187
  - value 224, 226
- setting tool visibility 188
- specifying current set 234
- stored information 369
- tool names 369, 393
- types 369
- value 370

Property Class 368
Property Class Methods 370
Property Class Properties 368
property file paths
- retrieving 392
- rose add-ins 392

PropertyFilePath 392
ProtectedAccess 55, 85, 120, 246, 249, 340, 382
Protocol 342
protocols
- operations 342
- retrieving or setting 342

PublicAccess 55, 85, 120, 246, 249, 340, 382

Q

Qualification 343
qualifications
- operations 343
- retrieving or setting 343

R

Rank 545
rank
- retrieving or setting 545
- use cases 545

Rational Unified Solutions Partner Program 575
ReadSetting 400
realize relations 371, 374
- adding 99, 309
- classes 99, 105, 115
- client class, retrieving 372
- client components, retrieving 372
- client modules, retrieving 372
- deleting 105, 310
- modules 313
- retrieving 115, 313
- supplier classes, retrieving 373
- supplier components, retrieving 373
- supplier modules, retrieving 373

RealizeRelation Class 371
RealizeRelation Class Methods 371
RealizeRelation Class Properties 371
receiver objects, retrieving 273
red
- fill color 418
- fonts 422
- lines 425

RedirectTo 543
redrawing
  diagrams 213
  item views 428
references, resolving 301
Refresh 171
refreshing
  controllable units 171
  diagrams 213
refreshing, controllable units 171
registry
  add-ins 400, 401
  root paths 392
Relation Class 374
Relation Class Methods 375
Relation Class Properties 374
relation views, adding 209
relations 133, 244, 248, 374
  checking for
    clients 376
    suppliers 377
  clients, retrieving 375
  supplier names 374
  suppliers, retrieving 376
relationships 133, 244, 248
  category dependencies 78
  dependency 189
  package dependencies 78
RelocateAssociation 75
RelocateCategory 75
RelocateClass 76
RelocateClassDiagram 76
RelocateModule 528
RelocateModuleDiagram 529
RelocateScenarioDiagram 77
RelocateState 453, 504
RelocateSubSystem 529
Remove 153
RemoveActivityView 476
RemoveAll 153
RemoveAllParameters 346
RemoveAssignedModule 118
RemoveAssociation 131
RemoveCategory 131
RemoveClass 132
RemoveComponentView 321
RemoveDecisionView 477
RemoveDevice 194
RemoveDeviceConnection 201, 367
RemoveInstanceView 449
RemoveItemView 214
RemoveNoteView 215
RemoveProcessor 195
RemoveProcessorConnection 201, 367
RemoveStateView 478
RemoveSubsystemView 322
RemoveSwimLaneView 479
RemoveSynchronizationView 480
RemoveUseCase 132
Render 215
RenderEnhanced 216
RenderEnhancedToClipBoard 216
RenderIconToClipboard 233, 243
rendering enhanced, diagrams 216
rendering, diagrams 215, 217
RenderToClipboard 217
ResolveReferences 301
return types, retrieving or setting 343
ReturnType 343
rich data types
  ClassKind 83
  Concurrency 84, 339
  Containment 54, 245, 381
  ExportControl 55, 85, 120, 246, 249, 340, 382
  LinkVisibility 261, 262
  Part 305
  Scheduling 362
  StateKind 451
  Type 307
role 1 links
  object instances 259
    shared visibility 259
    specified visibility 261

role 2 links
  object instances 259
    shared visibility 260
    specified visibility 262
Role Class 377
Role Class Methods 385
Role Class Properties 377
role collections 48
role name directions, associations 52
Role1 47
Role2 48
Roles 48
roles
  aggregate classes 378
  associate 107
  associate items 379
  associations 51, 379
  attributes 383, 384
  cardinality 379
  checking if name directional 51
  classes 116, 380
    retrieving names 386
  constraints 380
  containment 381
  corresponding 50
  export control 382
  friend 384
  keys 383
    adding 386
    deleting 385
  navigable 383
  operations 384
  other class 50
  retrieving 47, 48, 50, 51, 116, 556
  static 383
  use cases 384, 556
root categories
  determining 78
  retrieving 281
root packages
  determining 78
  retrieving 281
root registry paths
  add-ins 392
  retrieving 392
root subsystem
  checking if 530
  retrieving 282
root use case categories or packages, retrieving 282
RootCategory 281
RootRegistryPath 392
RootSubsystem 282
RootUseCaseCategory 282
rose add-ins
  activating 395
  checking if
    active 399
    language add-in 400
  company name 388
  context menu items
    adding 395
    retrieving 398
  copyright 388
  deactivating 397
  display name, retrieving 399
  event handler 389
  executing scripts 397
  fundamental types 389
  help file paths 390
  installation directory 390
  menu file path 391
  name 391
  property file path 392
  registry settings
    creating 401
    retrieving 400
  root registry paths 392
  server name 393
  shortcut menu items
    adding 395
    retrieving 398
  tool names 393
  versions 394

Rose applications, initializing event 577
Rose.ini 44
  retrieving profile strings 33
RoseAddIn Class 387
RoseAddIn Class Methods 394
RoseAddIn Class Properties 387
RoseAddInEventTypes 401
RoseAddInManager Class 402
RoseAddInManager Class Methods 403
RoseAddInManager Class Properties 402
RoseItem Class 405
RoseItem Class Methods 408
RoseItem Class Properties 405
RoseItemView Class 412
RoseItemView Class Methods 427
RoseItemView Class Properties 412
RoseObject Class 434
RoseObject Class Methods 435
RoseObject Class Properties 434
rsActivity 161
rsAnyKind 142
rsAttribute 161
rsClass 161
rsComponent 161
rsDecision 161
rsDefault 161
rsDeploymentUnit 161
rsDiagram 161
rsDisabled 268
rsDisabledAndChecked 268
rsDisabledAndUnchecked 268
rsEnabled 268
rsEnabledAndChecked 268
rsEnabledAndUnchecked 268
rsExternalDoc 161
rsFriend 142
rsModel 161
rsOnNewModel 401
rsOperation 161
rsPackage 162
rsProperties 162
rsRole 162
rsState 162
rsSwimlane 162
rsSynchronization 162
rsTransition 162
rsTypeAny 143
rsTypeAssociation 143
rsTypeDependency 143
rsTypeHas 143
rsTypeInherits 143
rsTypeInstantiation 143
rsTypeRealizes 143
rsUseCase 162

S

Save 42, 172
SaveAs 43, 172
saving models event 595
scenario diagrams
  adding 68, 551
  adding object instances 442, 443
  deleting 71, 555
  instance views 440
  links, retrieving selected 447
  messages
    creating 444
    retrieving 446
    retrieving selected 448
  object instances
    deleting 445
    removing 449
  objects
    creating 445
    retrieving 447
    retrieving selected 448
  relocating 77
  retrieving 62, 546
  types, retrieving 446
  use cases 543, 546, 551, 555
ScenarioDiagram Class 440
ScenarioDiagram Class Methods 441
ScenarioDiagram Class Properties 440

ScenarioDiagrams 62, 546
scenarios 440
Scheduling 362
scheduling, determining or setting 362
screen 24, 25, 26
scripts
  applications 36, 41
  compiling 29
  creating (RoseScript only) 36
  executing 30, 397
  loading (RoseScript only) 35
  new (RoseScript only) 36
  opening (RoseScript only) 41
  unloading (RoseScript only) 31
selected diagrams, retrieving 297
selected, item views 432
selecting shortcut (context) menu items event 596
Semantics 343
semantics
  operations 343
  retrieving or setting 343
sender objects, retrieving 274
Sequence Diagram 446, 551
sequence numbers, retrieving 275
Sequential 84, 339
server names, retrieving 393
ServerName 393
SetAssignedSubsystem 77
SetCurrentPropertySetName 234
SetLinkClassName 52
SetRoleForNameDirection 52
SetSelected 433
SetToolVisibility 188
shortcut menu items
  enabling event 585
  selecting event 596
shortcut menus 158, 159, 160, 161, 268, 395, 398
ShowAllAttributes 136
ShowAllOperations 137
ShowOperationsSignature 137
simple 271
Size 344
size
  font 422
  operations 344
  retrieving 344
  setting 344
sizing icons 136
Space 88
space algorithms, retrieving or setting 88
Specification 305
specifications
  custom 503
  opening 411, 503, 504
  opening event 593
  state machines 503, 504
specifications, opening 411
State Class 449
State Class Methods 452
State Class Properties 450
state diagrams 454, 515
  activities 466
  activity views 458, 462, 464, 467, 476
  decision views 459, 463, 464, 467, 477
  decisions 468
  retrieving 484
  state machines 455, 484
  state vertex views 465, 472
  state vertices 469
  state views
    adding 459
    removing 478
    retrieving 473
    retrieving all 473
    retrieving selected 469
  states, retrieving selected 468
  swimlane views 460, 465, 470, 474, 479

state diagrams, *continued*
- synchronization views 461, 466, 471, 475, 480
- synchronizations 470
- transitions, retrieving selected 471

state kinds, retrieving or setting 451

state machine owners
- items 407
- retrieving 407, 486
- retrieving parent Rose item 508
- state machines 486, 506, 507, 508

state machines
- abstract states 7, 9, 14, 482, 499
- activities 483, 492, 499
- activity diagrams 492
- adding 7
- classes 99, 105, 486
- creating 99, 507, 553
- decisions 483, 493, 500
- deleting 9, 105, 508
- diagrams 500
- documentation 484
- external documents 485, 494, 497
- parent classes 486
- retrieving 14, 89, 455, 506, 510, 546
- sample script 547
- specifications 503, 504
- state diagrams 455, 484
- state machine owners 486, 506, 507, 508
- state vertices 487, 498, 501, 510
- statechart diagrams 495
- states 487
  - adding 495
  - deleting 497
  - relocating 504
  - retrieving 501
- stereotypes 485, 488
- swimlanes 488, 496, 498
- synchronizations 489, 496, 502

state machines, *continued*
- transitions
  - retrieving 503
  - retrieving all 502
- use cases 546

state transitions 234

state vertex views
- retrieving 465, 472
- state diagrams 465, 472

state vertices
- deleting 498
- retrieving 540, 541
  - all 501
  - from state machines 487
  - from swimlanes 533
  - parent 510
  - selected 469
- state diagrams 469
- state machines 487, 498, 501, 510
- swimlanes 514, 533
- transitions 511, 512, 514

state views 454, 515
- adding 459
- removing 478
- retrieving 465, 472, 473
  - all 473
  - selected 469
- state diagrams 459, 469, 473, 478
- states, retrieving 516

statechart diagrams
- adding 495
- retrieving all 500
- state machines 495

StateDiagram Class 454

StateDiagram Class Methods 456

StateDiagram Class Properties 455

StateKind 451

StateMachine 89, 546

StateMachine Class 480

StateMachine Class Methods 490

StateMachine Class Properties 481

StateMachineOwner 407, 486

StateMachineOwner Class 505
StateMachineOwner Class Methods 506
StateMachineOwner Class Properties 505
StateMachines 506
States 487
states 234
 abstract 3, 11
 adding 495
 deleting 497, 498
 history 450
 kinds 451
 relocating 504
 retrieving 540, 541
  all 11, 499, 501
  child states 3
  from state machines 482, 487, 501
  from swimlanes 533
  from view 516
  initial (source) 540
  parent 510
  selected 468, 469
 state diagrams 468
 state machines 487, 495, 497, 501, 504
 state views 516
 substates
  adding 452
  deleting 453
  relocating 453
 target, retrieving 541
 top level 504
 transitions 540, 541, 543
 user-defined events, retrieving 453
StateVertex Class 509
StateVertex Class Methods 511
StateVertex Class Properties 509
StateVertices 487
StateView Class 515
StateView Class Methods 515
StateView Class Properties 515
Static 56, 247, 383
static 334
 determining 247, 383
 has relationships 247
 processes 359
 retrieving 56
 roles 383
 setting 247, 383
Stereotype 407, 488
StereotypeDisplay 415
stereotypes
 displaying as none, label, decoration, or icon 415
 items 406, 407
 localized 406
 retrieving or setting 406, 407, 485, 488
 state machines 485, 488
strikethrough 422
SubActivities 2
subactivities
 retrieving 2
 retrieving all 10
subclasses, retrieving 116
SubDecisions 3
subdecisions
 retrieving 3
 retrieving all 11
subitem views 426
SubStates 3
substates
 deleting 453
 relocating 504
 retrieving all 11
 states 453
 top level state 504
SubSynchronizations 4
subsynchronizations, retrieving all 12
Subsystem Class 516
Subsystem Class Methods 519
Subsystem Class Properties 516
subsystem dependencies, retrieving 525

subsystem module visibility relationships, retrieving 313
subsystem views
  adding 318
  removing 322
  retrieving 315
subsystem views, retrieving subsystems 531
Subsystems 518
subsystems 153
  adding 521
  adding to module diagrams 318
  assigning 77
  categories, retrieving 524
  checking for 74
  checking if
    root subsystem 530
    top level 530
  child subsystems 518
    adding 521
    deleting 522
    retrieving all 523
  classes, retrieving 524
  context 324
  deleting 522
  dependencies, retrieving 313, 525
  importing 300
  module diagrams 517
    adding 521
    relocating 529
  module visibility relationships 324, 525
    supplier subsystems 325
  modules 304, 315, 517
    adding 520
    deleting 522
    relocating 528
    retrieving all 523
  packages, retrieving 524
  parent 518
  relocating 529
subsystems, *continued*
  retrieving 73, 294, 304, 315, 324, 325, 518, 531
    all 523
    from module diagrams 320
    parent 518
    selected 299, 320
  root 282
  subsystem dependencies, retrieving 525
  subsystem views 531
  top level 518
  visible subsystems, retrieving 528
SubsystemView Class 530
SubsystemView Class Methods 531
SubsystemView Class Properties 530
SubsystemViews 315
SubType 307
subunits
  opening 40
  retrieving all 165
  retrieving immediate 166
SubViews 426
superclasses, retrieving 117
supplier
  determining if a device 157
  determining if a processor 157
supplier cardinality, has relationships 247
supplier modules, retrieving 324
supplier names, retrieving or setting 374
supplier subsystems, retrieving 325
SupplierCardinality 121, 247
SupplierIsDevice 157
SupplierModule 324
SupplierName 374
suppliers
  relations 374, 376, 377
  retrieving 376
SupplierSubsystem 325
SupportsFillColor 433
SupportsLineColor 434

SuppressAttributes 138
SuppressOperations 138
SwimLane Class 532
SwimLane Class Methods 532
SwimLane Class Properties 532
swimlane views
- adding 460
- removing 479
- retrieving
  - for swimlanes 474
  - from state diagrams 465
  - selected 470
- state diagrams 460, 465, 470, 474, 479
- swimlanes 474

SwimLanes 488
swimlanes
- adding 496
- deleting 498
- retrieving 488, 514
- state machines 488, 496, 498
- state vertices 514, 533
- swimlane views 474

SwimLaneView Class 533
SwimLaneView Class Methods 534
SwimLaneView Class Properties 534
Synchronization 271
synchronization views
- adding 461
- horizontal 536
- removing 480
- retrieving 465, 466, 472, 475
- retrieving selected 471
- state diagrams 461, 466, 471, 475, 480
- synchronizations 470, 475, 537

synchronization, determining or setting 271
Synchronizations 489
synchronizations
- abstract states 4, 12
- adding 496
- deleting 498
- retrieving 540, 541
  - all 12, 501, 502
  - children 4
  - from state machines 487, 489
  - from swimlanes 533
  - from view 537
  - parent 510
  - selected 469, 470
- state diagrams 470
- state machines 489, 496, 502
- synchronization views 470, 475, 537

Synchronous 84, 339
synchronous 271
SyncItem Class 534
SyncItem Class Methods 535
SyncItem Class Properties 535
SyncItemView Class 535
SyncItemView Class Methods 537
SyncItemView Class Properties 536

T

Target 16
target states 541
targets
- actions 16
- of transitions 543
- retrieving or setting 16

TaskType 307
Text 327
text, point size 422
Time 344
time
- operations 344
- retrieving or setting 344

timeout 271

tool names
    properties 369
    retrieving 369, 393
ToolName 369
ToolNames 393
tools, retrieving names 228
Top 25
top
    applications 25
    retrieving or setting 25
top level
    subsystems 530
    use cases 545
TopLevel 78, 530
Transition Class 538
Transition Class Methods 538
Transition Class Properties 538
Transitions 511
transitions
    actions
        retrieving 539
        retrieving triggers 542
    adding 512, 513
    deleting 514
    events, retrieving triggers 542
    redirecting 543
    retrieving 503, 511
        all 502
        selected 471
    retrieving source 540
    retrieving targets 541
    state diagrams 471
    state machines 502, 503
    state vertices 511, 512, 514
    states
        redirecting to 543
        retrieving initial (source) 540
        retrieving target 541
    targets 543
transparent, fill color 419
triggers
    actions 542
    events 542
Type 57, 307, 369
TypeCast 439
typecasting (RoseScript only) 435
typecasting, objects 439
types
    context menu items 161
    determining 307
    of relationships 143
    of scenario diagrams 446
    operation return types 343
    setting 307
    shortcut menu items 161

U

UML 281, 326
UMLNotation 326
UnAssignAssociation 267
unchecked 160, 268
Uncontrol 173
underlining 423
unique ids
    categories 288
    classes 289
    items 291
    objects 150
    packages 288
    retrieving 229
Unload 173
Unlock 174
unlocking, controllable units 174
Update 217
updating, diagrams 217
URL 240

URLs
  applications 41
  checking for external documents 241
  external documents 240
  opening 41
  retrieving or setting 240
use case diagrams
  adding 66, 552
  checking for 130
use case view, retrieving 282
use cases 440
  abstract 544
  adding 68
  adding diagrams to 552
  adding to diagrams 125
  associations
    adding 549
    deleting 553
    retrieving 555
  class diagrams 544
    adding 550
    deleting 554
  deleting 71
  get all 73
  inherit relations
    adding 550
    deleting 554
    retrieving 556
  parent category or package 545
  parent use cases, retrieving 557
  rank 545
  removing from diagrams 132
  retrieving 62, 283, 295, 384
    from diagrams 129
    selected 300
  roles 384
  roles, retrieving 556
  scenario diagrams 546
    adding 551
    deleting 555
  state machines 546, 553
  top level 545
UseCase 384
UseCase Class 543
UseCase Class Methods 548
UseCase Class Properties 543
UseCases 62, 283
user-defined events
  abstract states 8, 9, 14
  adding 8
  deleting 9
  retrieving 14
Utility 83

V

Value 370
values
  for RoseItemView Color Properties 416
  retrieving or setting 370
Version 26, 394
versions
  applications 26
  retrieving 26, 394
vertical coordinates (y), item views 416
Virtual 250, 344
virtual symbols 353, 357
virtual, determining or setting 250, 344
Visibility 139
visibility
  of access adornments 139
  retrieving or setting 204, 259, 260, 261, 262
  shared 259, 260
  specified 261, 262
visibility of Rose, retrieving or setting 26
visibility relationships
  adding 310
  context modules 323
  context subsystems 324
  deleting 311
  retrieving 311, 312, 525

visibility relationships, *continued*
  sample script 526
  supplier modules 324
  supplier subsystems 325
Visible 26, 204
visible
  applications 26
  subsystems 528

W

Width 27, 415
width
  applications 27
  item views 415, 429, 430
  retrieving or setting 27, 415
window, main
  Height 23
  Left 24
  Top 25
  Width 27
windows metafiles 215, 216, 217
WriteErrorLog 43
WriteProfileString 44
WriteSetting 401

X

XPosition 416

Y

YPosition 416

Z

zoom factor, determining or setting 205
ZoomFactor 205