



Version 6.5 User Guide

Enterprise Architect is an intuitive, flexible and powerful UML analysis and design tool for building robust and maintainable software. From requirements gathering, through to analysis, modeling, implementation, testing, deployment and maintenance, Enterprise Architect is a fast, feature-rich, multi-user UML modeling tool, driving the long-term success of your software project



© Copyright 1998-2006 Sparx Systems

Enterprise Architect

Introduction

by Geoffrey Sparks

Enterprise Architect 6.5 is a complete UML based solution for analysing, designing, managing, sharing and building software systems.

Enterprise Architect 6.5 User Guide

© 1998-2006 Sparx Systems

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: September 2006

Publisher

Sparx Systems

Managing Editor

Geoffrey Sparks

Technical Editors

Geoffrey Sparks

Paul Mathers

Emma Newman

Melanie Coffey

Dermot O'Bryan

Deborah Johnson

John Redfern

Neil Capey

Evan Sparks

Special thanks to:

All the people who have contributed suggestions, examples, bug reports and assistance in the development of EA over the last six years. The task of developing and maintaining this tool has been greatly enhanced by their contribution.

Table of Contents

| | |
|--|-----------|
| Foreword | 1 |
| Part I Introduction | 3 |
| 1 What is Enterprise Architect? | 3 |
| 2 EA Features | 4 |
| 3 Copyright Notice | 5 |
| 4 Trademarks | 5 |
| 5 Differences Between Desktop, Professional and Corporate Editions | 6 |
| 6 EA Software Product License Agreement | 7 |
| 7 Ordering Enterprise Architect | 10 |
| 8 Support | 10 |
| 9 Available Helpfile Formats | 10 |
| 10 Your Feedback | 11 |
| 11 EA for Power Users | 11 |
| Part II Project Roles and EA | 13 |
| 1 Business Analyst | 13 |
| 2 Software Architects | 15 |
| 3 Software Engineer | 16 |
| 4 Developer | 17 |
| 5 Project Manager | 19 |
| 6 Testers | 21 |
| 7 Deployment and Rollout | 22 |
| 8 Technology Developer | 23 |
| 9 Database Administrator | 25 |
| Part III Getting Started | 28 |
| 1 Installation | 28 |
| 2 Quick Start | 29 |
| 3 Starting the Application | 32 |
| 4 Licence Management | 33 |
| Add License Key | 34 |
| Keystore Troubleshooting | 35 |
| 5 Registering a Full License | 35 |
| 6 Upgrade an Existing License | 37 |
| 7 Finding Your License Information | 38 |
| Part IV Using Enterprise Architect | 41 |

| | | |
|----------|---|------------|
| 1 | The Application Workspace | 41 |
| 2 | The Start Page | 43 |
| | The Start Page Options | 44 |
| | Removing Recent Projects | 44 |
| | Edit My Profile | 45 |
| 3 | Model Patterns | 47 |
| | Business Process Model Pattern | 48 |
| | Requirements Model Pattern | 48 |
| | Use Case Model Pattern | 49 |
| | Domain Model Pattern | 50 |
| | Class Model Pattern | 51 |
| | Database Model Pattern | 52 |
| | Component Model Pattern | 52 |
| | Deployment Model Pattern | 53 |
| | Testing Model Pattern | 54 |
| | Maintenance Model Pattern | 55 |
| | Project Model Pattern | 55 |
| 4 | Arranging Windows and Menus | 56 |
| | Dockable Windows | 56 |
| | Dockable Window 2005 Style | 57 |
| | Autohide Windows | 59 |
| | Tear Off Menus | 59 |
| 5 | The Main Menu | 60 |
| | The File Menu | 61 |
| | The Edit Menu | 62 |
| | The View Menu | 63 |
| | The Project Menu | 66 |
| | The Diagram Menu | 70 |
| | The Element Menu | 72 |
| | The Tools Menu | 76 |
| | The Customize Window | 77 |
| | Customize Commands | 78 |
| | Customize Toolbars | 79 |
| | Custom Tools | 81 |
| | Opening External Tools | 83 |
| | Passing Parameters to External Applications | 84 |
| | Customize Keyboard | 85 |
| | Customize Menu | 87 |
| | Customize Options | 88 |
| | The Settings Menu | 89 |
| | The Window Menu | 92 |
| | The Help Menu | 93 |
| 6 | View Options | 94 |
| | Diagram View | 94 |
| | Report View | 95 |
| | Search View | 98 |
| 7 | Searching a Project | 101 |
| | Searching the Search View | 101 |
| | Search Definitions | 102 |
| | Creating Search Definitions | 105 |
| | Pre-defined Search Definitions | 107 |
| | Adding Filters | 107 |

| | |
|---|------------|
| Fields and Conditions..... | 109 |
| 8 Workspace Toolbars | 110 |
| Default Tools Toolbar | 111 |
| Project Toolbar | 111 |
| Code Generation Toolbar | 112 |
| UML Elements Toolbar | 114 |
| Diagram Toolbar | 115 |
| Current Element Toolbar | 116 |
| Current Connector Toolbar | 117 |
| Format Toolbar | 118 |
| Workspace Views | 119 |
| Customize Toolbars | 119 |
| Diagram Tabs | 120 |
| Status Bar | 121 |
| 9 The Project Browser | 122 |
| Model Context Menu | 123 |
| Root Node Package Control Sub-Menu..... | 125 |
| Package Context Menu | 125 |
| Package Contol Sub-Menu..... | 127 |
| Add Sub-Menu..... | 127 |
| Documentation Sub-Menu..... | 128 |
| Code Engineering Sub-Menu..... | 129 |
| Build and Run Sub-Menu..... | 129 |
| Import/Export Sub-Menu..... | 130 |
| Contents Sub-Menu..... | 131 |
| Element Context Menu - Project Browser | 131 |
| Add Sub Menu..... | 133 |
| Diagram Context Menu - Project Browser | 133 |
| Order Package Contents | 134 |
| Setting Default Project Browser Behavior | 134 |
| 10 Dockable Windows | 136 |
| The Properties Window | 137 |
| The Resource Window | 139 |
| The Resource View..... | 140 |
| Favorites..... | 140 |
| The Notes Window | 141 |
| The System Window | 142 |
| The Source Code Viewer | 142 |
| The Element Browser | 144 |
| The Relationships Window | 145 |
| The Rules Window | 145 |
| The Hierarchy Window | 146 |
| The Tagged Values Window | 146 |
| Predefined Tagged Value Types..... | 148 |
| Predefined Reference Data..... | 150 |
| Creating a CustomTagged Value Type..... | 153 |
| Assigning defined Tagged Value to an Item..... | 154 |
| Assigning Information to a Tagged Value..... | 156 |
| Allow Duplicate Tags..... | 157 |
| The Project Management Window | 158 |
| The Output Window | 159 |
| Instant Help Window | 160 |
| The Pan & Zoom Window | 161 |

| | |
|--|------------|
| 11 The Quick Linker | 161 |
| Creating New Elements | 162 |
| Creating Connections Between Elements | 163 |
| 12 The UML Toolbox | 164 |
| UML Toolbox Shortcut Menu | 166 |
| Analysis Group | 166 |
| Use Case Group | 167 |
| Class Group | 168 |
| Composite Group | 169 |
| Communication Group | 170 |
| Interaction Group | 171 |
| Timing Group | 171 |
| State Group | 172 |
| Activity Group | 172 |
| Component Group | 173 |
| Deployment Group | 174 |
| Custom Group | 175 |
| Profile Group | 176 |
| Metamodel Group | 177 |
| WSDL Group | 177 |
| XML Schema Group | 178 |
| Requirement Group | 178 |
| Maintenance Group | 179 |
| User Interface Group | 179 |
| Perspectives | 182 |
| Configure Perspectives..... | 182 |
| Import Export Perspectives..... | 184 |
| 13 Package Tasks | 184 |
| Open a Package | 184 |
| Adding Packages | 185 |
| Renaming Packages | 185 |
| Drag a Package onto a Diagram | 186 |
| Show or Hide Package Contents | 186 |
| Delete a Package | 186 |
| 14 Diagram Tasks | 187 |
| Add a Diagram | 187 |
| Highlight Context Element | 187 |
| Layout a Diagram | 188 |
| Legends | 190 |
| Lock Diagram | 192 |
| Pan a Diagram | 192 |
| Scale Image to Page Size | 192 |
| Select a Diagram | 193 |
| Set Appearance Options | 194 |
| Visible Class Members..... | 195 |
| Set Diagram Page Size | 196 |
| Show or Hide Attributes and Operations | 197 |
| Undo Last Action | 198 |
| View Last and Next Diagram | 198 |
| Zoom a Diagram View | 198 |
| 15 Element Tasks | 199 |
| Add an Element | 199 |

| | |
|---|------------|
| Connecting Elements | 200 |
| Moving Elements and Packages | 200 |
| Auto Counters | 201 |
| Templates Package | 202 |
| Deleting an Element | 203 |
| 16 Element Inplace Editing Options | 204 |
| Inplace Element Item Tasks | 205 |
| Edit Element Name | 206 |
| Edit Attribute or Operation Stereotype | 207 |
| Edit Attribute and Operation Scope | 208 |
| Edit Attribute Keyword | 209 |
| Edit Operation Parameter Keyword | 211 |
| Edit Parameter Kind | 212 |
| Insert new Attribute or Operation | 214 |
| Insert Operation Parameter | 215 |
| Insert Maintenance Feature | 217 |
| Insert Testing Feature | 220 |
| 17 Defaults and User Settings | 223 |
| Configure Local Options | 223 |
| General | 225 |
| Standard Colors | 226 |
| Diagram..... | 227 |
| Behavior | 228 |
| Sequence | 229 |
| Objects | 230 |
| Element Visibility..... | 231 |
| Links | 232 |
| Communication Colors..... | 233 |
| XML Specifications..... | 234 |
| UML Element Toolbox..... | 235 |
| Source Code Engineering..... | 236 |
| Custom Layouts | 236 |
| Visual Styles | 237 |
| 18 Using Add-In's | 237 |
| Register Add-In | 237 |
| Register Zicom Mentor..... | 238 |
| The Add-In Manager | 240 |
| 19 Keyboard Shortcuts | 240 |
| Part V The UML Language | 244 |
| 1 What is UML | 244 |
| 2 UML Diagrams | 245 |
| Behavioral Diagrams | 246 |
| Activity Diagram..... | 248 |
| Use Case Diagram..... | 250 |
| State Machine Diagram (formerly State Diagram)..... | 251 |
| Regions | 253 |
| Pseudo-States..... | 254 |
| Interaction Diagrams..... | 255 |
| Timing Diagram..... | 255 |
| Sequence Diagram..... | 256 |
| Denote Lifecycle of an Element | 258 |

| | |
|---|------------|
| Layout of Sequence Diagrams..... | 259 |
| Sequence Element Activation..... | 260 |
| Lifeline Activation Levels..... | 261 |
| Sequence Elements..... | 263 |
| Sequence Message Label Visibility..... | 264 |
| Changing the Top Margin..... | 264 |
| Inline Sequence Elements..... | 265 |
| Communication Diagram (formerly Collaboration Diagram)..... | 266 |
| Communication Diagrams in Color..... | 268 |
| Interaction Overview Diagram..... | 269 |
| Structural Diagrams | 271 |
| Package Diagram..... | 272 |
| Class Diagram..... | 274 |
| Object Diagram..... | 276 |
| Composite Structure Diagram..... | 278 |
| Properties | 279 |
| Component Diagram..... | 281 |
| Deployment Diagram..... | 282 |
| Additional Diagrams | 284 |
| Analysis Diagram..... | 284 |
| Custom Diagram..... | 285 |
| Requirements Diagram..... | 286 |
| Maintenance Diagram..... | 287 |
| User Interface Diagram..... | 288 |
| Database Schema..... | 290 |
| Robustness Diagram..... | 290 |
| 3 UML Elements | 291 |
| Behavioral Diagram Elements | 291 |
| Structural Diagram Elements | 294 |
| Basic Elements | 295 |
| Action | 296 |
| Action Notation..... | 296 |
| Action Expansion Node..... | 298 |
| Action Pin | 299 |
| Local Pre/Post Conditions..... | 300 |
| Activity | 301 |
| Activity Notation..... | 303 |
| Activity Parameter Nodes..... | 303 |
| Activity Partition..... | 306 |
| Actor | 307 |
| Artifact | 308 |
| Choice | 308 |
| Class | 309 |
| Active Classes..... | 311 |
| Parameterized Classes (Templates)..... | 311 |
| Collaboration..... | 313 |
| Collaboration Occurrence..... | 314 |
| Combined Fragment..... | 316 |
| Create a Combined Fragment..... | 318 |
| Interaction Operators..... | 318 |
| Component..... | 321 |
| Datastore..... | 321 |
| Decision..... | 322 |
| Deployment Spec..... | 323 |

| | |
|---|-----|
| Diagram Gate..... | 324 |
| Document Artifact..... | 325 |
| Endpoint..... | 325 |
| Entry Point..... | 326 |
| Exception..... | 327 |
| Expansion Region..... | 327 |
| Add Expansion Region..... | 329 |
| Exit Point..... | 329 |
| Final..... | 330 |
| Flow Final..... | 331 |
| Fork/Join..... | 332 |
| Fork..... | 333 |
| Join..... | 334 |
| History..... | 335 |
| Information Item..... | 337 |
| Initial..... | 337 |
| Interaction Occurrence..... | 338 |
| Interface..... | 339 |
| Interruptible Activity Region..... | 340 |
| Add Interruptible Activity Region..... | 341 |
| Junction..... | 341 |
| Lifeline..... | 343 |
| Node..... | 343 |
| Object..... | 344 |
| Instance Classifier..... | 345 |
| Run-time State..... | 345 |
| Define a Run-time Variable..... | 346 |
| Remove a Defined Variable..... | 347 |
| Define a Run-time State..... | 347 |
| Package..... | 348 |
| Part..... | 348 |
| Partition..... | 349 |
| Port..... | 351 |
| Adding a Port to an Element..... | 351 |
| Managing Inherited and Redefined Ports..... | 352 |
| Qualifiers..... | 353 |
| Receive..... | 354 |
| Region..... | 355 |
| Send..... | 355 |
| State..... | 356 |
| Composite State..... | 357 |
| State Lifeline..... | 359 |
| State/Continuation..... | 360 |
| Continuation..... | 361 |
| State Invariant..... | 362 |
| Structured Activity..... | 363 |
| SubMachine..... | 364 |
| Synch..... | 365 |
| System Boundary..... | 365 |
| Terminate..... | 367 |
| Use Case..... | 367 |
| Use Case Extension Points..... | 368 |
| Using Rectangle Notation..... | 370 |
| Value Lifeline..... | 370 |

| | |
|--|------------|
| Inbuilt and Extended Stereotypes | 371 |
| Analysis Stereotypes..... | 372 |
| Boundary..... | 373 |
| Create a Boundary..... | 373 |
| Business Modeling Stereotypes..... | 374 |
| Composite Elements..... | 375 |
| Control | 376 |
| Create a Control Element..... | 376 |
| Entity | 377 |
| Create an Entity..... | 377 |
| Event | 377 |
| Hyperlinks..... | 378 |
| N-Ary Association..... | 381 |
| Other Elements..... | 381 |
| Process..... | 382 |
| Recursion..... | 382 |
| Requirements..... | 382 |
| Screen | 383 |
| Table | 385 |
| UI Element..... | 385 |
| Web Stereotypes..... | 387 |
| Worker | 388 |
| 4 UML Connections | 388 |
| Aggregate | 390 |
| Change Aggregation Link Form..... | 391 |
| Assembly | 391 |
| Associate | 392 |
| Diagonal Representation..... | 393 |
| Association Class | 393 |
| Link a New Class to an Existing Association..... | 394 |
| Communication Path | 395 |
| Compose | 396 |
| Connector | 396 |
| Control Flow | 397 |
| Delegate | 398 |
| Dependency | 399 |
| Applying a Stereotype..... | 399 |
| Deployment | 400 |
| Expose Interface | 401 |
| Extend | 401 |
| Generalize | 402 |
| Include | 403 |
| Information Flow | 403 |
| Realizing an Information Flow..... | 404 |
| Convey Information on a Flow..... | 404 |
| Interrupt Flow | 405 |
| Manifest | 406 |
| Message | 406 |
| Message (Communication Diagram)..... | 407 |
| Create a Communication Message..... | 407 |
| Re-Order Messages..... | 408 |
| Message (Sequence Diagram)..... | 410 |
| Self-Message | 411 |
| Call | 412 |

| | |
|--|------------|
| Changing the Timing Details..... | 413 |
| Message Examples..... | 414 |
| Message (Timing Diagram)..... | 415 |
| Create a Timing Message..... | 416 |
| Message Endpoint..... | 418 |
| Message Label..... | 419 |
| Nesting | 420 |
| Object Flow | 420 |
| Using Object Flows in Activity Diagrams..... | 421 |
| Occurrence | 422 |
| Package Import | 423 |
| Package Merge | 423 |
| Realize | 424 |
| Role Binding | 425 |
| Represents | 425 |
| Representation | 426 |
| Trace | 426 |
| Transition | 427 |
| Use | 429 |

Part VI Modeling with UML

431

| | |
|--|------------|
| 1 Working with UML Diagrams | 432 |
| Diagram Context Menu | 434 |
| Set the Default Diagram | 435 |
| Common Diagram Tasks | 436 |
| New Diagrams..... | 436 |
| Deleting Diagrams..... | 437 |
| Diagram Properties..... | 437 |
| Renaming Diagrams..... | 438 |
| Copy Diagram Element..... | 438 |
| Diagram Navigation Hotkeys..... | 439 |
| Convert Linked Element to Local Copy..... | 439 |
| Z Order Elements..... | 439 |
| Copy Image to Disk..... | 440 |
| Copy Diagram Image to Clipboard..... | 440 |
| Change Diagram Type..... | 440 |
| Open a Package..... | 441 |
| Duplicate a Diagram..... | 442 |
| Set Feature Visibility..... | 443 |
| Insert Diagram Properties Note..... | 444 |
| Autosize Elements | 445 |
| Drop Elements from the Project Browser..... | 445 |
| Pasting from the Project Browser..... | 446 |
| Pasting Multiple Items from the Project Browser..... | 447 |
| Pasting Composite Elements..... | 448 |
| Pasting Activities..... | 449 |
| Place Related Elements on Current Diagram | 450 |
| Swimlanes..... | 450 |
| Using the Image Manager..... | 452 |
| Select Alternate Image..... | 453 |
| Create Custom Diagram Background..... | 454 |
| Import Image Library..... | 454 |
| Show Realized Interfaces for a Class..... | 456 |

| | |
|---|------------|
| Label Menu Section..... | 457 |
| Document Options..... | 457 |
| 2 Working with UML Elements | 458 |
| Element Context Menu | 460 |
| Properties Menu Section..... | 461 |
| Custom Properties Dialog..... | 461 |
| Advanced Settings..... | 462 |
| Embedding Menu Section..... | 463 |
| Embedded Elements..... | 463 |
| Insert Related Elements..... | 464 |
| Features Menu Section..... | 465 |
| Code Engineering Menu Section..... | 465 |
| Appearance Menu Section..... | 466 |
| Common Actions Menu Section..... | 466 |
| Element Context Menu - Multiple Selection..... | 467 |
| Common Element Tasks | 468 |
| Creating Elements..... | 468 |
| New Elements..... | 468 |
| Set Element Parent..... | 469 |
| Show Element Usage..... | 470 |
| Set Up References (Cross References)..... | 471 |
| Copying Attributes and Operations Between Elements..... | 472 |
| Moving Attributes and Operations between Elements..... | 473 |
| Moving Elements between Packages..... | 475 |
| Changing Element Type..... | 475 |
| Moving Elements..... | 476 |
| Aligning Elements..... | 477 |
| Resizing Elements..... | 477 |
| Deleting Elements..... | 478 |
| Customize Visible Elements..... | 479 |
| Creating Notes and Text..... | 480 |
| Configure an Element's Default Appearance..... | 482 |
| Get/Set Project Custom Colors..... | 484 |
| Attributes and Operations | 487 |
| Attributes..... | 487 |
| Attributes Main Page..... | 489 |
| Attributes Detail..... | 490 |
| Attribute Constraints..... | 490 |
| Attribute Tagged Values..... | 491 |
| Creating Properties..... | 492 |
| Displaying Inherited Attributes..... | 494 |
| Operations..... | 496 |
| Operations Main Page..... | 497 |
| Operations Detail..... | 499 |
| Initial Code | 500 |
| Operation Parameters..... | 501 |
| Operation Parameter Tagged Values..... | 502 |
| Operation Parameters by Reference..... | 503 |
| Operation Constraints..... | 505 |
| Operation Tagged Values..... | 505 |
| Override Parent Operations..... | 506 |
| Displaying Inherited Operations..... | 507 |
| Element Properties | 509 |
| Properties..... | 509 |

| | |
|--|------------|
| General Settings..... | 510 |
| Advanced Settings..... | 511 |
| Requirements..... | 511 |
| External Requirements..... | 512 |
| Constraints..... | 513 |
| Links | 514 |
| Scenarios..... | 515 |
| Associated Files..... | 516 |
| Tagged Values..... | 517 |
| Advanced Tag Management..... | 518 |
| Quick Add of Tagged Values..... | 520 |
| Object Classifiers..... | 521 |
| Using Classifiers..... | 521 |
| Boundary Element Settings..... | 523 |
| Compartments | 524 |
| Tag Compartment..... | 524 |
| Responsibility Compartment..... | 525 |
| Constraint Compartment..... | 525 |
| Testing Compartment..... | 525 |
| Maintenance Compartment..... | 525 |
| Linking Notes | 525 |
| Linking Notes to Internal Documentation..... | 525 |
| Link a Note to an Element..... | 526 |
| Linked Documents | 527 |
| Creating Linked Documents..... | 529 |
| Creating Document Artifacts..... | 529 |
| Linking Document to UML Elements (Corporate Edition Only)..... | 530 |
| Editing Linked Documents..... | 531 |
| Linked Document Templates..... | 532 |
| Creating Linked Document Templates..... | 532 |
| Editing Linked Document Templates..... | 533 |
| 3 Working with UML Connectors | 535 |
| Connector Context Menu | 535 |
| Properties Menu Section..... | 535 |
| Type Specific Menu Section..... | 536 |
| Common Actions Menu Section..... | 536 |
| Style Menu Section | 537 |
| Appearance Menu Section..... | 537 |
| Common Connector Tasks | 537 |
| Add a Note to a Link..... | 538 |
| Arranging Connectors..... | 539 |
| Change Connector Type..... | 540 |
| Change the Source or Target Element..... | 540 |
| Connecting Elements..... | 541 |
| Connector Styles..... | 542 |
| Create Link..... | 543 |
| Deleting Connectors..... | 545 |
| Generalization Sets..... | 545 |
| Hide/Show Connectors..... | 546 |
| Hide/Show Labels | 547 |
| Reverse Connector..... | 548 |
| Set Association Specializations..... | 549 |
| Set Relation Visibility..... | 549 |
| Show Uses Arrow Head..... | 550 |

| | |
|---|------------|
| Tree Style Hierarchy..... | 551 |
| Connector Properties | 552 |
| Connector Constraints..... | 553 |
| Source Role..... | 554 |
| Destination Role..... | 556 |
| Role Tagged Values..... | 556 |
| Message Scope..... | 557 |
| 4 UML Stereotypes | 557 |
| Applying Stereotypes | 559 |
| Custom Stereotypes | 560 |
| Stereotype Selector | 561 |
| Stereotype Visibility | 562 |
| Standard Stereotypes | 563 |
| Stereotypes with Alternate Images | 565 |
| Shape Scripts | 565 |
| Getting Started..... | 566 |
| Writing Scripts..... | 568 |
| Syntax Grammar..... | 569 |
| Shape Attributes..... | 569 |
| Drawing Methods..... | 570 |
| Color Queries | 572 |
| Conditional Branching..... | 573 |
| Query Methods..... | 573 |
| Displaying Element Properties..... | 573 |
| Sub-Shapes | 574 |
| Reserved Names..... | 575 |
| Miscellaneous | 575 |
| Example Scripts..... | 575 |
| Shape Editor..... | 578 |
| 5 MDG Technologies | 579 |
| Creating MDG Technologies | 579 |
| Adding Profile in MDG Technology Wizard..... | 583 |
| Adding Pattern in MDG Technology Wizard..... | 583 |
| Adding Tagged Values in MDG Technology Wizard..... | 584 |
| Adding Code Modules in MDG Technology Wizard..... | 585 |
| Adding MDA Transforms in MDG Technology Wizard..... | 586 |
| Adding Images in MDG Technology Wizard..... | 587 |
| Working with MDG Technologies | 588 |
| Import MDG Technologies | 590 |
| Add MDG Technologies to UML Toolbox | 591 |
| 6 UML Profiles | 592 |
| Creating Profiles | 594 |
| Create a Profile Package..... | 594 |
| Add Stereotypes and Metaclasses to UML Profiles..... | 595 |
| Define Stereotype Tags..... | 596 |
| Define Stereotype Tags with Predefined Tag Types..... | 597 |
| Define Stereotype Tags with Supported Attributes..... | 598 |
| Using the Tagged Value Connector..... | 599 |
| Define Stereotype Constraints..... | 600 |
| Add Enumerations to UML Profiles..... | 601 |
| Add Redefinitions to UML Profiles..... | 602 |
| Export a UML Profile..... | 603 |
| Using Profiles | 605 |

| | |
|--|------------|
| Import a UML Profile | 605 |
| Add Profiles to UML Toolbox..... | 606 |
| Tagged Values in Profiles | 608 |
| Add Profile Connector to Diagram..... | 609 |
| Synchronize Tags and Constraints | 610 |
| Profile References | 611 |
| Supported Types..... | 611 |
| Profile Structure..... | 612 |
| Supported Attributes | 613 |
| Example Profile..... | 614 |
| 7 UML Patterns | 616 |
| Create a Pattern | 616 |
| Import a Pattern | 618 |
| Use a Pattern | 618 |
| 8 Requirements Management | 621 |
| Creating Requirements | 622 |
| External Requirements | 622 |
| Color Coded External Requirements | 623 |
| Internal Requirements | 624 |
| Move Internal Requirements to External Requirement | 625 |
| Requirement Properties | 627 |
| Composition | 628 |
| Implementation | 628 |
| Requirements Hierarchy | 628 |
| The Matrix | 629 |
| Dependency Report | 629 |
| 9 Element Relationship Matrix | 630 |
| Open the Relationship Matrix | 630 |
| Setting Source and Target Package | 631 |
| Setting Element Type | 632 |
| Setting the Link Type and Direction | 632 |
| Matrix Options | 633 |
| Modifying Relationships in Matrix | 633 |
| Exporting to CSV | 635 |
| Printing the Matrix | 636 |
| Profiles | 637 |
| 10 Business Modeling | 637 |
| Part VII MDA Transforms | 643 |
| 1 Transforming Elements | 644 |
| Chaining Transformations | 646 |
| 2 Importing Transforms | 646 |
| 3 Transformation Templates | 646 |
| 4 Built-in Transformations | 648 |
| C# Transformation | 648 |
| DDL Transformation | 650 |
| EJB Transformations | 652 |
| Java Transformation | 655 |
| JUnit Transformation | 657 |
| NUnit Transformation | 659 |
| WSDL Transformation | 661 |

| | |
|--|------------|
| XSD Transformation | 662 |
| 5 Writing Transformations | 665 |
| Default Transformation Templates | 666 |
| Intermediary Language | 666 |
| Objects | 666 |
| Connectors | 670 |
| Duplicate Information | 672 |
| Converting Types | 672 |
| Converting Names | 672 |
| Cross References | 673 |

Part VIII Model Management

675

| | |
|---|------------|
| 1 Project Discussion Forum | 676 |
| Categories, Topics and Posts | 677 |
| Adding a New Category | 677 |
| Adding a New Topic | 678 |
| Adding a New Post | 679 |
| Replying to a Post | 679 |
| Editing an Item | 680 |
| Deleting an Item | 681 |
| Forum Connections | 681 |
| Message Dialog | 683 |
| Adding Element Links | 683 |
| Copy Path to Clipboard | 684 |
| Context Menu | 685 |
| Forum Options | 685 |
| 2 EA Project Files | 686 |
| Open a Project | 687 |
| Create a New Project | 688 |
| Model Wizard | 688 |
| Setting Up a Database Repository | 689 |
| Setting Up an ODBC Driver | 690 |
| Setup a MySQL ODBC Driver | 690 |
| Setup a PostgreSQL ODBC Driver | 693 |
| Setup an Adaptive Server Anywhere ODBC Driver | 696 |
| Setup a Progress OpenEdge ODBC Driver | 700 |
| Connecting to a Data Repository | 702 |
| Connect to a MySQL Data Repository (Corporate Edition only) | 702 |
| Connect to a SQL Server Data Repository (Corporate Edition only) | 705 |
| Connect to an Oracle9i Data Repository (Corporate Edition only) | 708 |
| Connect to a PostgreSQL Data Repository (Corporate Edition only) | 710 |
| Connect to an Adaptive Server Anywhere Data Repository (Corporate Edition only) | 712 |
| Connect to a MSDE Server Data Repository (Corporate Edition only) | 714 |
| Connect to a Progress OpenEdge Repository (Corporate Edition only) | 714 |
| Creating a Repository | 716 |
| Create a New MySQL Repository (Corporate Edition only) | 716 |
| Create a New SQL Server Repository (Corporate Edition only) | 719 |
| Create a New Oracle9i Server Repository (Corporate Edition only) | 721 |
| Create a New PostgreSQL Repository (Corporate Edition only) | 721 |
| Create a New Adaptive Server Anywhere Repository (Corporate Edition Only) | 724 |
| Create a New MSDE Server Repository (Corporate Edition only) | 726 |

| | |
|---|------------|
| Create a New Progress OpenEdge Repository (Corporate Edition only)..... | 727 |
| Create and Open Model Files Discussion | 727 |
| Copy a Base Project | 728 |
| 3 Upgrading Models | 729 |
| The Upgrade Wizard | 729 |
| Upgrade Replicas | 730 |
| 4 Data and Model Integrity | 730 |
| Checking Data Integrity | 730 |
| Running SQL Patches | 732 |
| 5 Data Transfer | 732 |
| Perform a Data Transfer | 733 |
| Why Compare Models? | 734 |
| Comparing Models | 734 |
| Copy Packages from One Project to Another | 735 |
| 6 Upsizing Models | 736 |
| Upsizing to MySQL | 737 |
| Upsizing to SQL Server | 738 |
| Upsizing to Oracle | 740 |
| Upsizing to PostgreSQL | 742 |
| Upsizing to Sybase Adaptive Server Anywhere (ASA) | 743 |
| Upsizing to SQL Server Desktop Engine (MSDE) | 745 |
| Upsizing to Progress OpenEdge | 745 |
| 7 Model Maintenance | 747 |
| Rename a Project | 747 |
| Compact a Project | 747 |
| Repair a Project | 748 |
| 8 Manage Views | 749 |
| Add Additional Views | 750 |
| Rename Views | 751 |
| Delete Views | 752 |
| 9 Model Validation | 753 |
| Configuring Model Validation | 755 |
| Rules Reference | 755 |
| Well-Formedness (Element, Relationship, Feature, Diagram)..... | 756 |
| Element Composition..... | 756 |
| Property Validity (Element, Relationship, Feature)..... | 757 |
| OCL Conformance (Element, Relationship, Feature)..... | 757 |
| 10 MOF | 760 |
| Getting Started | 762 |
| Export MOF to XMI | 763 |
| Import MOF from XMI | 764 |
| 11 XMI Import and Export | 764 |
| Export to XMI | 765 |
| Import from XMI | 766 |
| Limitations of XMI | 768 |
| The UML DTD | 768 |
| Controlled Packages | 769 |
| Controlled Package Menu..... | 769 |
| Configure Packages..... | 770 |
| Disconnect Controlled Package..... | 772 |
| Save a Package..... | 772 |

| | |
|--|------------|
| Load a Package..... | 773 |
| Batch XMI Export..... | 774 |
| Batch XMI Import..... | 775 |
| Manual Version Control with XMI..... | 776 |
| 12 CSV Import and Export | 777 |
| CSV Specifications | 777 |
| CSV Export | 779 |
| CSV Import | 780 |
| 13 Version Control | 781 |
| Version Control Overview | 782 |
| Version Control Setup | 783 |
| Using Version Control | 783 |
| Version Control Reference | 784 |
| Version Control Setup Menu..... | 784 |
| Version Control Options Dialog..... | 785 |
| Version Control Menu..... | 787 |
| Version Control Options SCC..... | 788 |
| SCC Providers Dialog..... | 791 |
| SCC Version Control User Options..... | 792 |
| Version Control with CVS..... | 792 |
| CVS with Remote Repositories..... | 793 |
| CVS with Local Repositories..... | 797 |
| Version Control with Subversion..... | 800 |
| Setting up Subversion..... | 800 |
| Creating a new Repository Sub-tree..... | 801 |
| Create a Local Working Copy..... | 801 |
| Configure Version Control with Subversion..... | 802 |
| TortoiseSVN | 804 |
| Version Control with TFS..... | 805 |
| Configure Package for Version Control..... | 807 |
| Checking In and Checking out Packages..... | 809 |
| Offline Version Control..... | 810 |
| Review Package History..... | 812 |
| Remove Package from Version Control..... | 813 |
| Add Previously defined Version Control Configurations..... | 814 |
| SCC Version Control Upgrade for 4.5..... | 814 |
| Including Other Users Packages..... | 815 |
| Specifying Private or Shared Models..... | 816 |
| Using Nested Version Control Packages..... | 817 |
| 14 Model Sharing and Team Deployment | 817 |
| Sharing an Enterprise Architect Project | 818 |
| Sharing a Project | 819 |
| Distributed Development | 819 |
| User Security | 820 |
| Security Policy..... | 821 |
| Enabling Security..... | 822 |
| Maintaining Users..... | 823 |
| Set Up User Groups..... | 825 |
| Set Up Single Permissions..... | 826 |
| View All User Permissions..... | 827 |
| Maintaining Groups..... | 827 |
| Set Group Permissions..... | 828 |
| List of Available Permissions..... | 829 |

| | |
|--|------------|
| View and Manage Locks..... | 830 |
| Password Encryption..... | 832 |
| Locking Model Elements..... | 833 |
| Adding Connectors Between Locked Elements..... | 834 |
| Locking Packages..... | 835 |
| Apply a User Lock..... | 836 |
| Discovering Who Has Locked An Object..... | 838 |
| Managing Your Own Locks..... | 838 |
| Replication | 839 |
| Creating Replicas..... | 841 |
| Design Masters..... | 841 |
| Synchronizing Replicas..... | 842 |
| Remove Replication..... | 843 |
| Upgrading Replicas..... | 843 |
| Resolving Conflicts..... | 844 |
| 15 Baselines and Differences | 845 |
| Baselines | 845 |
| Managing Baselines..... | 846 |
| Creating Baselines..... | 847 |
| The Compare Utility (Diff) | 847 |
| Example Comparison | 848 |
| Merging Baselines | 849 |
| 16 Spell Checking | 849 |
| Using the Spell Checker | 849 |
| Correcting Words | 850 |
| User Dictionaries | 851 |
| Select Language | 851 |
| 17 Reference Data | 852 |
| People | 853 |
| Model Authors..... | 853 |
| Clients | 854 |
| Resources..... | 855 |
| Roles | 856 |
| General Types | 857 |
| Requirement Types..... | 857 |
| Status Types..... | 858 |
| Constraint Status Types..... | 860 |
| Constraint Types..... | 860 |
| Scenario Types..... | 861 |
| Maintenance | 862 |
| Problem Types..... | 862 |
| Testing Types..... | 863 |
| Metrics and Estimation | 864 |
| UML | 864 |
| Standard Element Stereotypes..... | 864 |
| Tagged Values..... | 865 |
| Data Types | 865 |
| Import and Export Reference Data | 866 |
| Export Reference Data..... | 867 |
| Import Reference Data..... | 867 |

| | |
|--|------------|
| 1 Estimation | 870 |
| TCF Dialog | 870 |
| ECF Dialog | 871 |
| Estimating Project Size | 873 |
| Default Hours | 874 |
| 2 Resource Allocation | 875 |
| Resource Management | 876 |
| Effort Management | 876 |
| Risk Management | 877 |
| Metrics | 877 |
| Resource Report | 878 |
| Roles | 879 |
| System Clients | 880 |
| Effort Types | 880 |
| Metric Types | 881 |
| Risk Types | 882 |
| 3 Testing | 883 |
| The Testing Workspace | 884 |
| The Test Details Dialog | 884 |
| Unit Testing | 885 |
| Integration Testing | 886 |
| System Testing | 886 |
| Acceptance Testing | 887 |
| Scenario Testing | 888 |
| Import Scenario as Test | 889 |
| Import Test from other elements | 890 |
| Test Details Report | 892 |
| Show Test Scripts in Compartments | 893 |
| Test Documentation | 893 |
| 4 Maintenance | 894 |
| The Maintenance Workspace | 895 |
| Show Maintenance Scripts in Compartments | 895 |
| 5 Changes and Defects | 896 |
| Defects (Issues) | 897 |
| Changes | 898 |
| Element Properties | 899 |
| Assign People to Defects or Changes | 899 |
| 6 Model Tasks List | 900 |
| Model Tasks Tab | 900 |
| Adding, Modifying and Deleting Tasks | 901 |
| 7 Maintenance Element Properties | 902 |
| 8 Project and Model Issues | 903 |
| Project Issues Dialog | 903 |
| Model Issues Tab | 904 |
| Add, Delete and Modify Issues | 905 |
| Using the Project Issues Dialog..... | 905 |
| Using the Model Issues Tab..... | 906 |
| Generate a Report | 907 |
| Reports - Using the Project Issues Dialog..... | 907 |
| Reports - Using the Model Issues Tab..... | 908 |
| Report Output Sample..... | 909 |

| | |
|---|------------|
| 9 Model Glossary | 910 |
| The Glossary Dialog | 911 |
| Model Glossary Tab | 911 |
| Add, Delete and Modify Glossary Entries | 912 |
| Using the Glossary Dialog..... | 912 |
| Using the Model Glossary Tab..... | 913 |
| Generate a Report | 914 |
| Glossary Report Output Sample | 915 |
| 10 Update Package Status | 916 |
| 11 Manage Bookmarks | 918 |
| | |
| Part X Code Engineering | 920 |
| | |
| 1 Reverse Engineer and Synchronizing | 920 |
| Import a Directory Structure | 921 |
| Import ActionScript | 922 |
| Import C | 922 |
| Import C++ | 922 |
| Import C# | 923 |
| Import Delphi | 923 |
| Import Java | 923 |
| Import PHP | 924 |
| Import Python | 924 |
| Import Visual Basic | 924 |
| Import VB.Net | 924 |
| Import Binary Module | 925 |
| MDG Link and Code Engineering | 926 |
| Handling of Classes not found during Import | 926 |
| Import Source Code | 926 |
| Synchronize Model and Code | 928 |
| 2 Generate Source Code | 929 |
| How to Generate Code | 930 |
| Generate a Single Class | 930 |
| The Code Generation Dialog | 932 |
| Generate a Group of Classes | 933 |
| Generate a Package | 933 |
| Generate Package Dialog | 935 |
| Update Package Contents | 935 |
| Namespaces | 936 |
| 3 Code Engineering Settings | 937 |
| Source Code Engineering | 937 |
| Source Code Options..... | 938 |
| Import Component Types..... | 939 |
| Options - Code Editors..... | 939 |
| Options - Object Lifetimes..... | 940 |
| Options - Attribute/Operations..... | 941 |
| Code Page for Source Editing..... | 942 |
| Local Paths | 943 |
| Local Path Dialog | 944 |
| Language Macros | 945 |
| Setting Collection Classes | 946 |
| Language Options | 948 |
| Options - ActionScript..... | 949 |

| | |
|---|------------|
| Options - C..... | 949 |
| Options - C#..... | 950 |
| Options - C++..... | 951 |
| Options - Delphi..... | 952 |
| Delphi Properties..... | 953 |
| Options - Java..... | 956 |
| Options - PHP..... | 956 |
| Options - Python..... | 957 |
| Options - Visual Basic..... | 958 |
| Options - VB.Net..... | 958 |
| Reset Options..... | 959 |
| 4 Code Template Framework | 960 |
| Code Templates | 960 |
| Base Templates..... | 961 |
| Custom Templates..... | 964 |
| Execution of Code Templates..... | 965 |
| Code Template Syntax..... | 965 |
| Literal Text | 966 |
| Macros | 966 |
| Template Substitution Macros..... | 967 |
| Field Substitution Macros..... | 967 |
| Tagged Value Macros..... | 974 |
| Function Macros..... | 974 |
| Control Macros..... | 977 |
| Variables | 980 |
| Variable Definitions | 980 |
| Variable References..... | 981 |
| The Code Template Editor | 982 |
| Overriding Default Templates..... | 984 |
| Adding New Stereotyped Templates..... | 984 |
| Creating Templates For Custom Languages..... | 985 |
| Importing and Exporting Code Templates..... | 986 |
| Synchronizing Code | 986 |
| Synchronizing Existing Sections..... | 986 |
| Adding New Sections to Existing Features..... | 987 |
| Adding New Features and Elements..... | 987 |
| 5 Modeling Conventions | 987 |
| Actionscript Conventions | 988 |
| C Conventions | 988 |
| C# Conventions | 989 |
| C++ Conventions | 990 |
| Managed C++ Conventions..... | 991 |
| C++/CLI Conventions..... | 991 |
| Delphi Conventions | 992 |
| Java Conventions | 993 |
| AspectJ Conventions..... | 993 |
| PHP Conventions | 994 |
| Python Conventions | 994 |
| VB.Net Conventions | 994 |
| Visual Basic Conventions | 995 |
| Part XI Build and Run | 998 |
| 1 Setup for Build and Run | 999 |

| | |
|--|-------------|
| Managing Scripts | 1000 |
| Build Script | 1000 |
| Build Command..... | 1001 |
| Recursive Builds..... | 1002 |
| Test Command..... | 1003 |
| Run Command..... | 1004 |
| Deploy Command..... | 1005 |
| Debug Command..... | 1005 |
| Java | 1006 |
| Attach to VM..... | 1007 |
| Debugging Java Web Servers..... | 1008 |
| .NET | 1008 |
| Debugging ASP .NET..... | 1009 |
| Debugging Assemblies..... | 1012 |
| Debugging another process..... | 1013 |
| Debugging - COM interop..... | 1014 |
| Debugging - CLR Versions..... | 1015 |
| 2 Profiling and Debugging | 1015 |
| System Requirements | 1015 |
| Using the Debugger | 1016 |
| The Debug Toolbar..... | 1017 |
| The Debug Window..... | 1019 |
| Local Variables..... | 1020 |
| Output | 1021 |
| Stack | 1021 |
| Recording History..... | 1022 |
| Breakpoints | 1023 |
| States | 1023 |
| Adding | 1024 |
| Deleting | 1024 |
| Disabling / Enabling..... | 1024 |
| Generating Sequence Diagrams | 1024 |
| Recording and Diagramming a Debug Session..... | 1025 |
| The Debug Workbench | 1028 |
| Workbench Variables..... | 1029 |
| Creating Workbench Variables..... | 1030 |
| Deleting Workbench Variables..... | 1032 |
| Invoking Methods..... | 1033 |
| 3 Unit Testing | 1034 |
| Setting Up Unit Testing | 1034 |
| Running Unit Tests | 1035 |
| Recording Test Results | 1036 |
| Part XII XML Technologies | 1039 |
| 1 XML Schema (XSD) | 1039 |
| Model XSD | 1039 |
| UML Profile for XSD..... | 1041 |
| XSD Datatypes Package..... | 1048 |
| Abstract XSD models..... | 1048 |
| Default UML to XSD mappings..... | 1050 |
| Generate XSD | 1051 |
| Import XSD | 1051 |

| | |
|--|-------------|
| 2 Web Services (WSDL) | 1052 |
| Model WSDL | 1052 |
| WSDL Namespace..... | 1053 |
| WSDL Document..... | 1054 |
| WSDL Service..... | 1055 |
| WSDL Port Type..... | 1057 |
| WSDL Message..... | 1057 |
| WSDL Binding..... | 1058 |
| WSDL Port Type Operation..... | 1059 |
| WSDL Message Part..... | 1061 |
| Generate WSDL | 1061 |
| Import WSDL | 1061 |
| | |
| Part XIII Data Modeling | 1064 |
| 1 A Data Model Diagram | 1065 |
| 2 Create a Table | 1065 |
| 3 Set Table Properties | 1066 |
| Set Table Owner | 1068 |
| Set MySQL Options | 1069 |
| Set Oracle Table Options | 1070 |
| 4 Create Columns | 1071 |
| 5 Primary Key | 1073 |
| Primary Key Extended Properties | 1074 |
| 6 Foreign Keys | 1075 |
| 7 Stored Procedures | 1079 |
| 8 Views | 1081 |
| 9 Indexes, Triggers and Check Constraints | 1083 |
| 10 Generate DDL | 1085 |
| 11 Generate DDL for a Package | 1087 |
| 12 Data Type Conversion Procedure | 1088 |
| 13 Data Type Conversion for a Package | 1089 |
| 14 DBMS Datatypes | 1091 |
| 15 Import Database Schema from ODBC | 1091 |
| Select a Data Source | 1093 |
| Select Tables | 1094 |
| The Imported Class Elements | 1095 |
| | |
| Part XIV Creating Documents | 1098 |
| 1 RTF Documents | 1098 |
| Generate RTF Document | 1099 |
| RTF Report Dialog..... | 1100 |
| Word Substitution..... | 1102 |
| RTF Document Options | 1102 |
| RTF Templates Dialog | 1104 |
| New RTF Style Template Editor..... | 1104 |
| Selecting Model Elements for Documentation..... | 1105 |
| RTF Template Editor - Adding Content..... | 1107 |

| | |
|--|-------------|
| RTF Template Editor Tabular Sections..... | 1107 |
| RTF Template Editor Child Sections..... | 1109 |
| RTF Template Editor Commands..... | 1111 |
| Scrolling through text..... | 1112 |
| File and Print Options..... | 1112 |
| Line Editing | 1114 |
| Block Editing..... | 1114 |
| Clipboard | 1115 |
| Image and Object Imports..... | 1116 |
| Character Formatting..... | 1117 |
| Paragraph Formatting..... | 1118 |
| Tab Support | 1120 |
| Page Breaks and Repagination..... | 1120 |
| Header, Footers and Bookmarks..... | 1121 |
| Table Commands..... | 1121 |
| Sections and Columns..... | 1123 |
| Stylesheets and Table of Contents..... | 1124 |
| Text/Picture Frame and Drawing Objects..... | 1125 |
| View Options..... | 1125 |
| Navigation Commands..... | 1126 |
| Search and Replace Commands..... | 1126 |
| Highlighting Commands..... | 1127 |
| The Legacy RTF Dialog | 1128 |
| Document a Single Element..... | 1129 |
| The RTF Report Dialog..... | 1129 |
| Set the Main RTF Properties..... | 1130 |
| Apply a Filter..... | 1131 |
| Exclude Elements..... | 1132 |
| RTF Diagram Format..... | 1132 |
| Model Include..... | 1133 |
| RTF Options..... | 1133 |
| RTF Selections..... | 1134 |
| Generate the Report..... | 1135 |
| Legacy RTF Style Templates..... | 1135 |
| Diagram Only Report..... | 1137 |
| Report Templates..... | 1138 |
| Include or Exclude a Package from Report..... | 1138 |
| Save as Document..... | 1139 |
| Custom Language Settings..... | 1140 |
| Using MS Word | 1142 |
| Open a Report in Microsoft Word..... | 1142 |
| Changing Linked Images to Embedded Images..... | 1142 |
| Bookmarks..... | 1143 |
| Other Features of Word..... | 1145 |
| Add Table of Contents..... | 1145 |
| Add Table of Figures..... | 1146 |
| Add Headers and Footers..... | 1147 |
| Manipulating Tables in Word..... | 1148 |
| Refresh Links..... | 1150 |
| Other Documents | 1151 |
| Dependency Report..... | 1151 |
| Implementation Report..... | 1152 |
| Set Target Types Dialog..... | 1154 |
| Testing Report..... | 1154 |

| | |
|--|-------------|
| 2 HTML Reports | 1155 |
| Creating an HTML Report | 1155 |
| The Generate HTML Report Dialog | 1156 |
| Making the Output Available on the Web | 1158 |
| Web Style Templates | 1158 |
| 3 Virtual Documents | 1160 |
| Create a Document Object | 1161 |
| Add Packages to Your Document Object | 1161 |
| Rearrange the Package Order | 1162 |
| Delete a Package from Your Document Object | 1163 |
| Generate the Document | 1164 |

Part XV Automation and Scripting 1167

| | |
|---|-------------|
| 1 The Automation Interface | 1167 |
| Using the Automation Interface | 1168 |
| Connecting to the Interface..... | 1168 |
| Set Up VB | 1169 |
| Examples and Tips..... | 1171 |
| Call from EA | 1172 |
| Available Resources..... | 1173 |
| Reference | 1173 |
| Interface Overview..... | 1174 |
| App | 1177 |
| Enumerations..... | 1178 |
| MDGMenus Enum..... | 1178 |
| EnumRelationSetType Enum..... | 1178 |
| ConstLayoutStyles Enum..... | 1178 |
| ObjectType Enum..... | 1179 |
| XMIType Enum..... | 1179 |
| PropType Enum..... | 1180 |
| Repository..... | 1180 |
| Repository | 1182 |
| Author | 1190 |
| Client | 1191 |
| Collection | 1191 |
| Datatype | 1193 |
| EventProperties..... | 1194 |
| EventProperty..... | 1194 |
| Package | 1194 |
| ProjectIssues..... | 1198 |
| ProjectResource..... | 1199 |
| Property Type..... | 1200 |
| Reference | 1201 |
| Stereotype | 1201 |
| Task | 1202 |
| Term | 1203 |
| Element..... | 1204 |
| Constraint | 1206 |
| Effort | 1206 |
| Element | 1207 |
| File | 1211 |
| Issue (Maintenance)..... | 1212 |
| Metric | 1213 |

| | |
|---|-------------|
| Requirement | 1213 |
| Resource | 1214 |
| Risk | 1215 |
| Scenario | 1216 |
| TaggedValue..... | 1216 |
| Test | 1217 |
| Element Features..... | 1218 |
| Attribute | 1219 |
| Attribute Constraint..... | 1220 |
| Attribute Tag | 1221 |
| Method | 1221 |
| Method Constraint..... | 1223 |
| Method Tag | 1223 |
| Parameter | 1224 |
| Partitions | 1225 |
| Embedded Elements..... | 1225 |
| Transitions | 1226 |
| Custom Properties..... | 1227 |
| Properties | 1227 |
| Connector..... | 1228 |
| Connector | 1229 |
| Connector Constraints..... | 1230 |
| Connector End..... | 1231 |
| Connector Tag..... | 1232 |
| Role Tag | 1233 |
| Diagram..... | 1234 |
| Diagram | 1235 |
| Diagram Links..... | 1236 |
| Diagram Objects..... | 1237 |
| SwimlaneDef..... | 1238 |
| Swimlanes | 1238 |
| Swimlane | 1239 |
| Project Interface..... | 1239 |
| Project | 1240 |
| Code Samples..... | 1247 |
| Open the Repository..... | 1247 |
| Iterate Through an EAP File..... | 1248 |
| Add and Manage Packages..... | 1248 |
| Add and Manage Elements..... | 1249 |
| Add a Connector..... | 1249 |
| Add and Manage Diagrams..... | 1250 |
| Adding and Deleting Attributes and Methods..... | 1251 |
| Element Extras..... | 1252 |
| RepositoryExtras..... | 1254 |
| Stereotypes | 1256 |
| Working with Attributes..... | 1256 |
| Working with Methods..... | 1257 |
| 2 The Project Interface (XML) | 1258 |
| The Read Only Interface | 1259 |
| Automation Interfaces | 1259 |
| Opening a Project | 1260 |
| Enumerating Views | 1260 |
| Enumerating | 1262 |
| Enumerating Diagrams | 1262 |

| | |
|--------------------------------------|-------------|
| Get Diagrams | 1263 |
| Enumerating Elements | 1265 |
| Get Element | 1266 |
| Enumerating Links | 1274 |
| Get Link | 1275 |
| Get Diagram Image | 1277 |
| 3 Add-Ins | 1277 |
| Overview | 1278 |
| Add-In Tasks | 1278 |
| Creating Add-Ins | 1279 |
| Defining Menu Items | 1279 |
| Deploying Add-Ins | 1280 |
| Tricks and Traps | 1281 |
| Add-In Search | 1283 |
| XML Format (Search Data) | 1283 |
| Add-In Events | 1284 |
| EA_Connect | 1284 |
| EA_Disconnect | 1285 |
| EA_GetMenuItems | 1285 |
| EA_GetMenuState | 1286 |
| EA_MenuClick | 1287 |
| EA_OnOutputItemClicked | 1287 |
| EA_OnOutputItemDoubleClicked | 1288 |
| EA_ShowHelp | 1289 |
| Broadcast Events | 1290 |
| EA_FileOpen | 1290 |
| EA_FileClose | 1291 |
| EA_OnPreDeleteElement | 1291 |
| EA_OnPreDeleteConnector | 1292 |
| EA_OnPreDeleteDiagram | 1293 |
| EA_OnPreDeletePackage | 1294 |
| EA_OnPreDeleteTechnology | 1294 |
| EA_OnPreNewElement | 1295 |
| EA_OnPreNewConnector | 1296 |
| EA_OnPreNewAttribute | 1297 |
| EA_OnPreNewMethod | 1298 |
| EA_OnPreNewPackage | 1299 |
| EA_OnPostNewElement | 1300 |
| EA_OnPostNewConnector | 1300 |
| EA_OnPostNewAttribute | 1301 |
| EA_OnPostNewMethod | 1302 |
| EA_OnPostNewPackage | 1303 |
| EA_OnDeleteTechnology | 1304 |
| EA_OnImportTechnology | 1304 |
| EA_OnContextItemChanged | 1305 |
| EA_OnContextItemDoubleClicked | 1306 |
| EA_OnNotifyContextItemModified | 1307 |
| EA_OnPostTransform | 1308 |
| Custom Views | 1308 |
| Creating a Custom View | 1309 |
| MDG Add-Ins | 1309 |
| MDG Events | 1310 |
| MDG Add-Ins MDGBuild Project | 1310 |
| MDG Add-Ins MDGConnect | 1311 |

| | |
|--|------|
| MDG Add-Ins MDGDisconnect..... | 1312 |
| MDG Add-Ins MDGGetConnectedPackages..... | 1313 |
| MDG Add-Ins MDGGetProperty..... | 1313 |
| MDG Add-Ins MDGMerge..... | 1314 |
| MDG Add-Ins MDGNewClass..... | 1316 |
| MDG Add-Ins MDGPostGenerate..... | 1317 |
| MDG Add-Ins MDGPostMerge..... | 1317 |
| MDG Add-Ins MDGPreGenerate..... | 1318 |
| MDG Add-Ins MDGPreMerge..... | 1319 |
| MDG Add-Ins MDGPreReverse..... | 1320 |
| MDG Add-Ins MDGRunExe..... | 1320 |
| MDG Add-Ins MDGView..... | 1321 |

Part XVI Glossary of Terms 1324

| | |
|------------------------------|-------------|
| 1 A (Glossary) | 1324 |
| 2 B (Glossary) | 1326 |
| 3 C (Glossary) | 1327 |
| 4 D (Glossary) | 1331 |
| 5 E (Glossary) | 1333 |
| 6 F (Glossary) | 1334 |
| 7 G (Glossary) | 1335 |
| 8 H (Glossary) | 1336 |
| 9 I (Glossary) | 1336 |
| 10 J (Glossary) | 1338 |
| 11 L (Glossary) | 1338 |
| 12 M (Glossary) | 1338 |
| 13 N (Glossary) | 1341 |
| 14 O (Glossary) | 1341 |
| 15 P (Glossary) | 1342 |
| 16 Q (Glossary) | 1345 |
| 17 R (Glossary) | 1345 |
| 18 S (Glossary) | 1347 |
| 19 T (Glossary) | 1351 |
| 20 U (Glossary) | 1353 |
| 21 V (Glossary) | 1354 |

Part XVII Acknowledgements 1356

Index 1357

Foreword

This user guide provides an introduction to the features contained in Enterprise Architect 6.5 - a UML CASE tool for developing and building software systems with UML.

Part



1 Introduction



Welcome to the Sparx Systems Enterprise Architect User Guide. This guide is intended to get you up to speed with software modeling, construction and management using UML and the features of Enterprise Architect, a UML based CASE tool.

Quick Start

- [What is EA?](#)
- [Getting Started](#)
- [Using Enterprise Architect](#)
- [The UML Language](#)
- [Modeling with UML](#)
- [MDA Transforms](#)
- [Model Management](#)
- [Project Management](#)
- [Code Engineering](#)
- [Build and Run](#)
- [XML Technologies](#)
- [Data Modeling](#)
- [Creating Documents](#)
- [Automation and Scripting](#)
- [Glossary](#)

See Also

- [Enterprise Architect Features](#)
- [Copyright Notice](#)
- [Trademarks](#)
- [Differences Between Editions](#)
- [License Agreement](#)
- [Ordering Enterprise Architect](#)
- [Support](#)
- [Available Helpfile Formats](#)
- [Your Feedback](#)
- [Zicom Mentor](#)
- [Acknowledgements](#)

1.1 What is Enterprise Architect?

Enterprise Architect is a CASE (Computer Aided Software Engineering) tool for the design and construction of software systems. EA supports the [UML 2.0](#) specification, which describes a visual language by which maps or models of a project can be defined.

EA is a progressive tool that covers all aspects of the development cycle, providing full traceability from initial design phase through to deployment and maintenance. It also provides support for testing, maintenance and change control.

Some of the key features of Enterprise Architect are:

- Create UML model elements for a wide range of purposes
- Place those elements in diagrams and packages

- Create connectors between elements
- Document the elements you have created
- Generate code for the software you are building
- Reverse engineer existing code in several languages

Using EA, you can forward and reverse engineer ActionScript, C++, C#, Delphi, Java, Python, PHP, VB.NET and Visual Basic classes, synchronize code and model elements, and design and generate database elements. High quality documentation can be quickly exported from your models in industry standard .RTF format and imported into Word for final customization and presentation.

Enterprise Architect supports all UML 2.0 models/diagrams. You can model business processes, web sites, user interfaces, networks, hardware configurations, messages and more. Estimate the size of your project work effort in hours. Capture and trace requirements, resources, test plans, defects and change requests. From initial concept to maintenance and support, Enterprise Architect has the features you need to design and manage your development and implementation.

Note: UML is an open modeling standard, defined and maintained by the Object Management Group. For full details on UML, including the current UML specification documents, visit <http://www.omg.org> and follow the links.

Tip: Users unfamiliar with UML should take the time to fully explore this user guide and the example project supplied with Enterprise Architect. The online [UML Tutorial](#) will also be of value.

1.2 EA Features

Enterprise Architect is a powerful means by which to specify, document and build your software projects. Using UML notation and semantics, you can design and model complex software systems from the ground up. Use Enterprise Architect to generate and reverse engineer source code in a variety of languages, import database designs from ODBC data source, and import and export models using industry standard XMI.

Enterprise Architect Features

- Model complex software and hardware systems in UML-compliant notation
- Generate and reverse engineer Actionscript, C++, C#, Delphi, Java, PHP, Python, Visual Basic and VB.NET (Professional and Corporate Editions only)
- Model databases and generate DDL scripts. Reverse database schema from ODBC connections (Professional and Corporate Editions only)
- Manage change, maintenance, test scripts and more
- Model dependencies between elements
- Set object classifiers
- Model system dynamics and state
- Model class hierarchies
- Model deployment, components and implementation details
- Collect project issues, tasks and system glossary
- Assign resources to model elements and track effort expended versus required effort
- Output detailed and quality documentation in RTF and HTML formats
- Output models in XMI 1.0, XMI 1.1, XMI 1.2 and XMI 2.1 compatible format for import or export to other XMI compliant tools
- Import models in XMI 1.0, XMI 1.1, XMI 1.2 and XMI 2.1 format from other tools
- Version Control through XMI using SCC, CVS and Subversion version control configurations.
- UML Profiles are available to create custom modeling extensions
- Save and Load complete diagrams as UML Patterns
- Show links between elements in tabular format using the Relationship Matrix
- Script and work with the UML elements and automate common tasks using a detailed Automation Interface
- Connect to SQL Server, MySQL or Oracle 9i and 10g databases (Corporate edition only)
- Migrate changes across a distributed environment with JET Replication
- Use Controlled Packages based on XMI import/export
- Perform MDA Style Transforms

Plus much more!

Tip: The EA Online User Manual is best viewed with Internet Explorer Version 4 or higher.

1.3 Copyright Notice

Copyright © 1998-2006 Sparx Systems Pty. Ltd. All rights reserved.

The software contains proprietary information of Sparx Systems Pty Ltd. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. Please read the [license agreement](#) for full details.

Due to continued product development, this information may change without notice. The information and intellectual property contained herein is confidential between Sparx Systems and the client and remains the exclusive property of Sparx Systems. If you find any problems in the documentation, please report them to us in writing. Sparx Systems does not warrant that this document is error-free. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Sparx Systems. Licensed users are granted the right to print a single hardcopy of the user manual per licensed copy of the software, but may not sell, distribute or otherwise dispose of the hardcopy without written consent of Sparx Systems.

Sparx Systems Pty. Ltd.
7 Curtis St,
Creswick, Victoria 3363,
AUSTRALIA

Phone: +61 (3) 5345 1140
Fax: +61 (3) 5345 1104

Support Email: support@sparxsystems.com.au
Sales Email: sales@sparxsystems.com.au

Website: www.sparxsystems.com.au

1.4 Trademarks

Acknowledgement of Trademarks

The following are Trademarks of Microsoft

- Microsoft Word
- Microsoft Office
- Windows®
- ActiveX

The following are Registered Trademarks of OMG

- CORBA®
- the OMG Object Management Group logo
- The Information Brokerage®
- CORBA Academy®
- IIOP®
- XMI®

The following are Trademarks of the OMG

- OMG™
- Object Management Group™
- the CORBA logo
- ORB™
- Object Request Broker™
- the CORBA Academy design
- OMG Interface Definition Language™
- IDL™
- CORBAservices™
- CORBAfacilities™
- CORBAmed™
- CORBAnet™
- Unified Modeling Language™
- UML™
- the UML Cube logo
- MOF™
- CWM™
- Model Driven Architecture™
- MDA™
- OMG Model Driven Architecture™
- OMG MDA™

1.5 Differences Between Desktop, Professional and Corporate Editions

Enterprise Architect is available in three editions, the **Desktop**, **Professional** and **Corporate**. The desktop edition does not support the code generation functions of the Professional and Corporate editions. The Professional Edition has all of the features available in the Desktop edition with the addition of code engineering and model sharing. The Corporate Edition has all of the capabilities of the Desktop and Professional Editions but adds the ability to connect to MySQL, SQL Server, PostgreSQL, Sybase Adaptive Server Anywhere and Oracle 9i and 10g DBMS back ends as the shared repository. This edition also supports user security, user logins, user groups and user level locking of elements.

Functionality for each version is described below:

| Functionality | Corporate Edition | Professional Edition | Desktop Edition |
|--|--------------------------|-----------------------------|------------------------|
| .EAP Files | Yes | Yes | Yes |
| Shared Models | Yes | Yes | No |
| Source Code Engineering | Yes | Yes | No |
| Database Engineering | Yes | Yes | No |
| Microsoft Access Repository | Yes | Yes | Yes |
| SQL Server, MySQL, Oracle 9i and 10g, PostgreSQL, MSDE, Adaptive Server Anywhere Database Repositories | Yes | No | No |
| Version Control | Yes | Yes | Yes |
| Replication | Yes | Yes | No |
| MDG Technologies | Yes | Yes | No |
| MDG Link for Eclipse and MDG Link for Visual Studio.NET | Yes | Yes | No |
| Security | Yes | No | No |

EA Desktop Edition

The Desktop edition is targeted at single developers producing UML analysis and design models. This

edition includes all the features of the Professional edition except code engineering (import/export of source code and DDL), the Active-X interface and the ability to share a model amongst multiple users.

EA Professional Edition

Aimed at work groups and developers, the Professional edition supports shared projects through replication and shared network files. This edition has an ActiveX interface for interrogating EA projects and extracting information in XML format. The Professional edition also fully supports code import/export and synchronization of model elements with source code and reverse engineering Oracle 9i and 10g, SQL Server and MS Access databases. Support for MDG Technologies and MDG Link (sold separately) is included with the Professional version of EA. The shared repository available in the Professional Edition is restricted to the .EAP file format (JET database).

EA Corporate Edition

Aimed at larger development teams, the Corporate edition supports everything in the Desktop and Professional versions, plus the ability to connect to MySQL, SQL Server, PostgreSQL, Sybase Adaptive Server Anywhere and Oracle 9i and 10g DBMS back ends as the shared repository. This provides additional scalability and improved concurrency over the shared .EAP file approach to model sharing. Support for MDG Technologies and MDG Link (sold separately) is included with the Corporate version of EA. This edition also supports user security, user logins, user groups and user level locking of elements. The Corporate edition support for user/group based security - with locking at diagram and element levels. Security comes in two modes: in the first mode, all elements are considered 'writeable' until explicitly locked by a user or group; in mode two, all elements are considered locked until checked out with a user lock.

***Tip:** In order to help you understand the differences between these editions and the advantages and limitations of each, the trial version of Enterprise Architect can be opened in any desired configuration. When EA starts, select the mode you wish to trial - you can restart in another mode next time if desired.*

The fully functional 30 day trial version of EA is available free of charge at www.sparxsystems.com.au/bin/easetup.exe.

See Also

[Ordering Enterprise Architect](#)

1.6 EA Software Product License Agreement

SOFTWARE PRODUCT LICENSE AGREEMENT

Enterprise Architect - UML CASE Tool - Desktop, Professional and Corporate Editions, Version 6.5
Copyright (C) 1998-2006 Sparx Systems Pty Ltd. All Rights Reserved

IMPORTANT-READ CAREFULLY: This End User Licence Agreement ("EULA") is a legal agreement between YOU as Licensee and SPARX for the SOFTWARE PRODUCT identified above. By installing, copying, or otherwise using the SOFTWARE PRODUCT, YOU agree to be bound by the terms of this EULA. If YOU do not agree to the terms of this EULA, promptly return the unused SOFTWARE PRODUCT to the place of purchase for a full refund.

The copyright in the SOFTWARE PRODUCT and its documentation is owned by Sparx Systems Pty Ltd A.B.N 38 085 034 546. Subject to the terms of this EULA, YOU are granted a non-exclusive right for the duration of the EULA to use the SOFTWARE PRODUCT. YOU do not acquire ownership of copyright or other intellectual property rights in any part of the SOFTWARE PRODUCT by virtue of this EULA.

Your use of this software indicates your acceptance of this EULA and warranty.

DEFINITIONS

In this End User Licence Agreement, unless the contrary intention appears, "ACADEMIC EDITION" means an edition of the Software Product purchased for educational purposes at an

academic discount price.

"EULA" means this End User Licence Agreement

"SPARX" means Sparx Systems Pty Ltd A.B.N 38 085 034 546

"Licensee" means YOU, or the organisation (if any) on whose behalf YOU are taking the EULA.

"Registered Edition of Enterprise Architect" means the edition of the SOFTWARE PRODUCT which is available for purchase from the web site: <http://www.sparxsystems.com.au/ea_purchase.htm>. following the thirty day free evaluation period.

"SOFTWARE PRODUCT" or "SOFTWARE" means Enterprise Architect, UML Case Tool, Desk top, Professional and Corporate editions, which includes computer software and associated media and printed materials, and may include online or electronic documentation.

"Support Services" means email based support provided by SPARX, including advice on usage of Enterprise Architect, investigation of bugs, fixes, repairs of models if and when appropriate and general product support.

"SPARX support engineers" means employees of SPARX who provide on-line support services.

"Trial edition of Enterprise Architect" means the edition of the SOFTWARE PRODUCT which is available free of charge for evaluation purposes for a period of 30 days.

"EA LITE" means the LITE version of Enterprise Architect that is distributed free of charge as a read-only viewer of .EAP files.

GRANT OF LICENCE

In accordance with the terms of this EULA YOU are granted the following rights:

a) to install and use one copy of the SOFTWARE PRODUCT, or in its place, any prior version for the same operating system, on a single computer. As the primary user of the computer on which the SOFTWARE PRODUCT is installed, YOU may make a second copy for your exclusive use on either a home or portable computer.

b) to store or install a copy of the SOFTWARE PRODUCT on a storage device, such as a network server, used only to install or run the SOFTWARE PRODUCT over an internal network. If YOU wish to increase the number of users entitled to concurrently access the SOFTWARE PRODUCT, YOU must notify SPARX and agree to pay an additional fee.

c) to make copies of the SOFTWARE PRODUCT for backup and archival purposes.

EVALUATION LICENCE

The Trial version of Enterprise Architect is not free software. Subject to the terms of this agreement, YOU are hereby licensed to use this software for evaluation purposes without charge for a period of 30 days.

Upon expiration of the 30 days, the Software Product must be removed from the computer. Unregistered use of Enterprise Architect after the 30-day evaluation period is in violation of Australian, U.S. and international copyright laws.

SPARX may extend the evaluation period on request and at their discretion.

If YOU choose to use this software after the 30 day evaluation period a licence must be purchased (as described at http://www.sparxsystems.com.au/ea_purchase.htm). Upon payment of the licence fee, YOU will be sent details on where to download the registered edition of Enterprise Architect and will be provided with a suitable software 'key' by email.

EA LITE

Subject to the terms of this Agreement EA LITE may be installed on any machine indefinitely and free of charge. There are no fees or Sparx support services in relation to EA LITE.

ADDITIONAL RIGHTS AND LIMITATIONS

YOU hereby undertake not to sell, rent, lease, translate, adapt, vary, modify, decompile, disassemble, reverse engineer, create derivative works of, modify, sub-licence, loan or distribute the SOFTWARE PRODUCT other than as expressly authorised by this EULA.

YOU further undertake not to reproduce or distribute licence key-codes except under the express and written permission of SPARX .

If the Software Product purchased is an Academic Edition, YOU ACKNOWLEDGE THAT the licence is limited to use in an educational context, either for self-education or use in a registered teaching institution. The Academic Edition may not be used to produce commercial software products or be used in a commercial environment, without the express written permission of SPARX.

ASSIGNMENT

YOU may only assign all your rights and obligations under this EULA to another party if YOU supply to the

transferee a copy of this EULA and all other documentation including proof of ownership. Your licence is then terminated.

TERMINATION

Without prejudice to any other rights, SPARX may terminate this EULA if YOU fail to comply with the terms and conditions. Upon termination YOU or YOUR representative shall destroy all copies of the SOFTWARE PRODUCT and all of its component parts or otherwise return or dispose of such material in the manner directed by SPARX.

WARRANTIES AND LIABILITY

WARRANTIES

SPARX warrants that the SOFTWARE PRODUCT will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt, and any Support Services provided by SPARX shall be substantially as described in applicable written materials provided to YOU by SPARX, and SPARX support engineers will make commercially reasonable efforts to solve any problems associated with the SOFTWARE PRODUCT.

EXCLUSIONS

To the maximum extent permitted by law, SPARX excludes, for itself and for any supplier of software incorporated in the SOFTWARE PRODUCT, all liability for all claims, expenses, losses, damages and costs made against or incurred or suffered by YOU directly or indirectly (including without limitation lost costs, profits and data) arising out of:

YOUR use or misuse of the SOFTWARE PRODUCT;

YOUR inability to use or obtain access to the SOFTWARE PRODUCT;

Negligence of SPARX or its employees, contractors or agents, or of any supplier of software incorporated in the SOFTWARE PRODUCT, in connection with the performance of SPARX'S obligations under this EULA; or

Termination of this EULA by either party for any reason.

LIMITATION

The SOFTWARE PRODUCT and any documentation are provided "AS IS" and all warranties whether express, implied, statutory or otherwise, relating in any way to the subject matter of this EULA or to this EULA generally, including without limitation, warranties as to: quality, fitness; merchantability; correctness; accuracy; reliability; correspondence with any description or sample, meeting your or any other requirements; uninterrupted use; compliance with any relevant legislation and being error or virus free are excluded. Where any legislation implies in this EULA any term, and that legislation avoids or prohibits provisions in a contract excluding or modifying such a term, such term shall be deemed to be included in this EULA. However, the liability of SPARX for any breach of such term shall if permitted by legislation be limited, at SPARX'S option to any one or more of the following upon return of the SOFTWARE PRODUCT and a copy of the receipt:

If the breach relates to the SOFTWARE PRODUCT:

the replacement of the SOFTWARE PRODUCT or the supply of an equivalent SOFTWARE PRODUCT;

the repair of such SOFTWARE PRODUCT; or the payment of the cost of replacing the SOFTWARE PRODUCT or of acquiring an equivalent SOFTWARE PRODUCT; or

the payment of the cost of having the SOFTWARE PRODUCT repaired;

If the breach relates to services in relation to the SOFTWARE PRODUCT:

the supplying of the services again; or

the payment of the cost of having the services supplied again.

TRADEMARKS

All names of products and companies used in this EULA, the SOFTWARE PRODUCT, or the enclosed documentation may be trademarks of their corresponding owners. Their use in this EULA is intended to be in compliance with the respective guidelines and licences. Windows, Windows 95, Windows 98, Windows NT, Windows ME, Windows XP and Windows 2000 are trademarks of Microsoft.

GOVERNING LAW

This agreement shall be construed in accordance with the laws of the Commonwealth of AUSTRALIA.

1.7 Ordering Enterprise Architect

Enterprise Architect is designed, built and published by Sparx Systems and available from [Sparx Systems](#).

The trial version of EA is identical to the registered edition with the exception that all diagrams are output to files with an embedded watermark. The trial software will stop working after the trial period has elapsed. On purchase of a suitable license(s), the registered version will be made available for download.

The latest information on pricing and purchasing is available at: [Sparx Systems Purchase/Pricing Website](#).

Purchase Options

- On-line using a secure credit-card transaction. See: [Pricing and Purchase Options](#).
- Fax
- Check or equivalent
- Bank transfer

For more information, contact sales@sparxsystems.com.au.

1.8 Support

Support is available to Registered Users of Enterprise Architect. All support issues are currently dealt with via email. Sparx Systems endeavour to provide a rapid response to all questions and concerns regarding Enterprise Architect.

You can contact the support team at support@sparxsystems.com.au.

An online user forum is also available for your questions and perusal, at www.sparxsystems.com.au/cgi-bin/yabb/YaBB.cgi.

1.9 Available Helpfile Formats

You can access the latest EA helpfiles from the following locations:

- **.CHM** format: www.sparxsystems.com.au/bin/EA.chm
- **.CHM** format inside a **.ZIP** file: www.sparxsystems.com.au/bin/EAHelp.zip
- **.PDF** format: www.sparxsystems.com.au/bin/EASUserGuide.pdf
- **.HTML** format: www.sparxsystems.com.au/EASUserGuide/index.html

Version and release date information for the helpfiles can be found at

- www.sparxsystems.com.au/ea_downloads.htm#Helpfiles
- or
- www.sparxsystems.com.au/registered/reg_ea_down.htm#Helpfiles (registered users)

1.10 Your Feedback

Sparx Systems like to stay in touch with what Enterprise Architect users need to accomplish their tasks efficiently and effectively. We value any suggestions, feedback and comments you may have regarding this product, the documentation or install process.

You can access our online feedback pages at

- www.sparxsystems.com.au/bug_report.htm and
- www.sparxsystems.com.au/feature_request.htm

Alternatively, you can contact Sparx Systems by email at: support@sparxsystems.com.au.

1.11 EA for Power Users

Need to get started with EA and UML modeling and are not sure how to begin? CD ROM based Enterprise Architect for Power Users provides a comprehensive, yet easy to access tutorial on a wide range of advanced capabilities of the Enterprise Architect visual modeling software.

Topics include a discussion of how to set up EA in a variety of single and multi-user environments, and strategies to support projects of any size.

There is a tutorial on UML 2.0 that explains what's new with this major new release of the UML standard, which is supported in Enterprise Architect.

EA's Data Modeling capabilities is explained in detail, and the tutorial features over 30 narrated Step-by-Step demonstrations of specific features of the EA modeling tool.

Also, the cost of the CD is creditable towards onsite Iconix Jumpstart(tm) training. Mention that you purchased the CD from Sparx, and receive an additional discount!

Enhance your productivity with EA for Power Users, see <http://www.sparxsystems.com.au/partners/iconix/iconixcdeafpu.html> for more details.

Part

2

2 Project Roles and EA

Enterprise Architect performs a number of tasks which may be suited to a variety of professions. Depending on your role within a project, the way you use Enterprise Architect will vary. This section describes some common working practices with EA for a range of project roles. There are tools for Business Analysts, Software Developers, Software Architects, Software Engineers, Project Managers, Support Personnel, Testers, Database Administrators, Deployment and Roll out, and Technology developers.

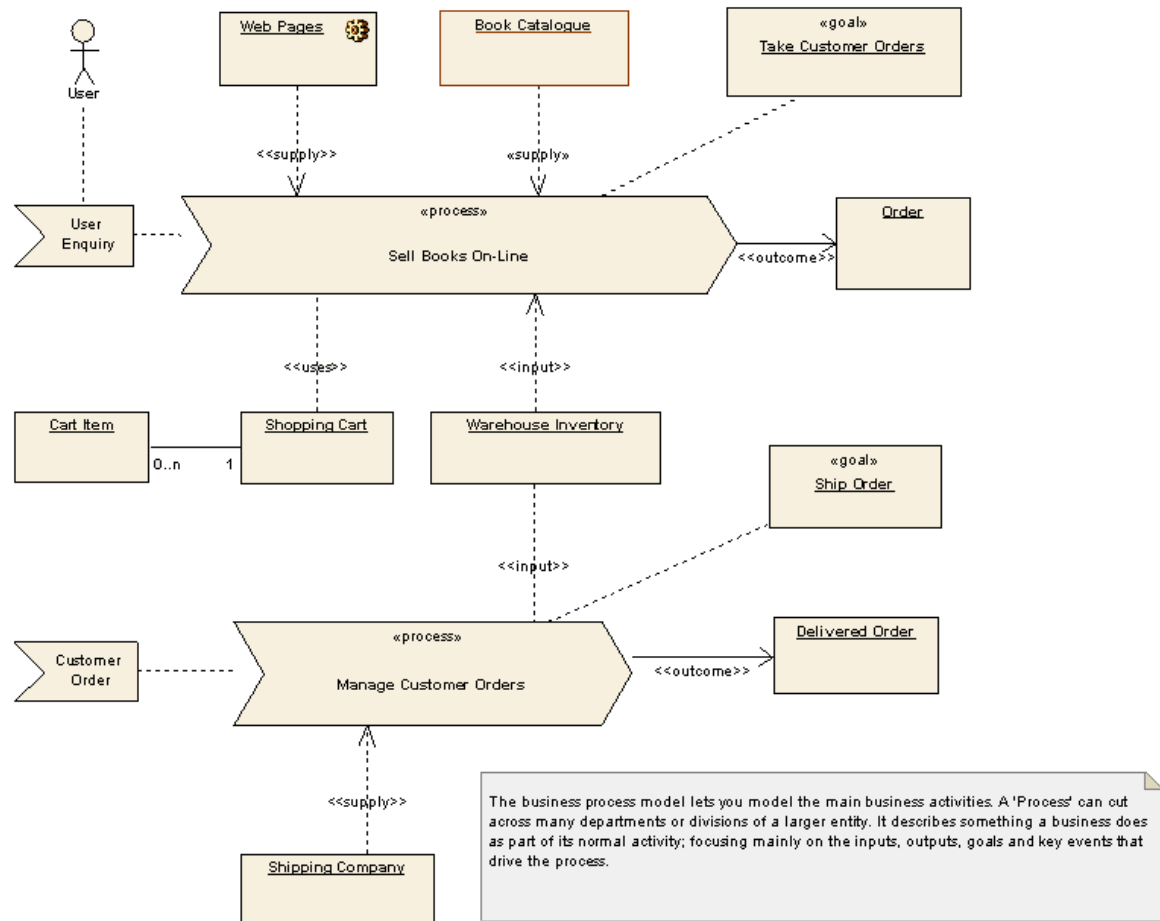
Use the following links to explore how EA can assist you in carrying out your role within a model driven project.

See Also

- [Business Analyst](#)
- [Developer](#)
- [Software Architect](#)
- [Software Engineer](#)
- [Project Manager](#)
- [Testers](#)
- [Database Administrator](#)
- [Deployment and Rollout](#)
- [Technology Developer](#)

2.1 Business Analyst

A Business Analyst may use EA to create high-level models of business processes. These include business requirements, activities, work flow, and the display of system behaviour. Using EA, a Business Analyst can describe the procedures that govern what a particular business does. Such a model is intended to deliver a high-level overview of a proposed system.



Model High Level Business Processes

With Enterprise Architect the Business Analyst can model high level processes of the business with [Analysis Diagrams](#). Analysis Diagrams are a subset of UML 2.0 Activity Diagrams and are less formal than other diagram types, but they provide a useful means for expressing essential business characteristics and needs.

Model Requirements

[Modeling requirements](#) is an important step in the implementation of a project. Enterprise Architect allows users to define the requirement elements, link requirements to the model elements for implementation, link requirements together into a hierarchy, report on requirements, and to move requirements into and out of model element responsibilities.

Model Business Activities

The Business Analyst can use [Activity Diagrams](#) to model the behavior of a system and the way in which these behaviors are related to the overall flow of the system. Activity diagrams do not model the exact internal behavior of the system but show instead the general processes and pathways at a high level.

Model Work Flow

To visualize the cooperation between elements involved in the work flow, the Business Analyst may use an Interaction Overview Diagram. With the Interaction overview diagram the Business Analyst can obtain an overview of sub activities that are involved in a system.

Display System Behavior

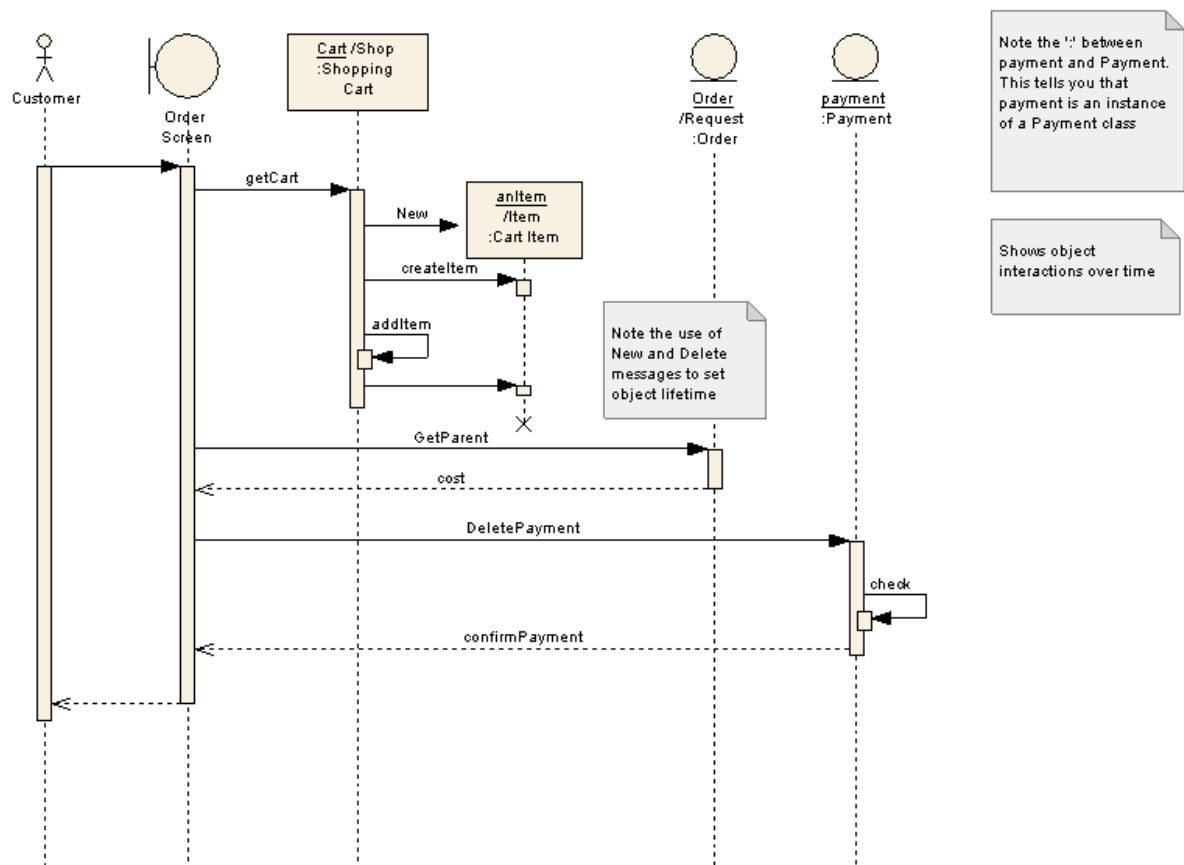
Displaying the behavior of a system as a Use Case gives the Business Analyst an easily understood tool for mapping the functional requirements and behavior of a system.

See Also

- [Business Modeling](#)
- [Analysis Diagrams](#)
- [Requirements](#)
- [Activity Diagrams](#)
- [Interaction Diagrams](#)
- [Use Case Diagram](#)

2.2 Software Architects

Software Architects can use EA to map functional requirements with use cases, perform real time modeling of objects using interaction diagrams, design the deployment model and detail the deliverable components using component diagrams.



Map functional requirements of the system

With Enterprise Architect the Software Architect can take the high level business processes that have been modelled by the Business Analyst and create detailed [Use Cases](#). Use cases are used to describe the proposed functionality of a system and are only used to detail a single unit of discrete work.

Map objects in real time

The Software Architect can use Interaction Diagrams (Sequence and Communication Diagrams) to model the dynamic design of the system. Sequence Diagrams are used to detail the messages that are passed between objects and the lifetimes of the objects. Communication Diagrams are similar to Sequence Diagrams, but are used instead to display the way in which the object interacts with other objects.

Map deployment of objects

The Software Architect can use deployment diagrams to provide a static view of the run-time configuration of processing nodes and the components that run on the nodes. Deployment diagrams can be used to show the connections between hardware, software and any middleware that is used on a system.

Detail deliverable components

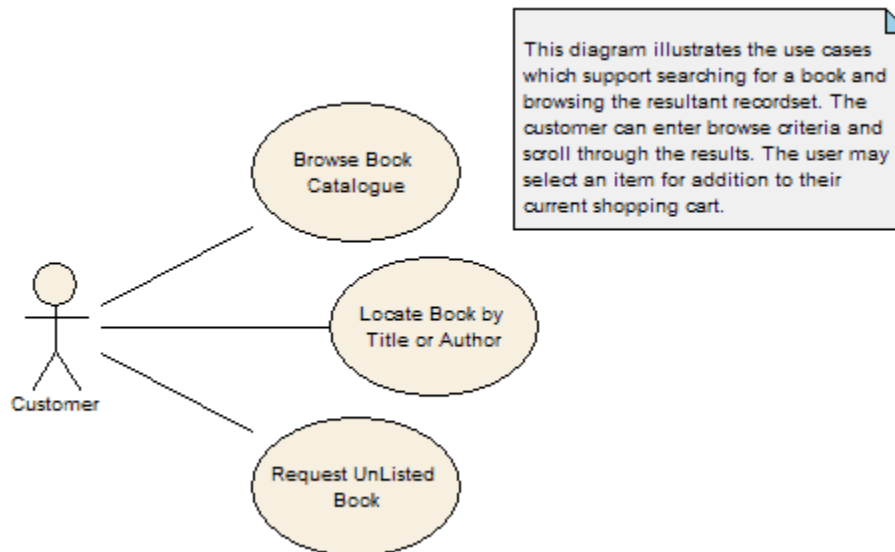
Using Component diagrams allows the Software Architect to model the physical aspects of a system. Components can be executables, libraries, data files or another physical resource that is part of a system. The component model can be developed from scratch from the class model or may be brought in from existing projects and from 3rd party vendors.

See Also

- [Analysis Diagrams](#)
- [Sequence Diagrams](#)
- [Communication Diagrams](#)
- [Deployment Diagrams](#)
- [Component Diagrams](#)

2.3 Software Engineer

Software Engineers using EA can map use cases into class diagrams, detail the interactions between classes, define the system deployment with deployment diagrams and define software packages with package diagrams.



Map Use Cases into Detailed Classes

With Enterprise Architect the Software Engineer can take Use Cases developed by the Software Architect,

and create classes which fulfil the objectives defined in the use cases. A class is one of the standard UML constructs which is used to detail the pattern from which objects will be produced at run time.

Detail Interaction between Classes

Interaction Diagrams (Sequence and Communication Diagrams) allow the Software Engineer to model the dynamic design of the system. Sequence Diagrams are used to detail the messages that are passed between objects and the lifetimes of the objects. Communication Diagrams are similar to Sequence Diagrams, but are used instead to display the way in which objects interact with other objects.

Define System Deployment

Deployment diagrams may be used to provide a static view of the run-time configuration of processing nodes and the components that run on the nodes. Deployment diagrams can be used to show the connections between hardware, software and any middleware that is used on a system, to explain the connections and relationships of the components.

Define Software Packages

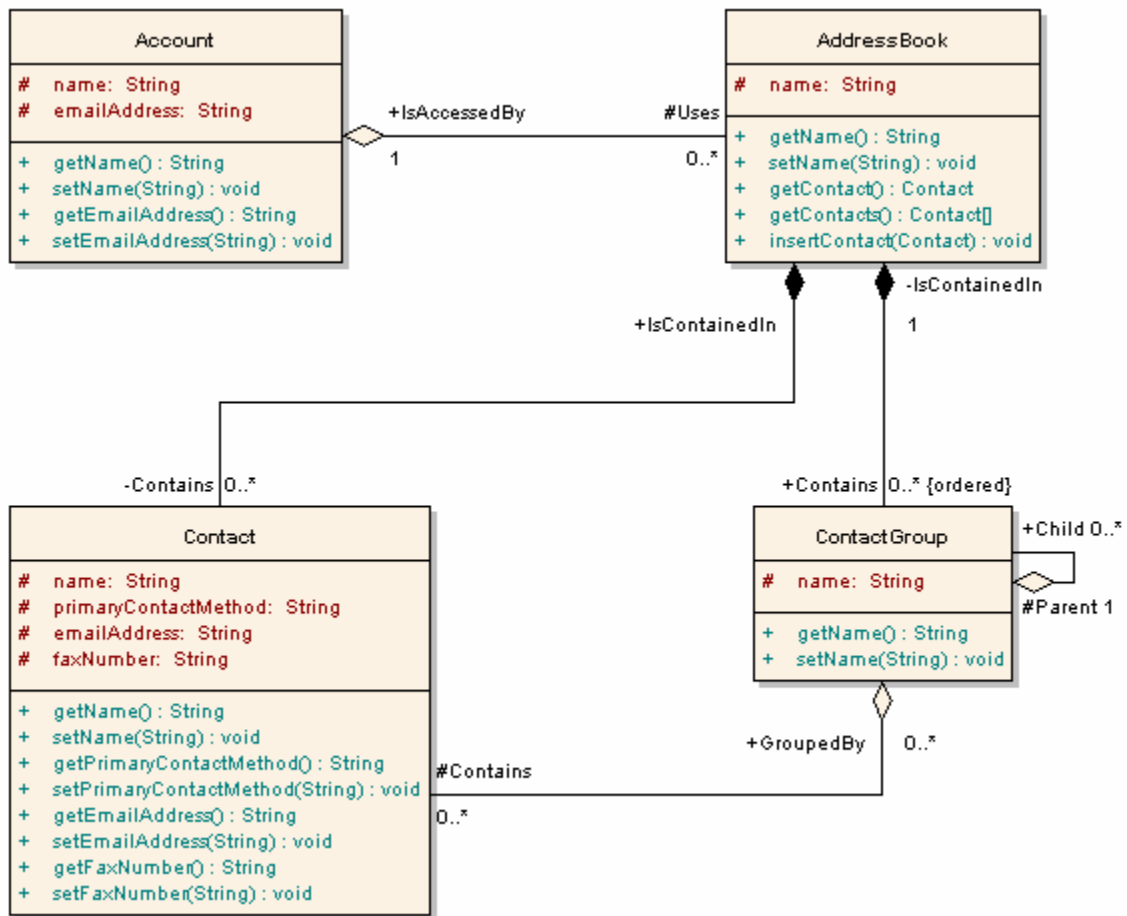
Using Package diagrams allows the Software Engineer to detail the software architecture. Package diagrams are used to organize diagrams and elements into manageable groups declaring the dependencies.

See Also

- [Use Case Diagrams](#)
- [Sequence Diagrams](#)
- [Communication Diagrams](#)
- [Deployment Diagrams](#)
- [Package Diagrams](#)

2.4 Developer

Developers can use EA to perform round trip code engineering which includes reverse engineering of existing code and generation of code from UML class diagrams. State Machine, Package and Activity diagrams can be used by the developer to better understand the interaction between code elements and the arrangement of the code.



Round trip engineering

Enterprise Architect gives the developer unparalleled flexibility, with the ability to round trip engineer software from existing source code to UML 2.0 diagrams and back again. Round trip Engineering involves both forward and reverse engineering of code. Keeping the [model and code synchronized](#) is an important aspect of round trip engineering.

Reverse engineering

EA allows developers to reverse engineer code from a number of supported languages and view the existing code as class diagrams. The developer can use class diagrams to illustrate the static design view of the system. Class diagrams consist of classes and interfaces and the relationships between them. The classes defined in UML class diagrams can have direct counterparts in the implementation of a programming language

Forward engineering

As well as being able to reverse engineer code EA offers the developer the option of forward engineering code (code generation). This allows the developer to make changes to their model with EA and have these changes implemented in the source code.

Determine the system state

To visualize the state of the system the developer can make use of State Machine Diagrams to describe how elements move between states, classifying its behavior, according to transition triggers and constraining guards. State Machine diagrams are used to capture system changes over time, typically being associated with particular classes (often a class may have one or more state machine diagrams used to fully describe its

potential states).

Visualize package arrangement

Package diagrams are used to help design the architecture of the system. They are used to organize the diagrams and elements in manageable groups, and to declare their dependencies.

Follow the flow of code

Activity Diagrams are used to allow for a better understanding of the flow of code. Activity diagrams illustrate the dynamic nature of the system. This allows the modeling of the flow of control between activities and represents the changes in state of the system.

See Also

- [Code Engineering](#)
- [Generate Code](#)
- [Reverse Engineer Code](#)
- [State Machine Diagrams](#)
- [Package Diagrams](#)
- [Activity Diagrams](#)
- [Class Diagrams](#)

2.5 Project Manager

EA provides support for the management of projects. Project Managers can use EA to assign resources to elements, measure risk and effort and estimate project sizes. Change Control and element maintenance is also available.

Set up Technical factors for estimation

Factor Number: Description: Weight: Assigned Value:

TCF04 Complex internal processing 1.00 4.00

Defined Technical Types

| Type | Description | Wei... | Value | Ex Value |
|-------|--|--------|-------|----------|
| TCF01 | Distributed System | 2.00 | 5.00 | 10.00 |
| TCF02 | Response or throughput performa... | 1.00 | 4.00 | 4.00 |
| TCF03 | End user efficiency (online) | 1.00 | 2.00 | 2.00 |
| TCF04 | Complex internal processing | 1.00 | 4.00 | 4.00 |
| TCF05 | Code must be re-usable | 1.00 | 2.00 | 2.00 |
| TCF06 | Easy to install | 0.50 | 5.00 | 2.50 |
| TCF07 | Easy to use | 0.50 | 3.00 | 1.50 |
| TCF08 | Portable | 2.00 | 3.00 | 6.00 |
| TCF09 | Easy to change | 1.00 | 3.00 | 3.00 |
| TCF10 | Concurrent | 1.00 | 2.00 | 2.00 |
| TCF11 | Includ special security features | 1.00 | 2.00 | 2.00 |
| TCF12 | Provide direct access for third parti... | 1.00 | 5.00 | 5.00 |
| TCF13 | Special user training facilities are re... | 1.00 | 3.00 | 3.00 |

Unadjusted TCF:

Provide project estimates

With Enterprise Architect the Project Manager has access to a comprehensive project estimation tool that calculates effort from use case and actor objects, coupled with project configurations defining the technical and environmental complexity of the work environment.

Resource management

Managing the allocation of resources in the design and development of system components is an important and difficult task. Enterprise Architect allows the project manager or development manager the ability to assign resources directly to model elements and track progress over time.

Risk management

The Metrics and Estimations tool may be used to assign Risk to an element within a project. The Risk Types allow the project manager to name the risk, define the type of risk, and give it a weighting.

Maintenance

EA allows the Project Manager to track and assign maintenance related items to elements within EA. This allows for the rapid capture and record keeping with maintenance tasks such as issues, changes, defects and tasks.

See Also

- [Estimation](#)
- [Resources](#)
- [Testing](#)
- [Life Cycle](#)
- [Changes and Defects](#)

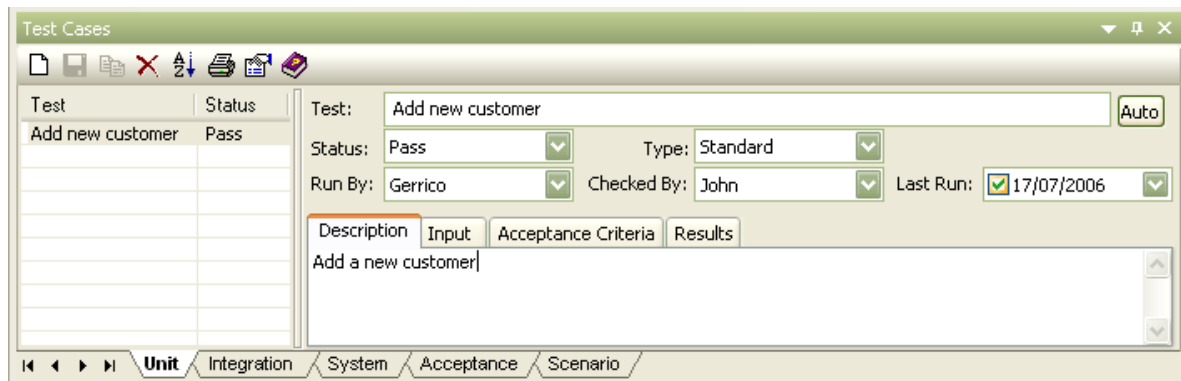
- [Model Tasks List](#)
- [Project and Model Issues](#)
- [Model Glossary](#)
- [Update Package Status](#)
- [Manage Bookmarks](#)

2.6 Testers

Enterprise Architect provides support for design testing by allowing the user to create test scripts against elements in the modeling environment.

Test cases may be assigned to individual model elements, requirements and constraints. Scenarios can be added to model elements, and element defects may be used to report problems associated with model elements.

For more detailed information on testing, consult the [Introduction to Testing in Enterprise Architect](#).



Test cases

With Enterprise Architect, Quality Assurance personnel can set a series of tests for each UML element. The test types include Unit Testing, Acceptance testing, System testing and Scenario Testing.

Import requirements, constraints and scenarios

To help ensure that testing maintains integrity with the entire business process EA offers the tester the ability to import requirements, constraints and scenarios defined in earlier iterations of the development life cycle. Requirements indicate contractual obligations that elements must perform within the model. Constraints are conditions which must be met in order to pass the testing process. Constraints may be Pre-conditions (states which must be true before an event is processed), Post Conditions (events which must occur after the event is processed) or invariant constraints (which must remain true through the duration of the event). Scenarios are textual descriptions of an object's action over time and can be used to describe the way a test works.

Create quality test documentation

EA provides the facility to generate high quality test documentation. EA produces test documentation in the industry standard .RTF file format.

Element defects changes

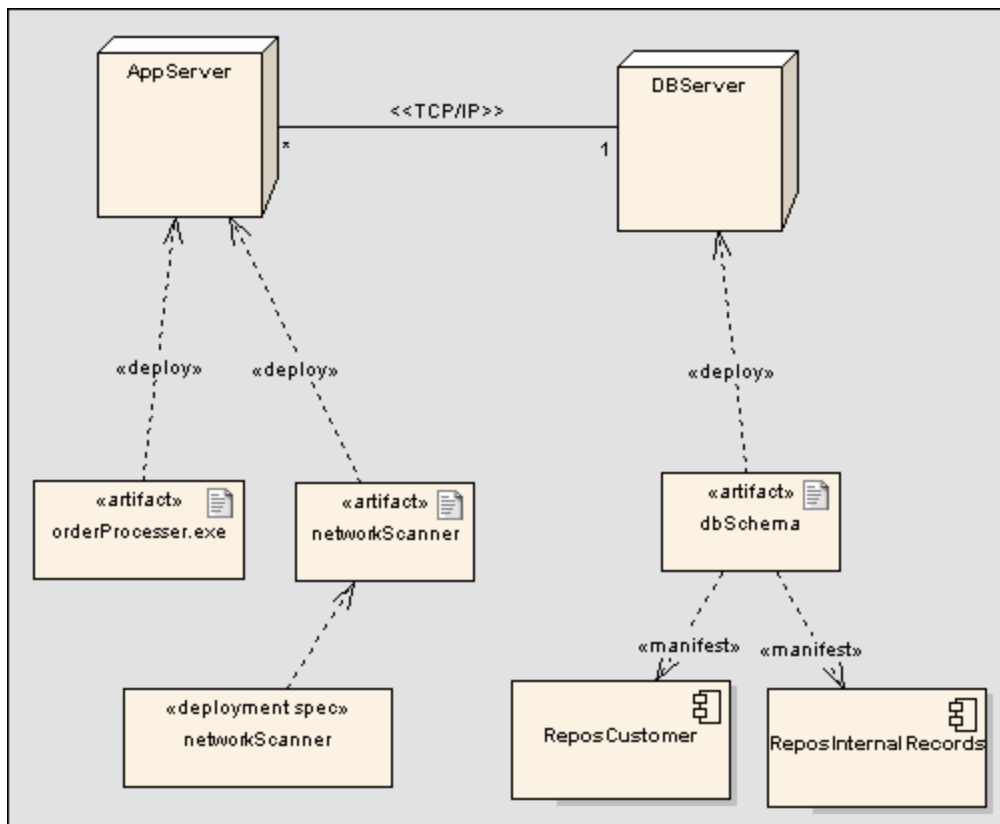
Defects tracking provides the facility to allocate defect reports to any element within the EA model. This allows all who are involved in the project to quickly view the status of defects, to see which defects need to be addressed and which have been dealt with.

See Also

- [Requirements](#)
- [Constraints](#)
- [Defects](#)
- [Introduction to Testing in Enterprise Architect](#)
- [The Testing Workspace](#)
- [The Test Details Dialog](#)
- [Unit Testing](#)
- [Integration Testing](#)
- [System Testing](#)
- [Acceptance Testing](#)
- [Scenario Testing](#)
- [Import Scenario as Test](#)
- [Import Test from other elements](#)
- [Test Details](#)
- [Show Test Scripts in Compartments](#)
- [Test Documentation](#)

2.7 Deployment and Rollout

The tasks involved in the deployment and rollout of a project can be modeled within EA. Network administrators can use deployment diagrams to display the network deployment. Workstation deployment can also be modeled using deployment diagrams. Maintenance tasks can be added to UML elements by users involved in project deployment.



Display Network Deployment

The Network Administrator can use deployment diagrams to provide a static view of the run-time configuration of nodes on the network, and the components that run on the nodes.

Work Station Deployment

By using deployment diagrams it is possible to detail the process of deploying workstations. Deployment Diagrams provide a static view of the run-time configuration of workstations and the components that are used in the workstations.

Maintenance

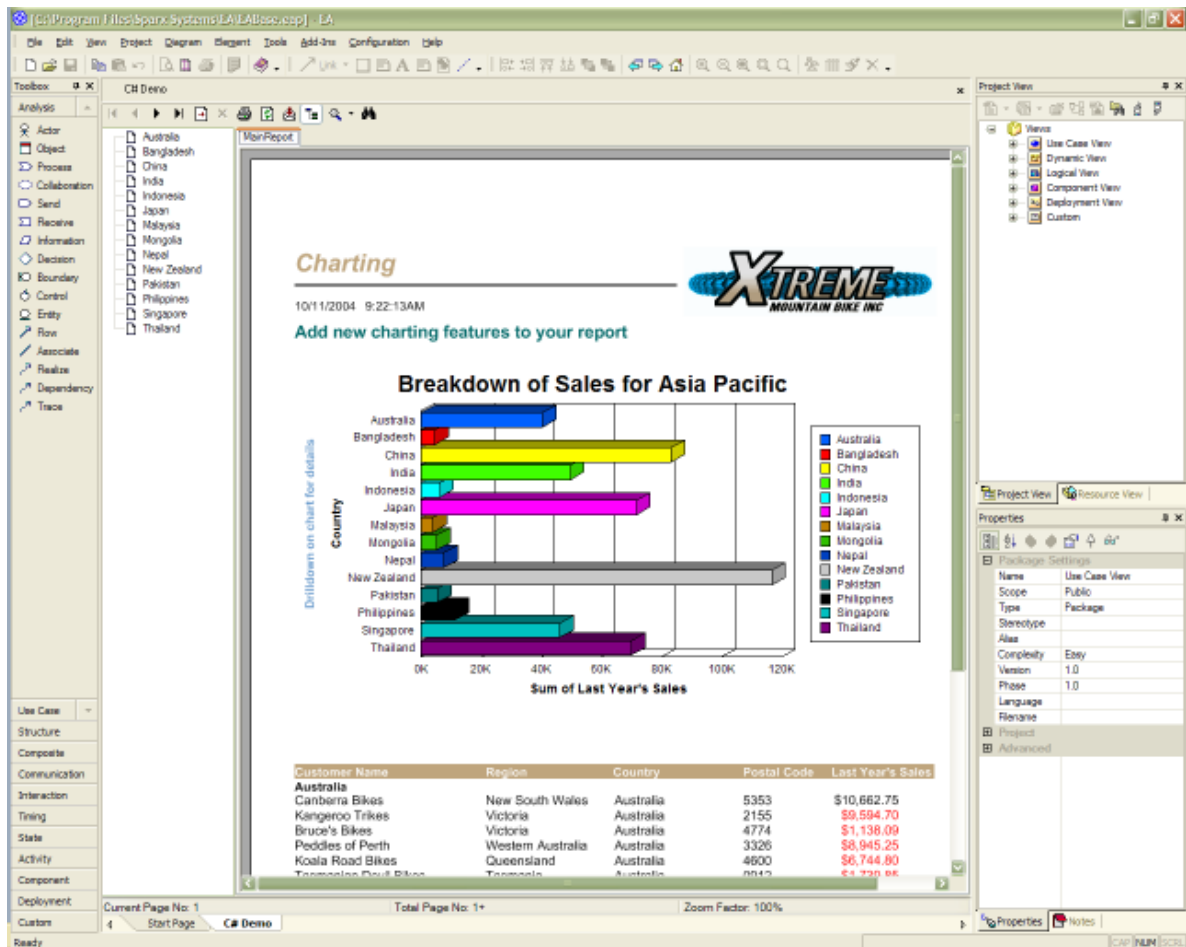
EA allows the user to track and assign maintenance related items to elements within EA. This allows for the rapid capture and record keeping of maintenance tasks such as issues, changes, defects and tasks. By providing a centralized facility for each element involved in the deployment process EA offers a powerful solution for tracing the maintenance of the items and processes involved in system deployment.

See Also

- [Deployment Diagrams](#)
- [Maintenance](#)

2.8 Technology Developer

Technology Developers are users of EA who seek to create customized additions to the functionality that is already present within EA. These additions include UML Profiles, UML Patterns, Code Templates, Tagged Value Types, MDG Technologies and EA Add-ins. By creating these extensions the Technology developer can customize the EA modeling process to specific tasks and speed up the development.



UML Profiles

By creating UML Profiles the technology developer can create a customized extension for building UML models that are specific to a particular domain. Profiles are stored as XML files and may be imported into any model as required.

UML Patterns

Patterns are sets of collaborating objects and classes that provide a generic template for repeatable solutions to modeling problems. As patterns are discovered in any new project, the basic pattern template may be created. Patterns can be re-used with the appropriate variable names modified for any future project.

Code Templates

Code templates are used to customize the output of source code generated by EA. This allows for the generation of code languages not specifically supported by EA and allows the user to define the way EA generates source code to comply with their own company style guidelines.

Tagged Values

Tagged values are used in EA to specify additional information about elements. They are used to extend the information relating to an element outside of the information directly supported by the UML language. Often tagged values are used during code generation process, or by other tools to pass on information that is used to operate on elements in particular ways.

MDG Technologies

MDG Technologies may be used to create a logical collection of resources that may contain UML Profiles, Patterns, Code Templates, Image files and Tagged Value types that can be accessed from a single point in the Resource view.

EA Add-In

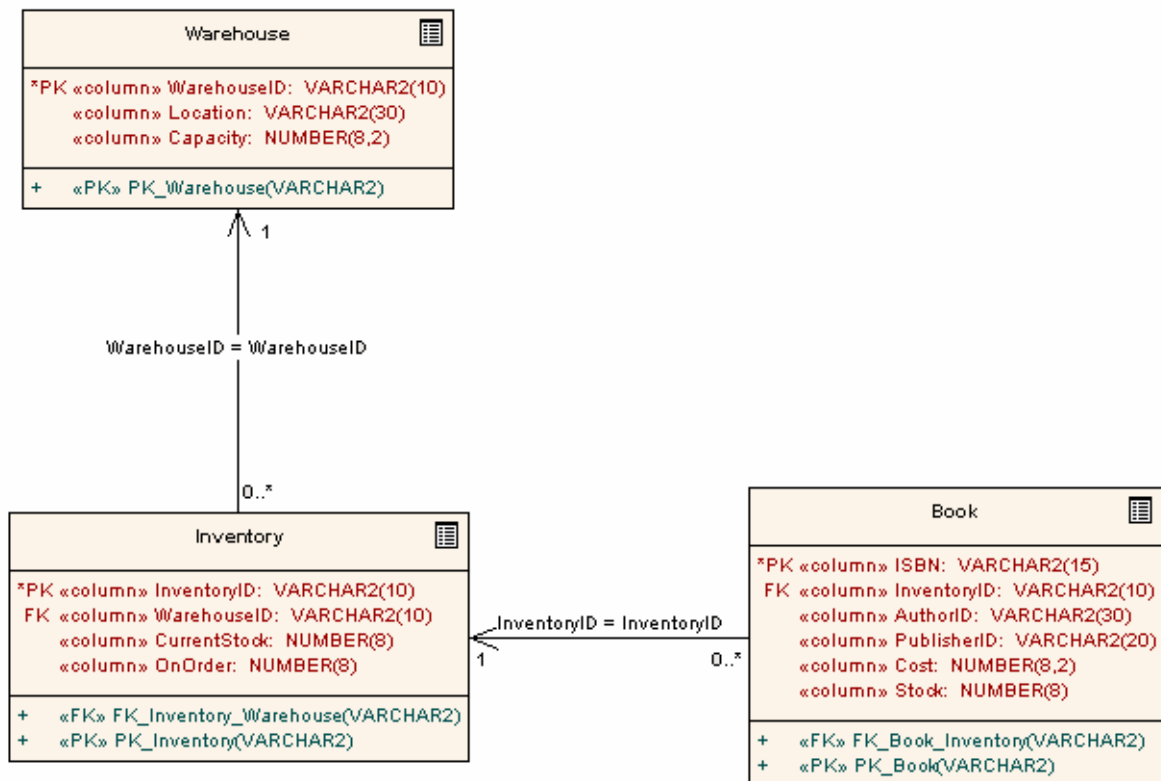
EA Add-ins allows users to build their own functionality into EA, creating their own mini programs which can extend the capabilities of EA, defining their own menus, and creating their own [Custom Views](#).

See Also

- [UML Profiles](#)
- [Patterns](#)
- [Code Templates](#)
- [Tagged Values](#)
- [MDG Technologies](#)
- [EA Add-ins](#)

2.9 Database Administrator

EA supports a range of features for the administration of databases, including the modeling of database structures, importing database structures from an existing database and the generation of DDL for the rapid creation of databases from a model.

**Create Logical Data Models**

With Enterprise Architect the Database Administrator can build database diagrams using the built-in UML Data Modeling Profile. This supports the definition of primary and foreign keys, cardinality, validation, triggers, constraints and indexes.

Generate Schema

By using Enterprise Architect's DDL generation function the Database Administrator can create a DDL script for creation of the database table structure from the model. EA currently supports JET-based databases, DB2, InterBase, MySQL, SQL SERVER, PostgreSQL, Sybase Adaptive Server Anywhere and Oracle 9i and 10g .

Reverse Engineer Database

Using an ODBC data connection the Database Administrator can import a database structure from an existing database to create a model of the database. Generating the model directly from the database allows the DBA to quickly document their work and create a diagrammatic account of a complex database through the graphical benefits of UML.

See Also

- [Database Schema](#)
- [Creating a Data Model Diagram](#)
- [Creating a Table](#)
- [Setting Properties of a Table](#)
- [Creating Columns](#)
- [Creating Primary Keys](#)
- [Creating Foreign Keys](#)
- [Creating Indexes and Triggers](#)
- [Generating DDL for a Table](#)
- [Generating DDL for a Package](#)
- [Converting Datatype for a Table](#)
- [Converting Datatype for a Package](#)
- [Customizing Datatypes for a DBMS](#)
- [Importing a Database Schema from an ODBC Data Source](#)

Part



3 Getting Started

The information in this section provides you with a quick start guide to Enterprise Architect. It illustrates how to open and create new projects, set up local preferences and navigate the Enterprise Architect application. When you have read through this section you should be able to begin modeling your own software projects with EA and UML.

In addition we recommend that you fully explore the sample project supplied with EA. It will assist you in learning to use EA, and offers tips on getting the most out of EA's features.

For additional help, check out the online [UML Tutorial](#) provided by Sparx Systems.

See Also

- [Installation](#)
- [Quick Start](#)
- [What is Enterprise Architect?](#)
- [What is UML?](#)
- [Starting the Application](#)
- [Licence Management](#)
- [Registering a Full Licence](#)
- [Upgrade an Existing Licence](#)
- [Finding Your Licence Information](#)

3.1 Installation

Enterprise Architect is distributed as a single executable setup file (.exe). The Corporate Edition requires additional files and supplementary installation processes if you plan to use the SQL Server, MySQL, PostgreSQL, Sybase Adaptive Server Anywhere or Oracle 9i and 10g options (see below). Please note that installation and maintenance of these DBMS systems is not covered under the support agreement.

The latest edition of the evaluation and registered versions of EA are always available from www.sparxsystems.com.au. Access to the registered version requires a username and password to gain access to the registered user area of the web site. These will be provided upon purchase of a license.

Installing EA

1. Run the EA setup program.
2. Generally you can accept all the default options without change.
3. If you wish to place EA in a directory other than the default, enter the name of the destination when prompted.
4. You may be prompted to restart your computer when the installation completes. Although this is not always necessary (if you already have the components EA requires installed on your computer), we recommend a restart just to be sure.

Corporate Edition users planning to use SQL Server, MySQL, PostgreSQL, Sybase Adaptive Server Anywhere or Oracle 9i and 10g as their model repository can access scripts that will create the required tables etc for the choice of DBMS. These can be found at one of the following pages:

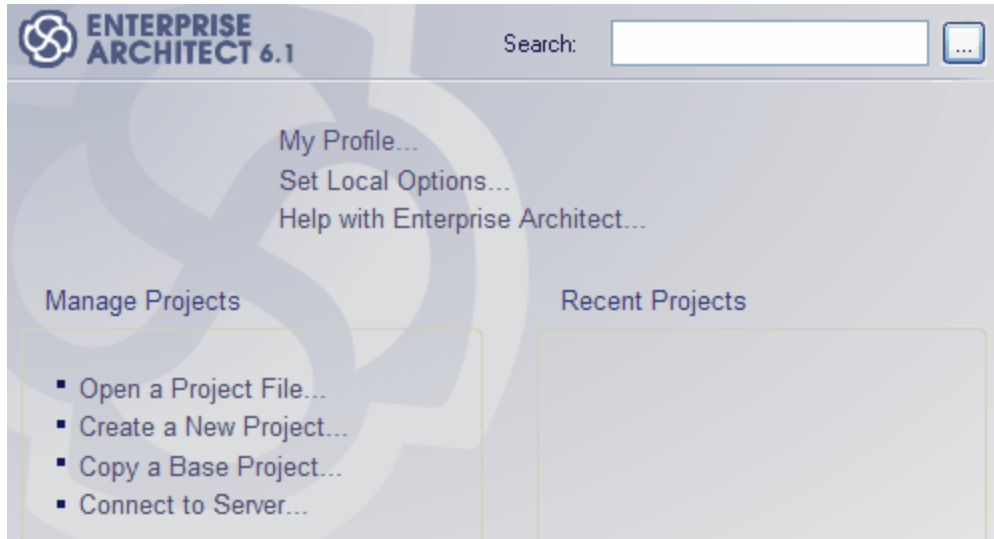
- The Corporate Edition Resources page at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
- The Trial Corporate Edition Resources page at www.sparxsystems.com.au/resources/corporate/

Note: EA requires Read/Write access to the Program files directory where EA has been installed.

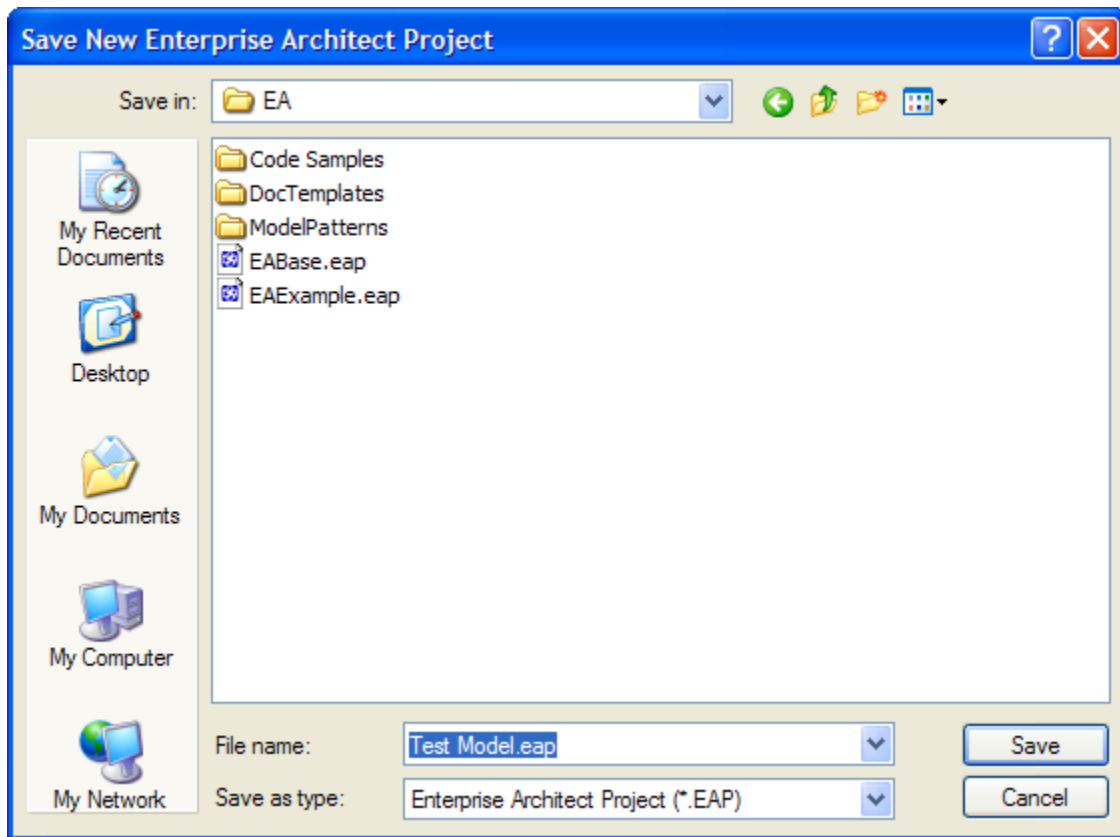
3.2 Quick Start

Welcome to Enterprise Architect! This quick start guide will help you to begin modeling with Enterprise Architect.

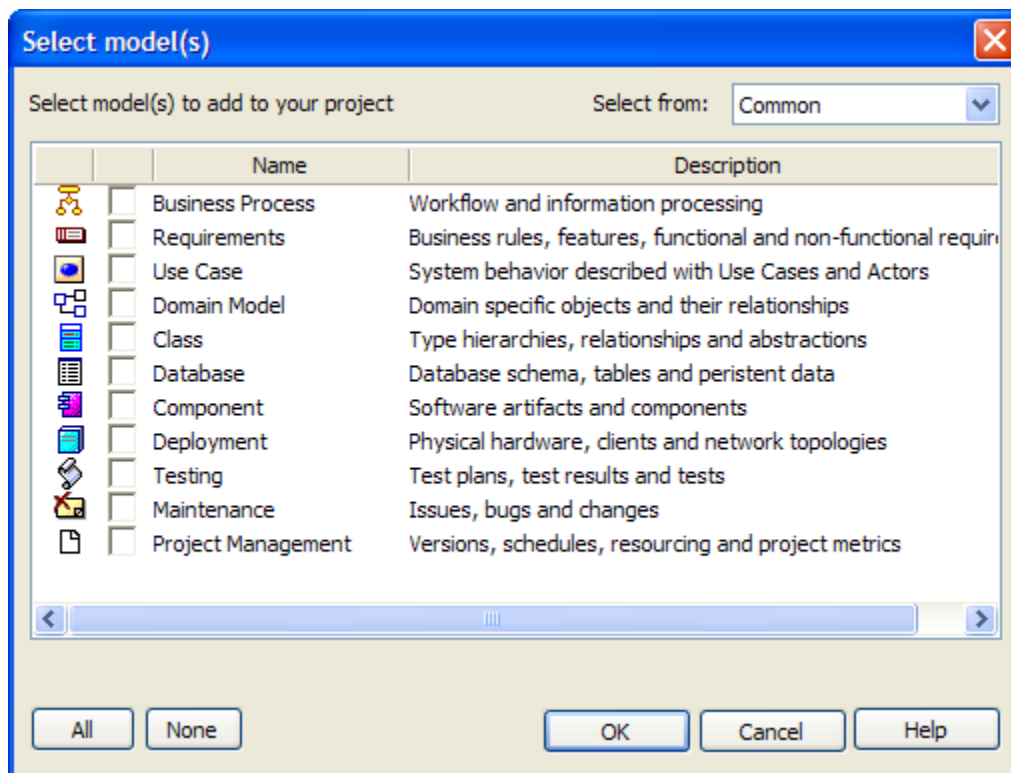
The Start Page



The [Start Page](#) will be where you begin our first [Project](#). To begin, click on *Create a New Project...*

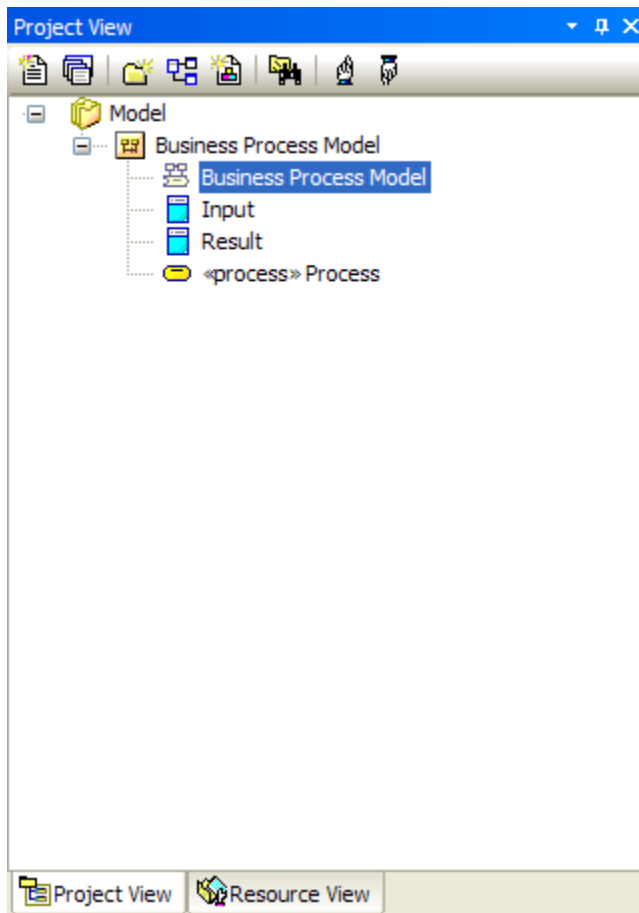


This will display the Save Enterprise Architect Project dialog. Give the Project a meaningful name, then click on the **Save** button to create the Project file.



This will display the [Model Wizard](#). From here, you can select one or more patterns that will provide you with the basics for your Project, as well as references to useful help files to get you started. Check the box of the model(s) that interest you, then click the **OK** button to create your Project.

Your new Project will be displayed in the [Project View](#) on the right-hand side of your screen.



To get started with your model pattern, expand the Package using the plus icon and double click the Diagram below.

For more information on using the [Model Patterns](#), use the contained help references, or use the links below.

See Also

- [Model Patterns](#)
- [Start Page](#)
- [Model Wizard](#)

3.3 Starting the Application

When you install Enterprise Architect on your computer, a new program folder called Enterprise Architect will be created in the Start menu (unless you changed the default name during installation).

Starting Enterprise Architect

You can start EA from the icon created on your Windows desktop during installation, or alternatively:

1. Open the Windows Start menu.
2. Locate the Enterprise Architect program folder.
3. Select Enterprise Architect.

In a few moments the [Start Page](#) will appear. From this dialog you may

- [Open a project file](#) (.EAP file)
- [Create a new project](#) (.EAP file)
- [Connect to a DBMS repository](#) (Corporate Edition only)

Note: By default, when you install EA, an empty 'starter' project called 'EABase.EAP' is installed, as well as an example project named 'EAExample.EAP'. We recommend that new users select the 'EAExample' file and explore it in some detail while they become familiar with UML and software engineering using Enterprise Architect.

See Also

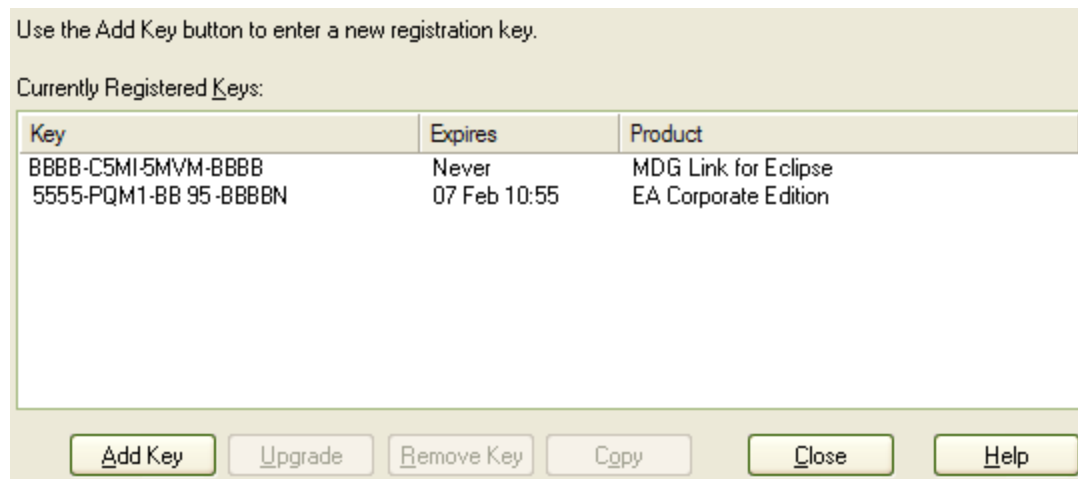
- [Connect to a MySQL Repository](#)
- [Connect to a SQL Server Repository](#)
- [Connect to an Oracle Repository](#)
- [Connect to PostgreSQL Repository](#)
- [Connect to an Adaptive Server Anywhere Repository](#)
- [Connect to a MSDE Server Repository](#)

3.4 Licence Management

Licence Management

The Licence Management dialog in Enterprise Architect allows the user to upgrade EA and to register Add-ins. To access Licence Management from within EA open the Help menu and select *Register and Manage Licence Key(s)*.

This will bring up the Licence Management dialog with the following options:



| Menu Item | Functionality |
|------------|---|
| Key | Displays the current key(s) registered with EA. |
| Add Key | The Add Key function allows the user to: <ul style="list-style-type: none"> • Add a new key, either to update to a higher version of Enterprise Architect or to register a new Add-in. • Obtain a key from the Enterprise Key Store (available for version 4.51 and above). For more information Regarding Adding Keys see the Add License Key topic. |
| Upgrade | Allows the EA Desktop and EA Professional editions to be upgraded. |
| Remove Key | Renders the Add-in or current version of EA inoperable. |
| Copy | Places the highlighted key into the clipboard. |
| Close | Closes the dialog. |
| Help | Opens up the help for this topic |

The following tasks may be run from the Licence Management dialog window:

- [Registering a Full Licence](#)
- [Upgrading an Existing Licence](#)
- [Registering an Add-in](#)

3.4.1 Add License Key

Two types of keys may be used in conjunction with Enterprise Architect; Private keys and Shared keys. Private keys allow the user to register an EA licence (Desktop, Professional or Corporate) or an Add-In key (MDG link for Eclipse and MDG Link for Visual Studio.NET) to the machine that they are currently using. Shared keys allow the user to obtain a product key from key store. Shared Keys are only available with a floating license using the Sparx Enterprise Key Store and require EA version 4.51 or higher.

Add a Private Key

To add a private key ensure that the *Add Registration Key* dialog is set to the *Enter Private Key* tab. Enter the *User Name* and *Company* into the relevant text fields and then copy a registration key into the licence key field. Press the *OK* button to confirm the key selection.

Enter Private Key Get Shared Key

Name: Sparx

Company: Sparx Systems

Copy registration key into space below, then press OK button

{F2222220-022D-11d5-A76F-DODO-DODO-SVDO-DODO-YDO-A0}

OK Cancel Help

Add a Shared Key

To add a shared key ensure that the *Add Registration Key* dialog is set to the *Get Shared Key* tab. Enter the *User Name* and *Company* into the relevant text fields and then use the *...* button to browse to the location of the Shared Key Store. Select the appropriate product and then press the *OK* button.

The screenshot shows a dialog box titled "Get Shared Key" with the following fields and controls:

- Name:** Sparx
- Company:** Sparx Systems
- Shared key store:** Q:\Stored_Keys (with a browse button "...")
- Select a Product:** EA Corporate Edition (in a list box)
- Buttons:** OK, Cancel, Help

Note: Shared Keys are only available with a floating license using the Sparx Enterprise Key Store. Shared Keys require EA version 4.51 or higher. Only the Key Administrator needs to install the Key Store application, users simply connect to the configured key file using EA as described above.

More Information

- [EA Corporate Floating License \(Online Webpage\)](#)
- [Keystore Troubleshooting](#)

3.4.2 Keystore Troubleshooting

Common Shared Key Issues

Error reading Key Store file:
Access to [filename] was denied.

All users who are to use the shared key facility require read+write access to the sskeys.dat file containing the shared keys. Please verify all required users have sufficient permissions to the file and try again. If the problem continues, contact Sparx Support.

Error reading Key Store file:
Key File has been moved.

As a security measure in the key store, the hard drive serial number is recorded upon creation of the file. The file then cannot be moved from the original location it was created. If the key store needs to be re-located for any reason, the administrator should re-create the key store in the new location using the original licence keys.

This message may also appear when configuring a key store on a Novell-based file system. When creating the keystore, the administrator should be prompted if this is on a Novell-based file system. By answering yes to this, the key store will instead record the logical path used to create it, and all users must connect to the key store using this same path.

3.5 Registering a Full License

The trial version of Enterprise Architect available for download is an evaluation version only. For the full version you will first need to purchase one or more licenses. The license code(s) supplied will determine which edition (Desktop, Professional or Corporate) is activated on installation.

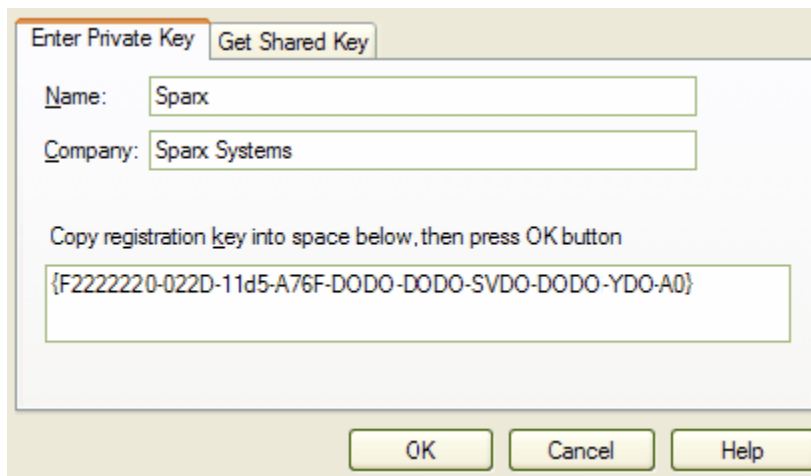
Registering EA

Follow these steps to obtain the full version and complete the registration process:

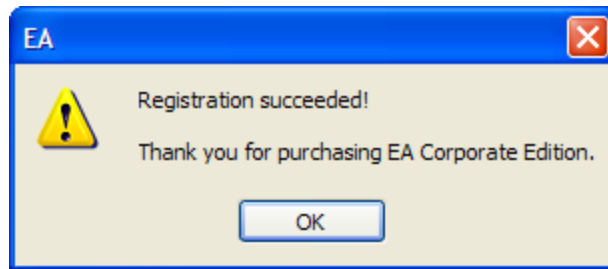
1. Purchase one or more licenses.
2. Once you have paid for a licensed version of Enterprise Architect, you will receive (via email or other suitable means)
 - a license key(s)
 - the address of a web site from which to download the full version
3. Save the license key and download the latest full install package from the address supplied.
4. Run the setup program to install the full version.
5. Open Enterprise Architect from the Start Menu or Desktop icon.
6. When the Licence Management dialog appears click on the **Add Key** button.



7. The Enter Registration dialog will then prompt the user to enter a license key - **including the { and } characters** (use copy and paste from an email to avoid typing mistakes).



8. The full version is now activated on your PC.

**See Also**

- [Upgrading an Existing License](#)

3.6 Upgrade an Existing License

Enterprise Architect comes in several editions - Desktop, Professional and Corporate. If you are using the Desktop or Professional edition, you may wish to upgrade your license at a future date. You can do this by purchasing an upgrade key from Sparx Systems (see the [Sparx Systems](#) website for purchase details).

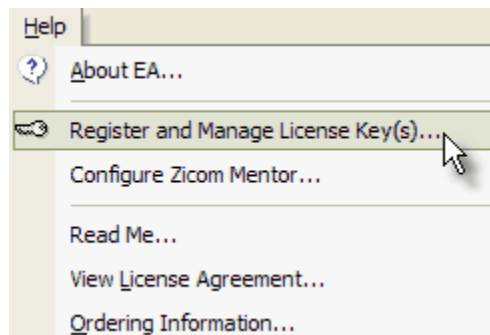
An upgrade key is a special key that will upgrade an existing license to a higher edition. Once you have purchased and received the appropriate key, follow the instructions below to unlock additional features.

Note: *The Lite version, the Trial version and the Corporate version cannot be upgraded. If you have purchased EA, you will need to download the registered version from www.sparxsystems.com.au/securedownloads/easetupfull.exe before you can enter your registration key.*

Upgrading EA

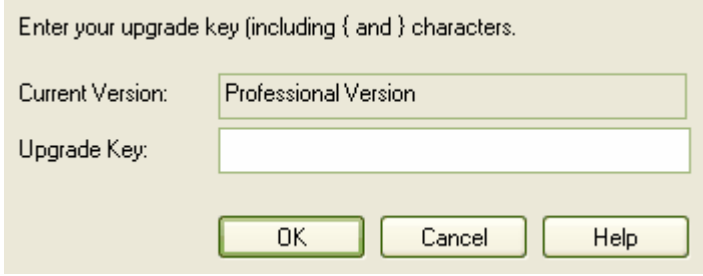
Follow these steps to upgrade from one edition to another:

1. Make sure you have a valid upgrade key purchased from Sparx Systems - you will typically receive this in an email or PDF format.
2. Open Enterprise Architect.
3. From the *Help* menu, select the *Register and Manage Licence Key(s)* option.



4. This will bring up the Licence Management Dialog, press the *Add Key* or the *Upgrade* button to enter a new licence key.

5. If the Add key option was taken the user will be presented with the *Enter Registration Key* dialog (enter the key you received for the upgraded edition of EA - **including the { and } characters** (use copy and paste from an email to avoid typing mistakes).
6. If the Upgrade option was taken the user will be presented with the Upgrade Key dialog (enter the key you received for the upgraded edition of EA - **including the { and } characters** (use copy and paste from an email to avoid typing mistakes).



Enter your upgrade key (including { and } characters).

Current Version: Professional Version

Upgrade Key:

OK Cancel Help

7. Press **OK**. If the key is valid, EA will modify the *Current Version* to reflect the upgrade.
8. Close EA and restart to enable the unlocked features.

Tip: Once you have successfully completed the upgrade, go to *Help | About EA*. Copy the registration key shown and store it somewhere safe - this is a key to the full license of the edition you have upgraded to. If you ever need to reinstall EA, you can register it with this key, so you won't need to go through the upgrade process again.

3.7 Finding Your License Information

You can find information about your EA license in the *About Enterprise Architect* dialog, located under the *Help | About EA* menu item.



Part

4

4 Using Enterprise Architect

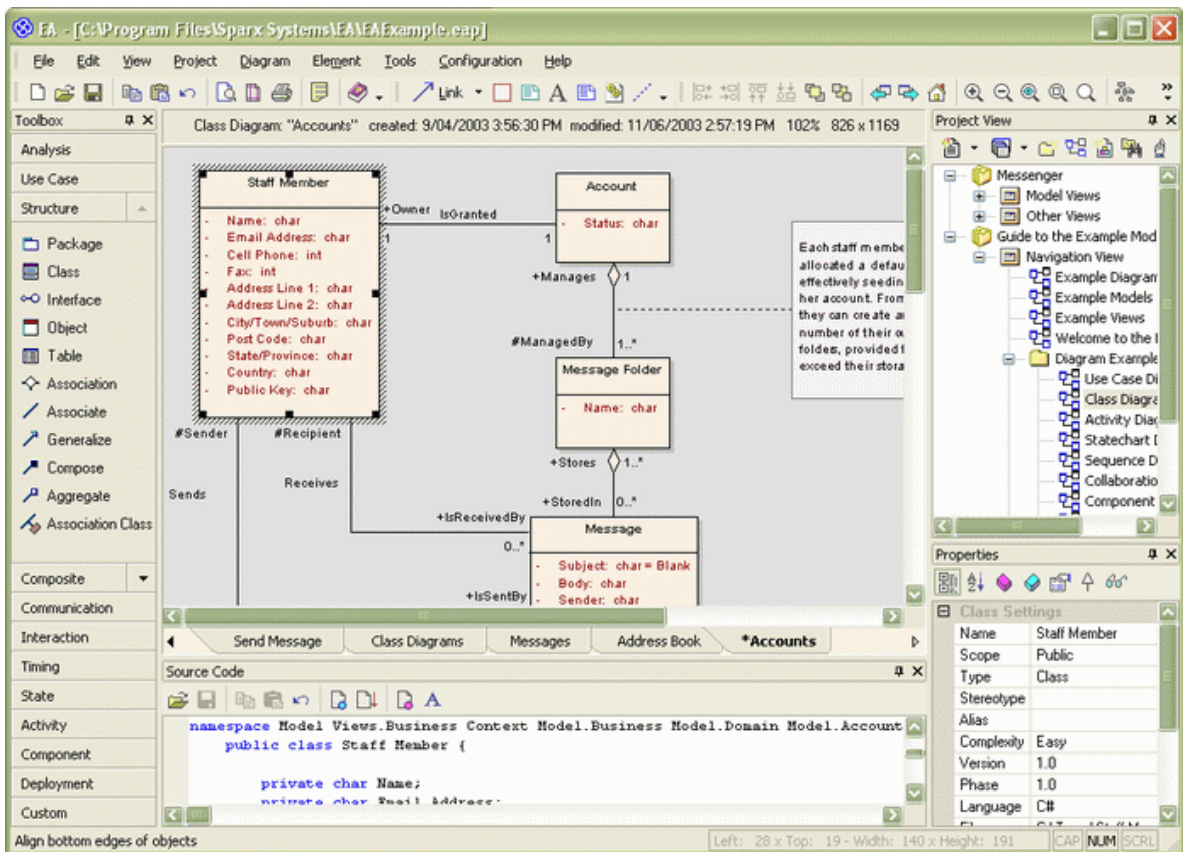
This topic provides a detailed exploration of the Enterprise Architect tools and features.

See Also

- [The Application Workspace](#)
- [The Start Page](#)
- [Model Patterns](#)
- [Arranging Windows and Menus](#)
- [The Main Menu](#)
- [View Options](#)
- [Searching a Project](#)
- [Workspace Toolbars](#)
- [The Project View Browser](#)
- [Dockable Windows](#)
- [The Quick Linker](#)
- [The UML Toolbox](#)
- [Package Tasks](#)
- [Diagram Tasks](#)
- [Element Tasks](#)
- [Element Inplace Editing Options](#)
- [Defaults and User Settings](#)
- [Register Add-In](#)
- [Keyboard Shortcuts](#)

4.1 The Application Workspace

The *Application Workspace* is comprised of several windows. In concept it is similar to programs like Microsoft Outlook and the Microsoft Visual Studio application suite.



Understanding the Application Workspace

The EA application workspace is comprised of the following components:

The Project Browser

The [Project Browser](#) is used to navigate your project. Double click on packages to open them and right click with the mouse to view context sensitive menu options for project elements.

The Property Browser

The [Property Browser](#) is a set of tabbed windows for notes, element properties, requirements and other aspects of your project.

The Diagram View

The large central area is the [Diagram View](#). This is where you can place new model elements and set their characteristics. Note that when you first open EA there is no active diagram - before you can add elements you must create and open a diagram.

The Main Menu and Toolbars

At the top of the workspace are the [Main Menu](#) and [Toolbars](#).

The UML Toolbox

The [UML Toolbox](#) is an Outlook style toolbar from which you can select model elements and relations to create.

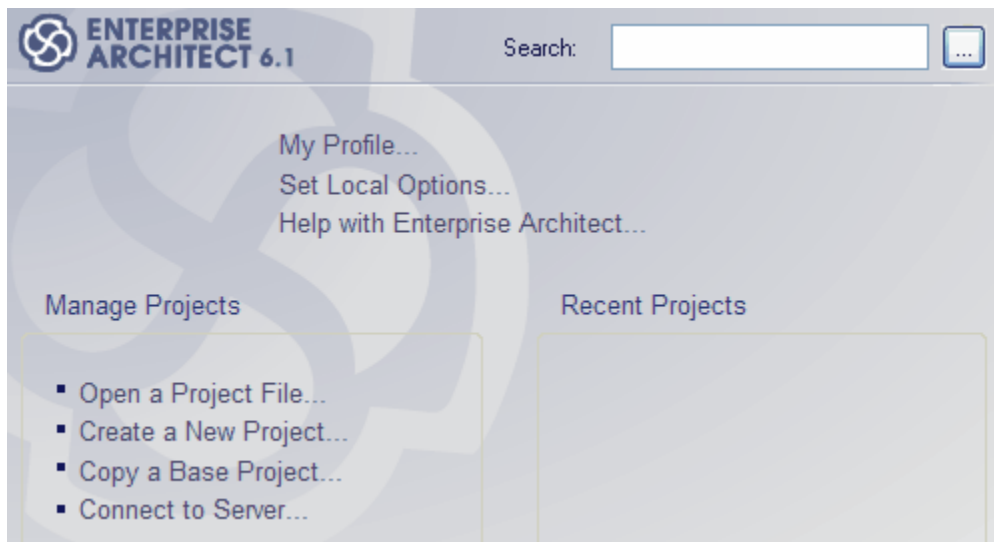
Together these elements provide a simple and flexible software engineering environment. Those who have used MS Office and Visual Studio should find the Enterprise Architect interface quite familiar.

4.2 The Start Page

When you start Enterprise Architect, the first page displayed is the *Start Page* (below). This is a convenient jumping off point for some common activities, such as:

- [Open a Project File](#) (.EAP file)
- [Create a New Project](#) (.EAP file)
- [Copy a Base Project](#) (.EAP file)
- [Connect to Server](#) (Corporate edition only)

Click on the hyperlink text to activate any of these options. The Recent Models section contains a list of models (both .EAP files and DBMS connections) recently used. Click once on a model to open it in EA.



| Element | Description |
|----------------------|--|
| Open a Project File | Displays the Open Project dialog. |
| Create a New Project | Save a new project and open the Model Wizard dialog. |
| Copy a Base Project | Select a different Base Project to generate a new model from. |
| Connect to Server | Allows you to specify a Data Source name to connect to. MySQL , SQL Server , Oracle 9i and 10g , PostgreSQL , MSDE , Adaptive Server Anywhere and Progress OpenEdge repositories are supported. Note: This feature is available in the Corporate edition only. |
| Recent Projects | Contains a list of the most recently used EA projects. Click on the project you wish to open. |

If your model has a [default diagram](#) set, the default diagram will open immediately over the top of the Start Page. You will still be able to access the Start Page from the [diagram tabs](#) below the diagram.

See Also

- [The Start Page Options](#)
- [Removing Recent Projects](#)
- [Edit My Profile](#)

4.2.1 The Start Page Options

The Start Page gives you the following options which can be accessed using the hyperlinks shown:

| Option | Description |
|--------------------------------------|--|
| My Profile | Opens the <i>Edit My Profile</i> dialog where you can: <ul style="list-style-type: none"> • enter your name • narrow the tools shown in the UML toolbox to a particular subset if you wish • preferred visual layout • access the Custom Tools window • access the Keyboard Accelerator Map |
| Set Local Options | Opens the Local Options dialog . |
| Help with Enterprise Architect | Opens the help files. |
| Open a Project File | Opens a browse window to let you select a model file to open. |
| Create a New Project | Opens the <i>Create New Enterprise Architect Project</i> dialog. |
| Copy a Base Project | Select a different Base Project to generate a new model from. |
| Connect to Server | Allows you to specify a Data Source name to connect to. MySQL , PostgreSQL , Sybase Adaptive Server Anywhere , SQL Server and Oracle 9i and 10g repositories are supported. <i>Note: This feature is available in the Corporate edition only.</i> |
| Recent Projects | A list of recently accessed models - click once to open. see Removing Recent Projects |

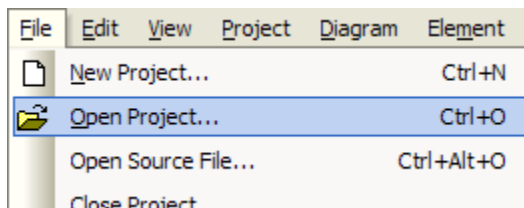
See Also

- [The Start Page](#)
- [Removing Recent Projects](#)
- [Edit My Profile](#)

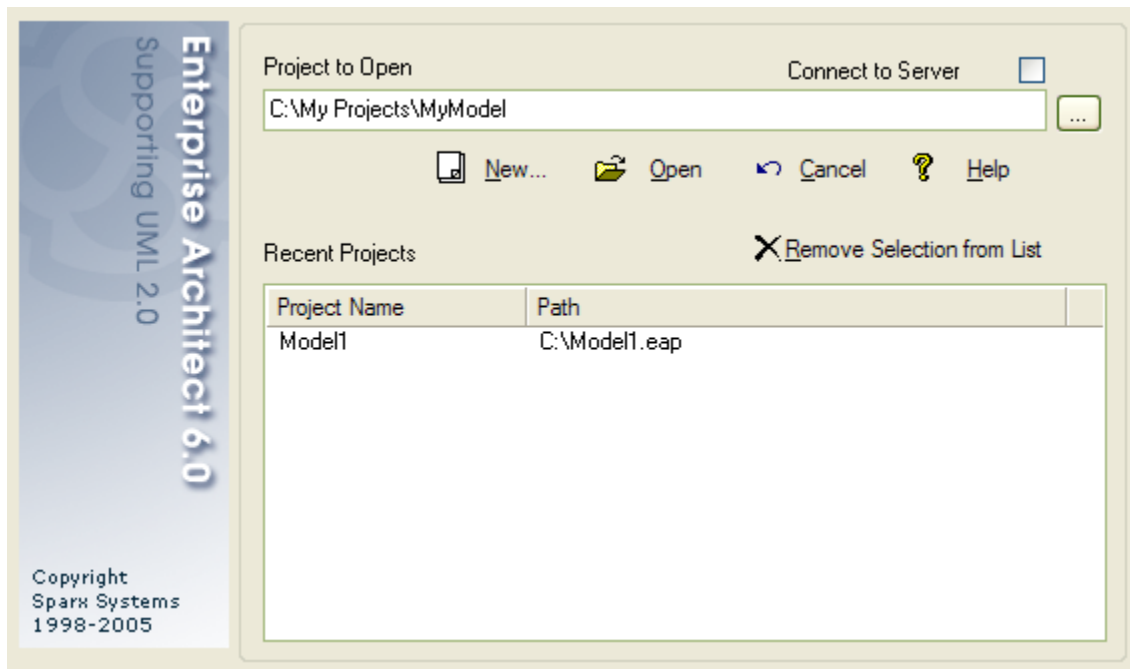
4.2.2 Removing Recent Projects

To remove a project link from the *Recent Projects list* on the Start Page, use the following instructions:

1. Select *File | Open Project* from the main or press [Ctrl+O].



2. In the Recent Projects text field highlight the project that you wish to remove.
3. Press the *Remove Selection from List* button.



Note: Removing the link to a Model from the Start Page only removes the link to the model and does not remove the .EAP file from the file system.

See Also

- [The Start Page](#)
- [The Start Page Options](#)
- [Edit My Profile](#)

4.2.3 Edit My Profile

The *My Profile* link on the start page opens the *Edit My Profile* dialog.

My Name:

UML Toolbox

By selecting a Perspective or using the Custom Configuration option, you can limit the UML Toolbox folders and the items contained in those folders to a smaller and more focused set. Use the Show All option to restore all folders and icons

Perspective:

Visual Style and Layout

XP Style
 2003 Style
 Classic Style
 2005 Style

User Interface

| Option | Description |
|-------------------------|---|
| My Name | Record your name |
| Configure UML Toolbox | You can narrow the tools shown in the UML toolbox to a particular subset - <ul style="list-style-type: none"> • By Role - Select a role from the dropdown list • Custom Configuration - Customize the UML toolbox in the Local Options window |
| Visual Style and Layout | Set your preferred visual layout - <ul style="list-style-type: none"> • Set the XP, 2003, Classic or 2005 visual style • Set the layout to default • Select a Custom Layout |
| Customize Interface | Access the Custom Tools window - you can also access this through Tools Options on the main menu |
| Show Keyboard Map | Access the Keyboard Accelerator Map - you can also access this through Help Keyboard Accelerator Map on the main menu |

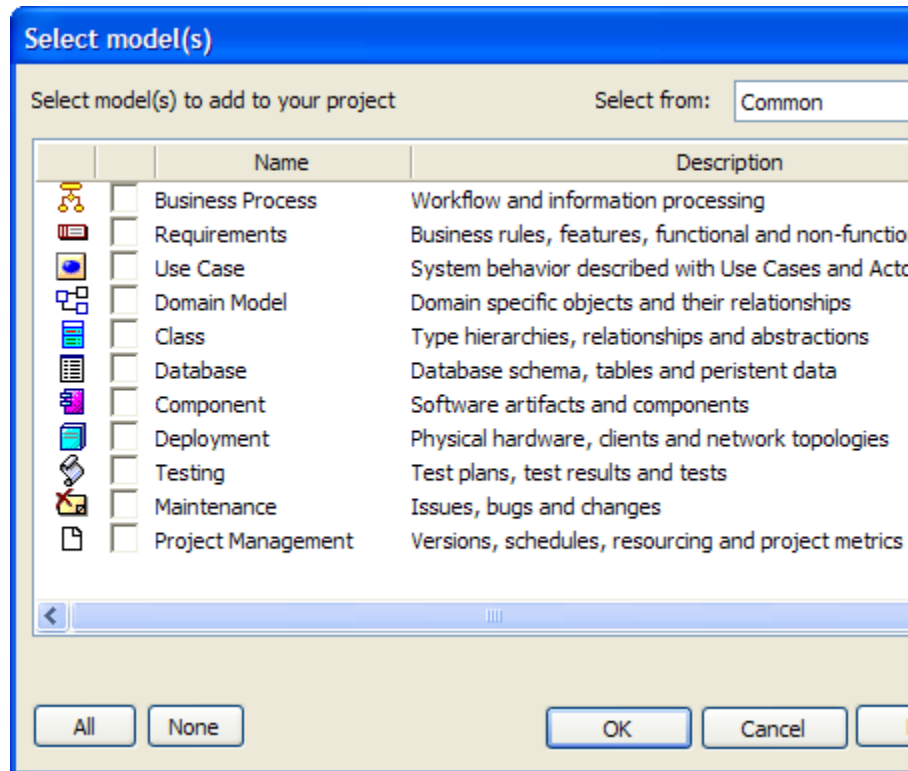
See Also

- [The Start Page](#)
- [The Start Page Options](#)
- [Removing Recent Projects](#)

4.3 Model Patterns

The Model Patterns contained in Enterprise Architect are designed to assist in the creation of Projects and Models for both new and experienced users.

They each provide a framework upon which you can create your model. For more information on each particular pattern, see below.



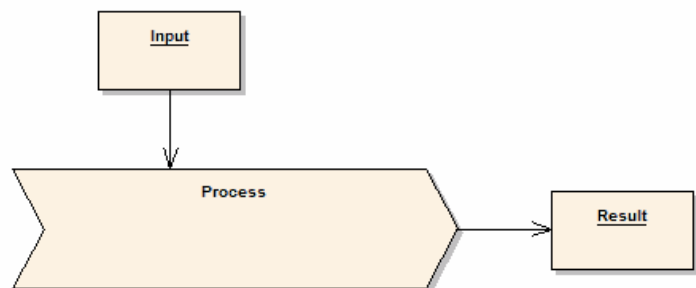
Pattern Format

The Business Process Model describes the both the behavior and the information flows within an organization or system.

As a model of business activity, it captures the significant events, inputs, resources, processing and outputs associated with relevant business processes.

[Read about Business Process Modeling](#)

[View Example](#)



All the Model Patterns provided with Enterprise Architect follow the above format.

Note

The note will introduce you to the Model Pattern and outline its purpose.

Help Links

These links will provide further information on how to use the model. Depending on the Model Pattern, examples and other useful information will also be linked to here.

Pattern

The pattern section in the Model Pattern should give you a framework for creating your own model.

The sections listed below provide an introduction to the terminology and icons used in the Model Patterns. It will give you a quick guide to the UML concepts important to the Patterns, and how they are applied in

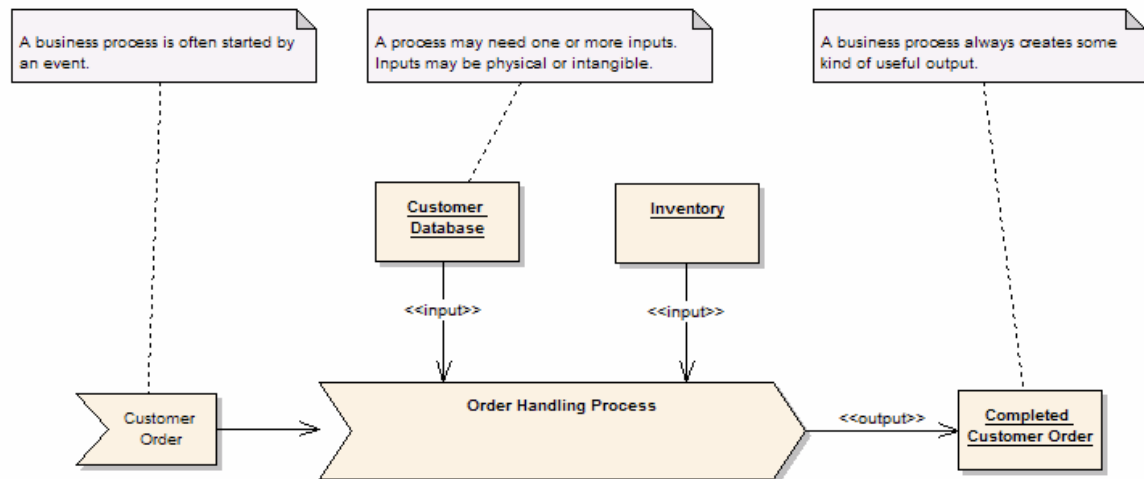
Enterprise Architect.

See Also

- [Business Process Model Pattern](#)
- [Requirements Model Pattern](#)
- [Use Case Model Pattern](#)
- [Domain Model Pattern](#)
- [Class Model Pattern](#)
- [Database Model Pattern](#)
- [Component Model Pattern](#)
- [Deployment Model Pattern](#)
- [Testing Model Pattern](#)

4.3.1 Business Process Model Pattern

The Business Process Model describes the both the behavior and the information flows within an organization or system. As a model of business activity, it captures the significant events, inputs, resources, processing and outputs associated with relevant business processes.



Online Resources

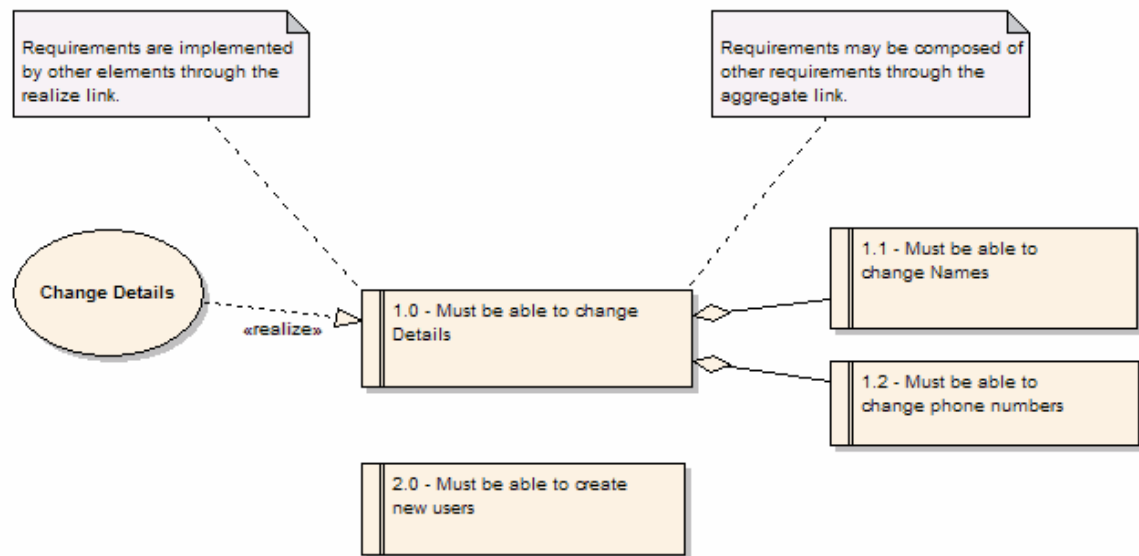
- [The Business Process Model](#)

See Also

- [Business Modeling](#)
- [Analysis Diagram](#)

4.3.2 Requirements Model Pattern

The Requirements Model is a structured catalogue of end-user requirements and the relationships between them. The [Requirements Management](#) built into Enterprise Architect can be used to define requirement elements, link requirements to model elements, link requirements together into a hierarchy and report on requirements.

**Online Resources**

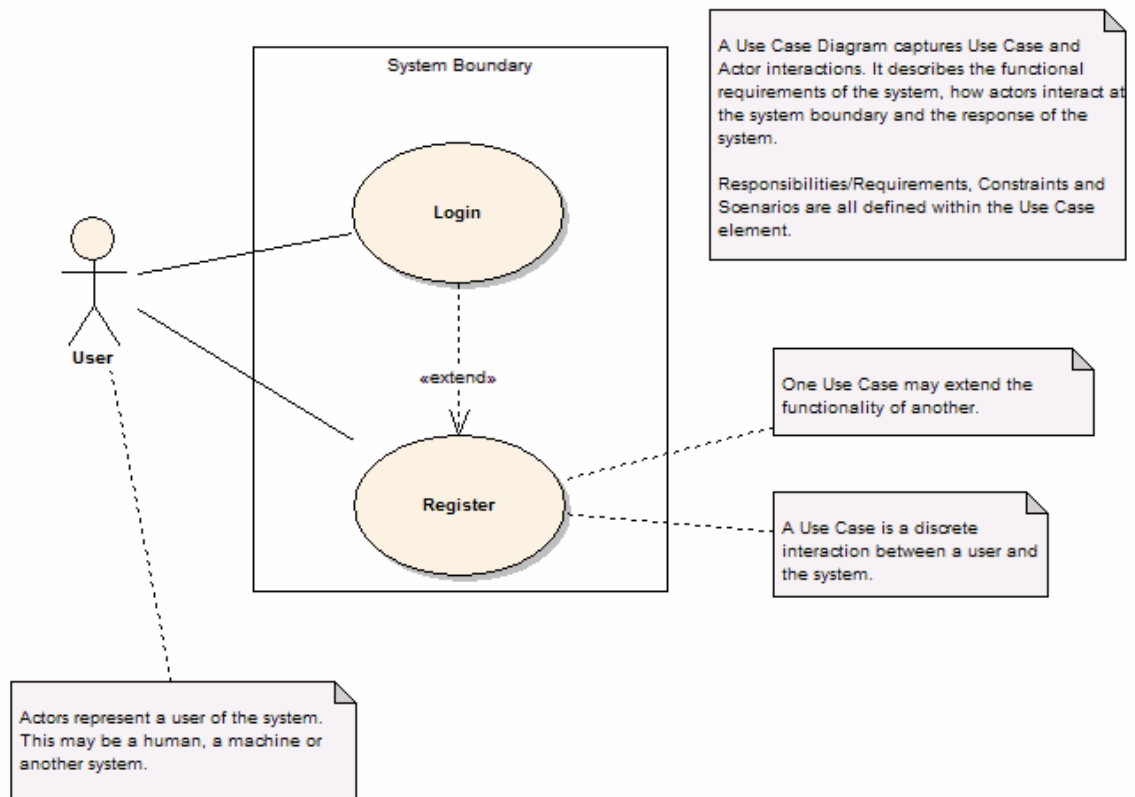
- [Requirements Management in Enterprise Architect](#)

See Also

- [Requirements Management](#)
- [Package Tasks](#)

4.3.3 Use Case Model Pattern

The Use Case Model describes a system's functionality in terms of Use Cases. Each Use Case represents a single repeatable interaction that a user or "actor" experiences when using the system, emphasizing the users perspective of the system and interactions. See [Use Case](#) for more information.



Online Resources

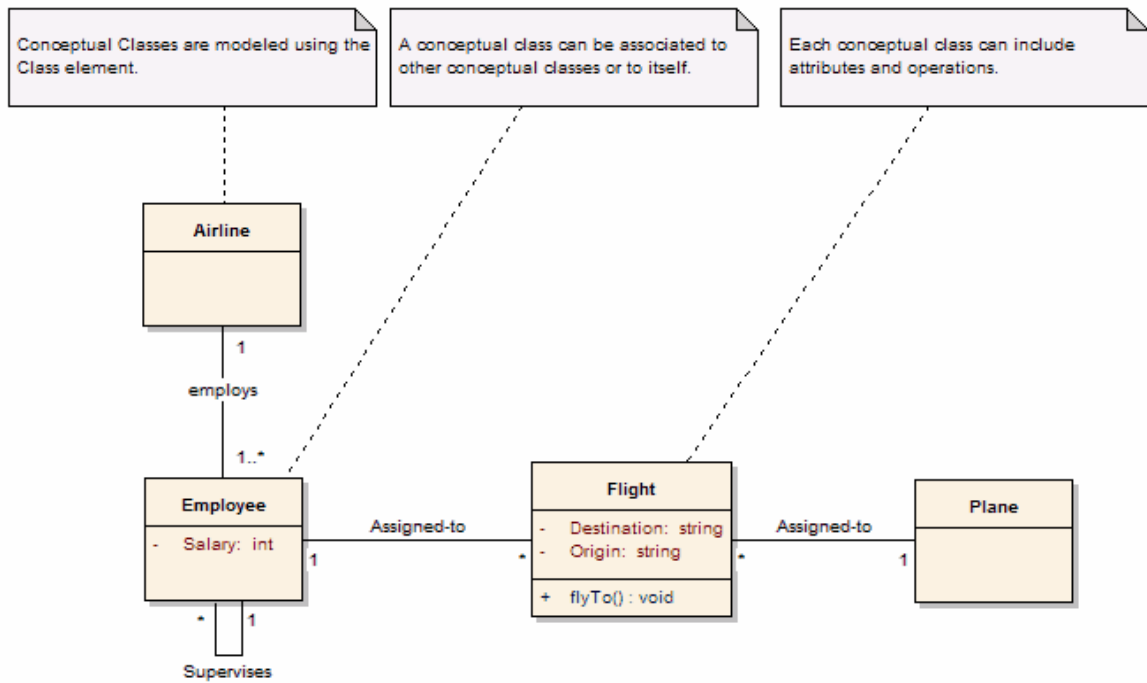
- [The Use Case Model](#)

See Also

- [Use Case](#)
- [Use Case Toolbox](#)
- [Use Case Diagram](#)

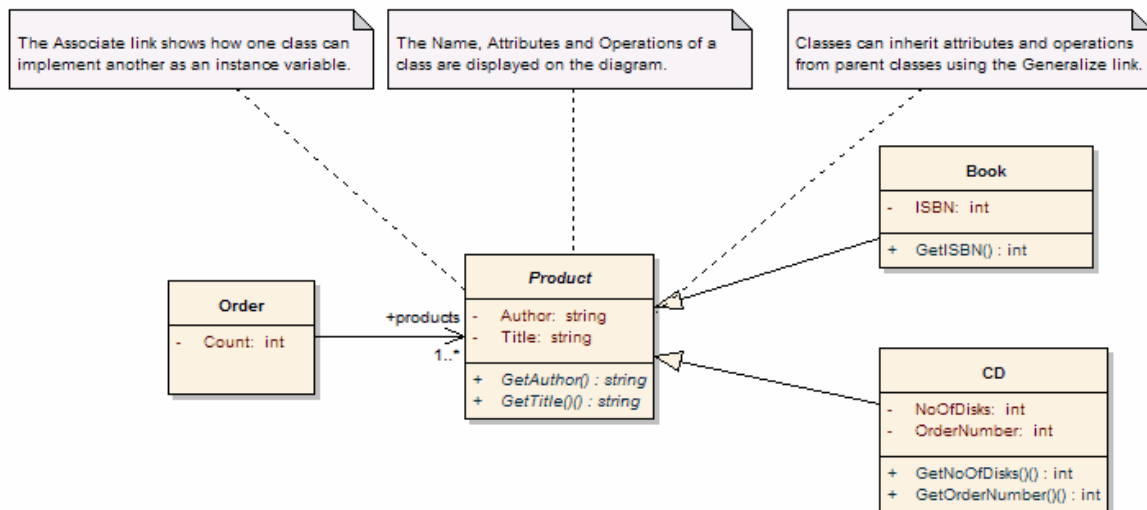
4.3.4 Domain Model Pattern

A Domain Model is a high-level conceptual model, containing physical and abstract objects, in an area of interest to the Project. It can be used to document relationships between and responsibilities of conceptual classes. It is also useful for defining the terms of a domain.



4.3.5 Class Model Pattern

The Class Model is a rigorous, logical model of the software system under construction. Classes generally have a direct relationship to source code or other software artifacts that can be grouped together into executable components.



Online Resources

- [The Logical Model](#)

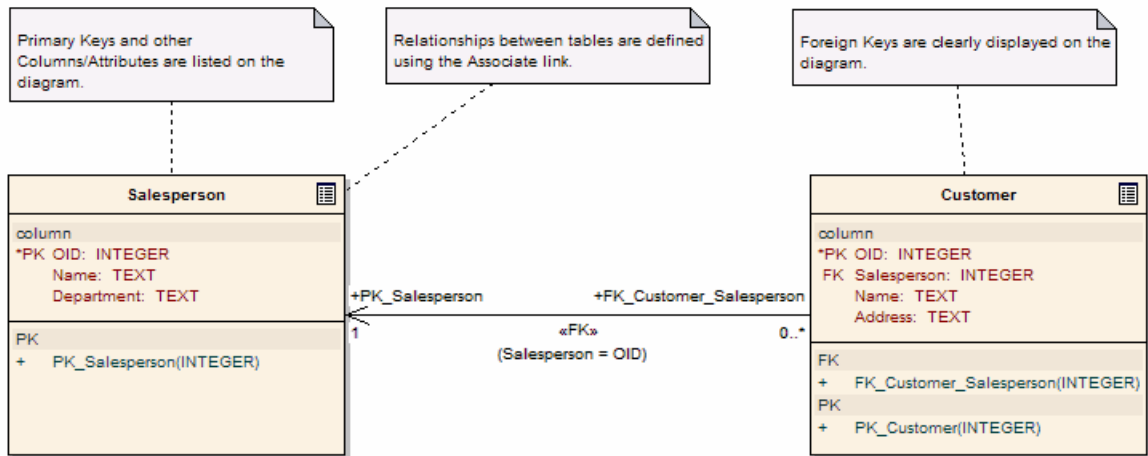
See Also

- [Class](#)
- [Class Diagram](#)

- [Class Toolbox](#)

4.3.6 Database Model Pattern

The Database Model describes the data which must be stored and retrieved as part of the overall system design. Typically this will mean relational database models which describe the tables and data in detail and allow generation of DDL scripts to create and setup databases.



Online Resources

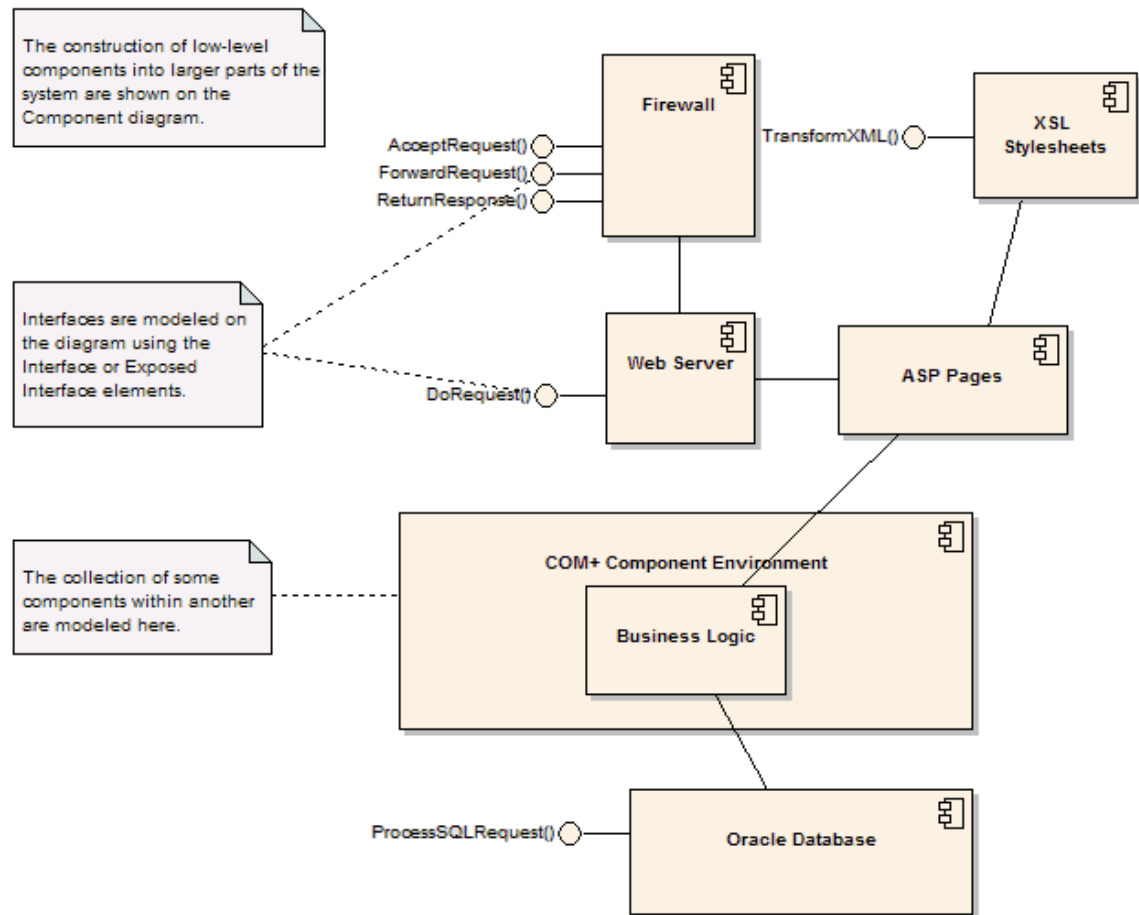
- [UML Database Modeling](#)

See Also

- [Database Modeling](#)

4.3.7 Component Model Pattern

The Component Model defines how classes, artifacts and other low level elements are collected into high level components, and the interfaces and connections between them. Components are compiled software artifacts that work together to provide the required behaviour within the operating constraints defined in the requirements model.



Online Resources

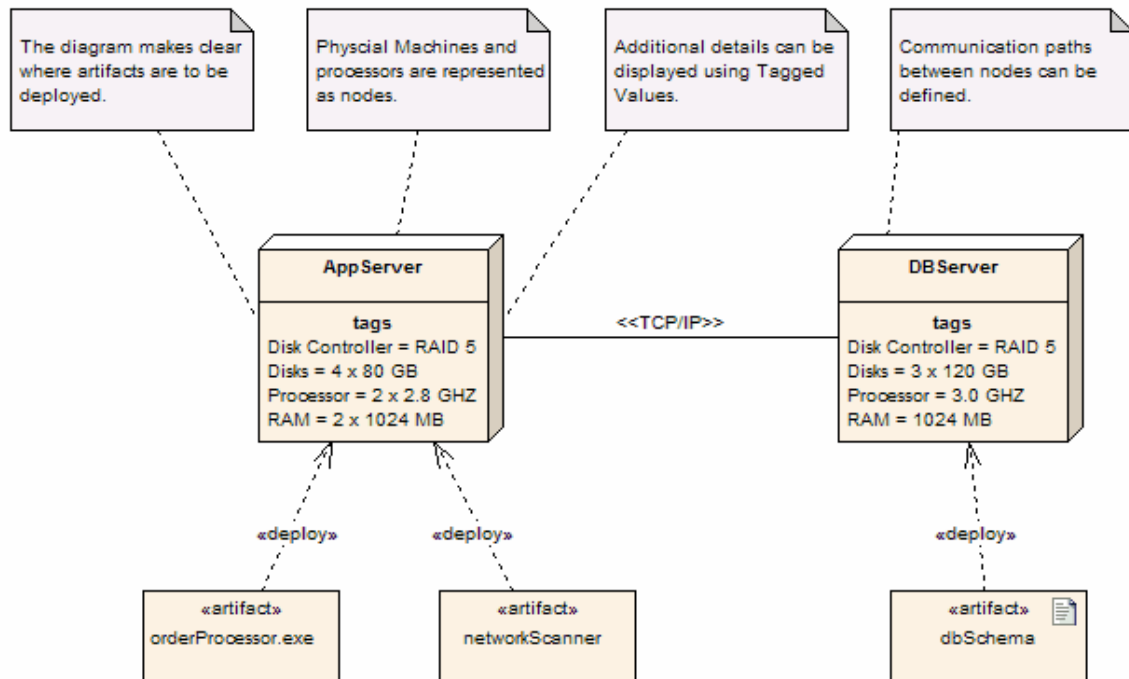
- [The Component Model](#)

See Also

- [Component](#)
- [Component Diagram](#)
- [Component Toolbox](#)

4.3.8 Deployment Model Pattern

The Deployment Model describes how and where a system will be deployed. Physical machines and processors are represented by nodes, and the internal construction can be depicted by embedding nodes or artifacts. As artifacts are allocated to nodes to model the system's deployment, the allocation is guided by the use of deployment specifications.



Online Resources

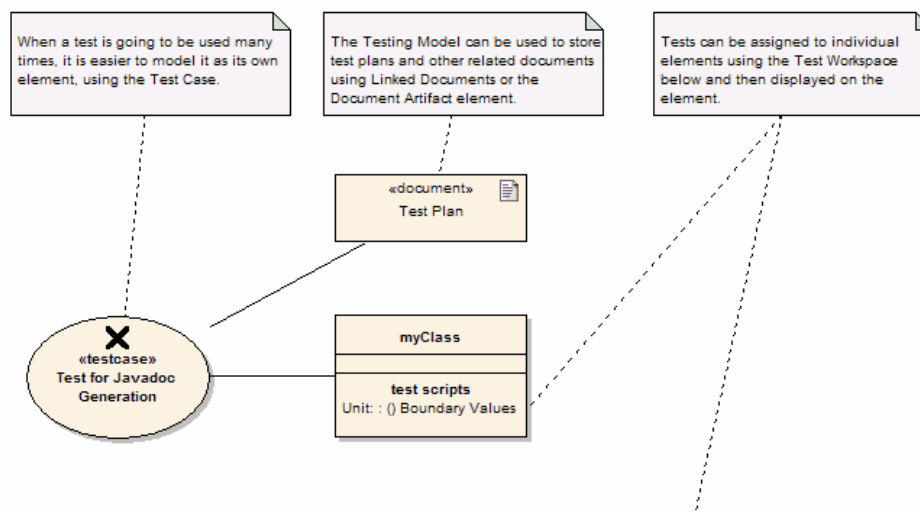
- [The Physical Model](#)

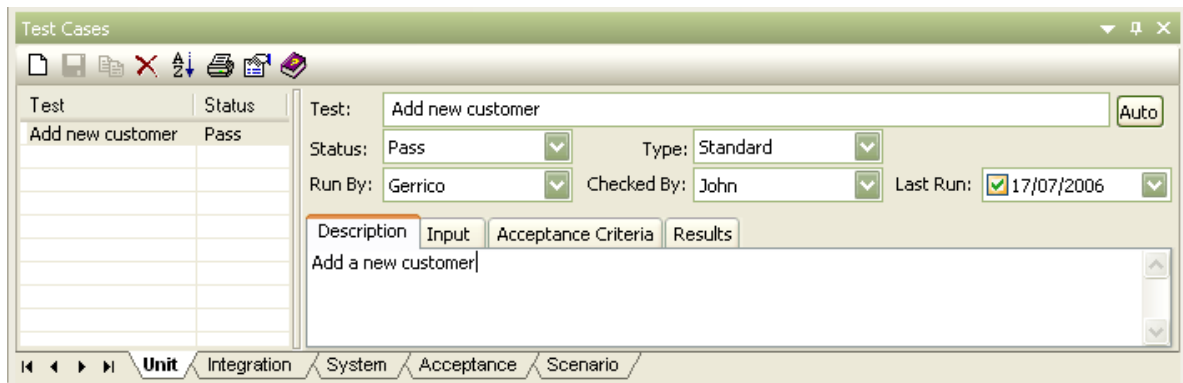
See Also

- [Deployment and Rollout](#)
- [Deployment Diagram](#)
- [Deployment Toolbox](#)
- [Displaying Tagged Values](#)

4.3.9 Testing Model Pattern

The Test Model describes and maintains a catalogue of tests, test plans and results that are executed against the current model.





Online Resources

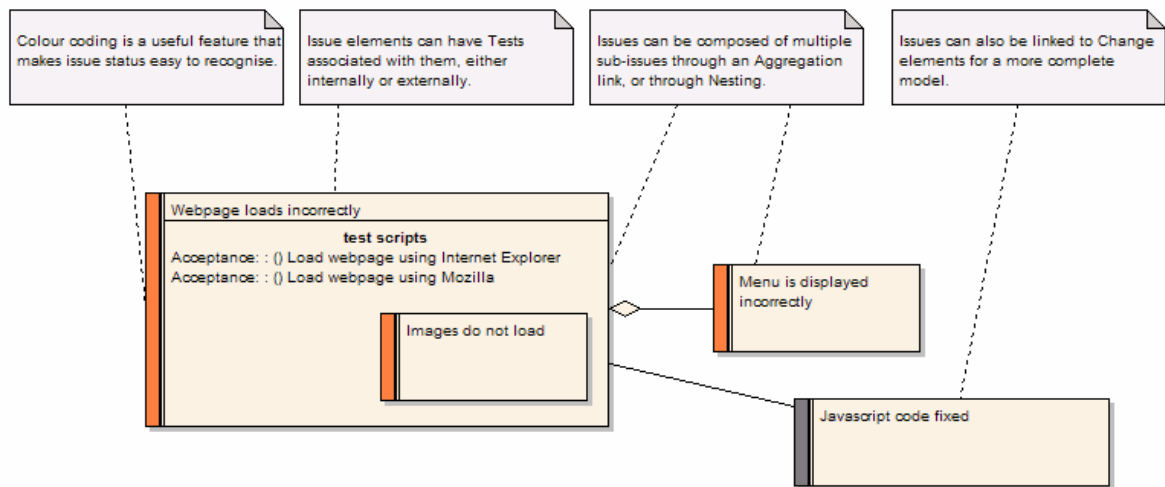
- [Testing Support in Enterprise Architect](#)

See Also

- [Testing](#)
- [Show Test Scripts](#)

4.3.10 Maintenance Model Pattern

The Maintenance Model allows visual representation of issues arising during and after development of a software product.. The Model can be enhanced with the integration of change elements and testing.



See Also

- [Maintenance](#)
- [Color Coding](#)

4.3.11 Project Model Pattern

The Project Model details the overall project plan, phases, milestones and resourcing requirements for the current project. Project managers may use Enterprise Architect to assign resources to elements, measure risk and effort and to estimate project size. Change control and maintenance are also supported (see [Maintenance Support](#)).

Online Resources

- [Project Manager](#)

See Also

- [Project Management](#)

4.4 Arranging Windows and Menus

Enterprise Architect allows you to rearrange the windows and some menus to suit your work habits.

See Also

- [Dockable Windows](#)
- [Dockable Windows, 2005 Style](#)
- [Autohide Windows](#)
- [Tear Off Menus](#)

4.4.1 Dockable Windows

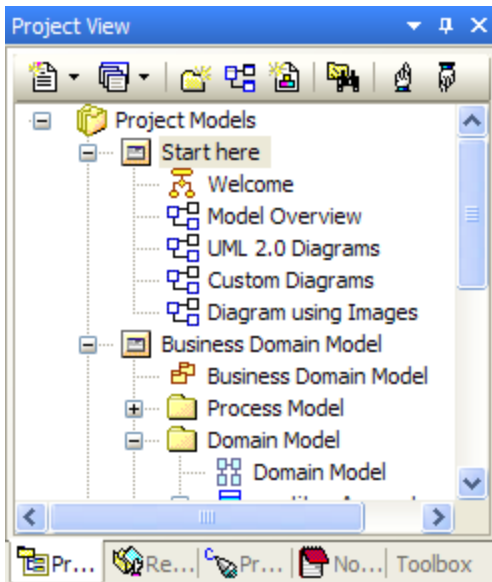
The *Project Browser* and *Property Browser* may be docked against any edge of the application workspace or freely floated. Drag the project or property browsers around the application workspace till you find a comfortable way of working. The examples below show two ways you can rearrange the windows to suit your work habits.

Floating Windows

Try floating the various windows instead of docking them. To do this, just drag the window by its title bar to the desired position.

Dock Required Windows into One Frame

You can also dock all of the windows you are using into a single frame. The following example shows the *Project Browser*, *Resource View*, *Properties*, *Notes* and *UML Toolbox* all in the one frame. This can be done with all dockable windows.



Note: The 2005 Style uses a [navigation compass](#) for docking.

4.4.2 Dockable Window 2005 Style

The 2005 style uses a different method for docking windows than the other styles. This is achieved by dragging the window over a *Navigation Compass* to specify a target destination or to dock the window into a tabbed location.

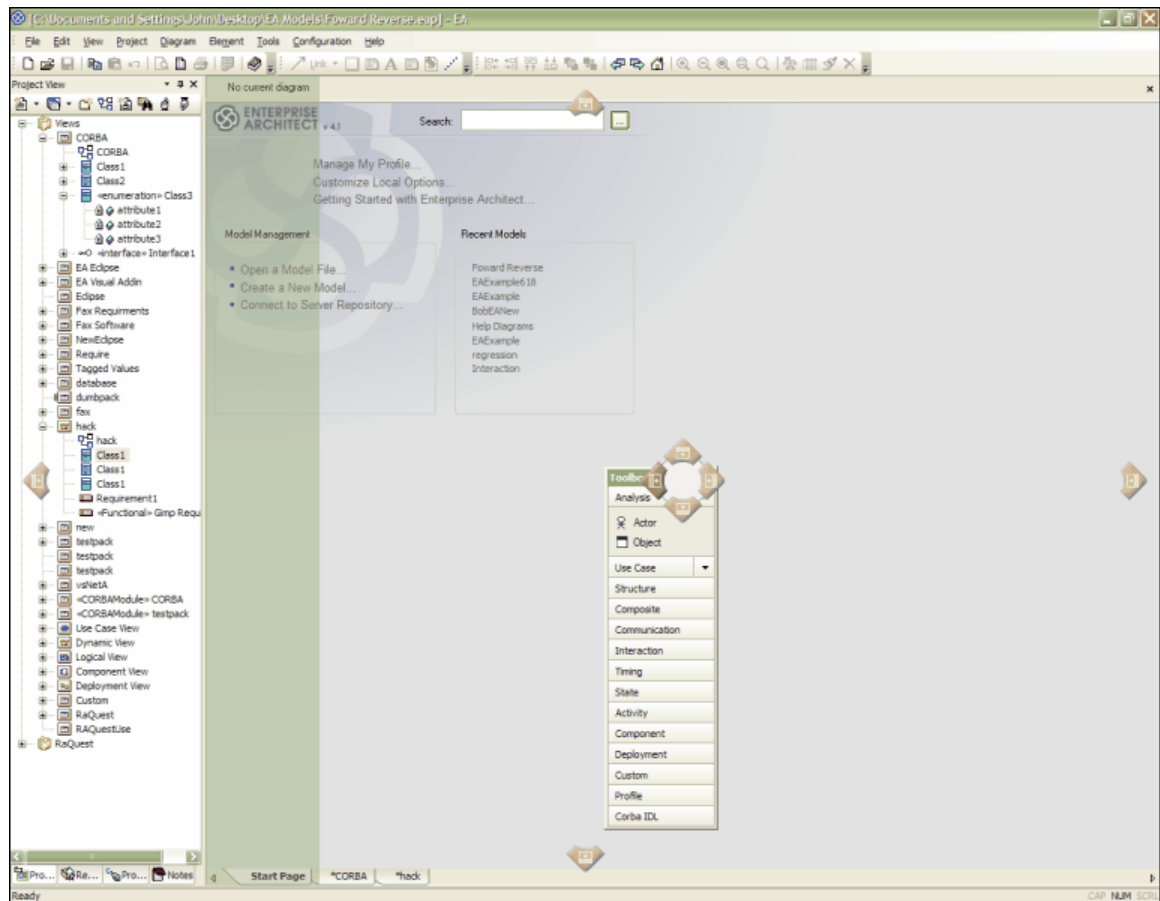
Moving a Window to a new location in EA

To move a window to a new destination use the following procedure:

- Select the item that you wish to move and start dragging it in the direction of its destination. This will activate the navigation compass (below). The navigation compass allows the user to dock a window at a desired destination by placing the window over one of the points of the compass. Moving the window to the middle of the compass will, when available, add the window to a tabbed set.



- When the window is dragged onto a compass point, the screen display will indicate the potential target destination by shading the area where the window will be placed. Releasing the mouse button over the compass point will confirm the destination and move the window. In the example below, the UML toolbar will be docked into the shaded area when the mouse button is released.



Moving a Window into a tabbed windows set

To move a window to tabbed windows group, follow these steps:

- Select the item that you wish to move and drag it over the window group that you wish to add the target window to. This will activate the navigation compass.
- Move the window to the centre of the navigation compass until the tabbed window icon becomes active (the window will disappear), then release the mouse button to confirm the selection.





- The window will now be added to the tabbed windows group

4.4.3 Autohide Windows

There are two ways to autohide your windows:

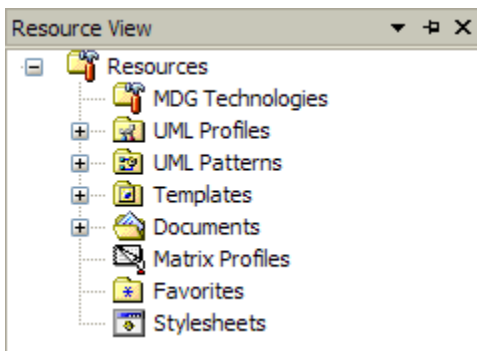
Autohide Using the Toggle Button

You can automatically hide browser frames and menus by clicking on the  button, located in the top right corner of the frame. To turn off the autohide for a particular set of menu within a frame, click the  button.

Using Automatically Hidden Windows

When you automatically hide a set of windows in a frame, the menu items contract to the outside of the application workspace.

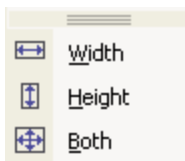
Hover over the icons with your mouse to access the menus.



Tip: You can also use the *View | Visual Style | Animate Autohide Windows* option to animate windows which have been automatically hidden.

4.4.4 Tear Off Menus

Some sub-menus in the EA main menu are tear off menus. This is indicated by the bar at the top. For example, the *Make Same* sub-menu in the *Element* menu is a tear off menu:



A tear off menu can be dragged out of the menu structure into its own window. Simply click on the bar at the top and drag.



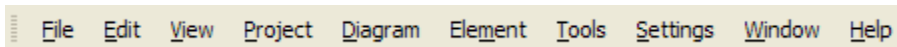
The menu will detach itself as shown here:



Once detached, the menu can also be docked in the toolbar section at the top of the screen, or on the edges of the workspace.

4.5 The Main Menu

The *Main Menu* provides mouse driven access to many high level functions related to the project life cycle, along with administration functions.



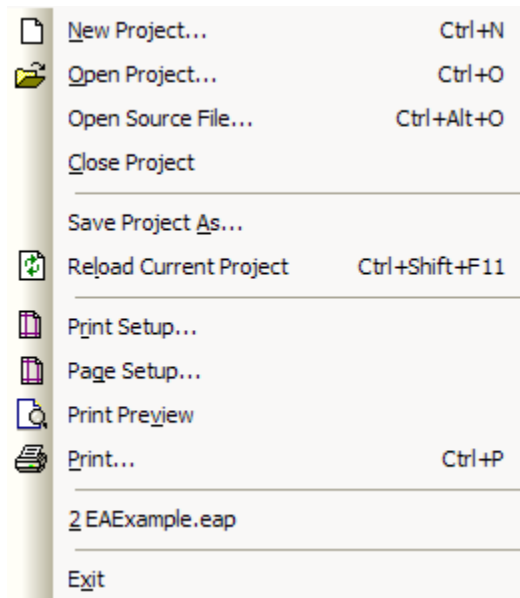
In order, the menus available are:

- The [File](#) menu
- The [Edit](#) menu
- The [View](#) menu
- The [Project](#) menu
- The [Diagram](#) menu
- The [Element](#) menu
- The [Tools](#) menu
- The [Settings](#) menu
- The [Window](#) menu
- The [Help](#) menu

The following section provides an overview of the functions available from the main menu and their general purpose.

4.5.1 The File Menu

The *File* menu provides options to create, open, close and save projects, and also to perform print tasks.



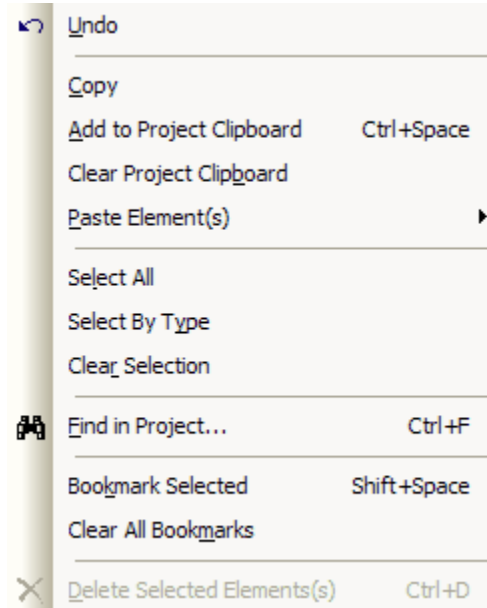
| Menu Item | Functionality |
|------------------------|--|
| New Project | Create a new Enterprise Architect project . [<i>Ctrl+N</i>] |
| Open Project | Open a project . [<i>Ctrl+O</i>] |
| Open Source File | Open a Source File [<i>Ctrl+Alt+O</i>] |
| Close | Close the current project. |
| Save Project As | Save the current project with a new name. |
| Reload Current Project | Reload the current project (use in a multi-user environment to refresh the Project Browser). [<i>Ctrl+Shift+F11</i>] |
| Page Setup | Configure the page settings for printing. |
| Print Setup | Configure your printer's settings. |
| Print Preview | Preview how the currently displayed diagram will print. |
| Print | Print the currently displayed diagram. [<i>Ctrl+P</i>] |
| Recent Files List | List the most recently opened projects, to a maximum of eight. |
| Exit | Exit Enterprise Architect. |

See Also

- [The Main Menu](#)
- [The Edit Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.2 The Edit Menu

The *Edit* menu provides the following functions which apply to diagram elements for the currently open diagram:



| Menu Item | Functionality |
|-----------------------------|--|
| Undo | Undo the last action performed. (Note: some actions cannot be undone.) |
| Copy | Copy the current selection into the buffer. |
| Add to Clipboard | Add the current element to the clipboard. [Ctrl+Space] |
| Clear Clipboard | Clear any elements from the clipboard. |
| Paste Element(s) | Paste clipboard elements into current diagram. See below. |
| Select All Elements | Select all elements concurrently on the current diagram. |
| Select By Type | Prompts you to specify which type of element you want to select. |
| Select No Elements | Deselect all elements. |
| Find | Allows you to search the entire project for particular phrases or words. [Ctrl+F] |
| Bookmark Selected | Bookmark the selected element(s). If the selected element is already bookmarked, selecting the Bookmark Selected menu item removes the bookmark. [Shift+Space] |
| Clear All Bookmarks | Clears bookmarks from any bookmarked elements in the current diagram. |
| Delete Selected Element (s) | Delete the selected element from the diagram. [Ctrl+D] |

The Paste Elements Sub-Menu

Paste what is in the buffer as either a new element or as a link to the element.

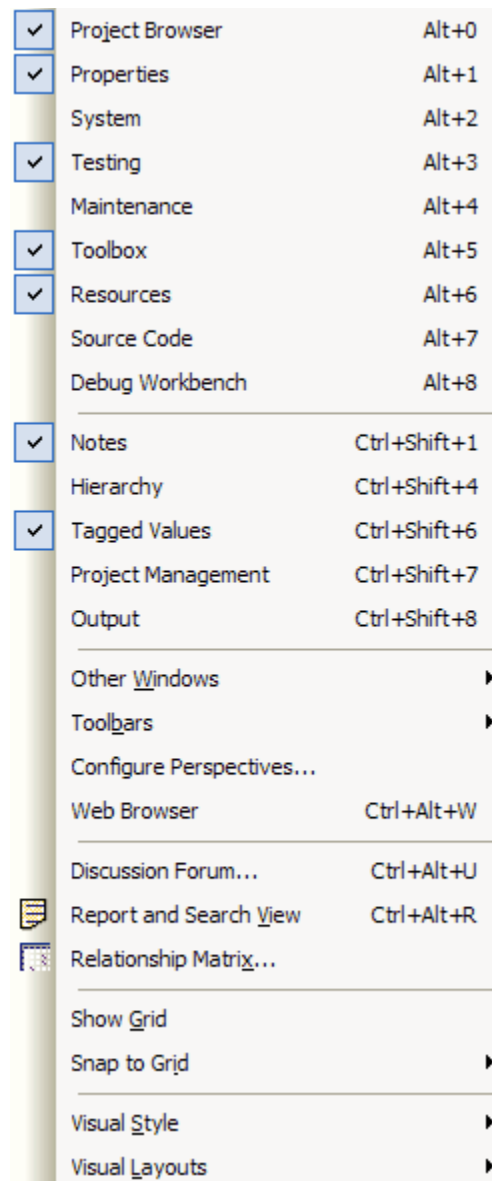
| Menu Item | Functionality |
|-------------------------|---|
| as Link | Paste the element(s) in the buffer as a link (ie. a reference) to the element. [Shortcut: Shift+Insert] |
| as New | Paste the element(s) in the buffer as a completely new element. [Shortcut: Ctrl+Shift+V] |

See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.3 The View Menu

From the *View* menu you may set local user preferences, including which toolbars or windows are hidden or visible.



| Element | Functionality |
|--|--|
| Project Browser | Check to show the Project Browser, uncheck to hide it. [Alt+0] |
| Properties | Check to show the Properties window, uncheck to hide it. [Alt+1] |
| System | Check to show the System window , uncheck to hide it. [Alt+2] |
| Testing | Check to show the Testing window , uncheck to hide it. [Alt+3] |
| Maintenance | Check to show the Maintenance window , uncheck to hide it. [Alt+4] |
| Toolbox | Check to show the Toolbox, uncheck to hide it. [Alt+5] |
| Resources | Check to show the Resource Browser, uncheck to hide it. [Alt+6] |
| Source Code | Check to show the Source Code Viewer , uncheck to hide it. [Alt+7] |
| Debug | Check to show the Debug window, uncheck to hide it. [Alt+8] |
| Notes | Check to show Notes, uncheck to hide them. [Ctrl+Shift+1] |
| Hierarchy | Check to show the Hierarchy window , uncheck to hide it. [Ctrl+Shift+4] |
| Tagged Values | Check to show the Tagged Values window , uncheck to hide it. [Ctrl+Shift+6] |
| Project Management | Check to show the Project Management window , uncheck to hide it. [Ctrl+Shift+7] |
| Output | Check to show the Output window , uncheck to hide it. [Ctrl+Shift+8] |
| Other Windows | See explanation below. |
| Toolbars | See explanation below. |
| Perspectives | Allows the user to alter the UML tools available to the UML modeler. |
| Web Browser | Open up a web browser window. [Ctrl+Alt+W] |
| Project Discussion Forum | Open the Project Forum [Ctrl+Alt+U] |
| Report and Search View | Display the current diagram in a Microsoft Outlook style report list. [Ctrl+Alt+R] |
| Relationship Matrix | Open the relationship matrix to cross reference elements to each other by connector type. |
| Show Grid | Check to show the grid, uncheck to hide it. |
| Snap to Grid | See explanation below. |
| Visual Style | See explanation below. |
| Visual Layouts | See explanation below. |

The Other Windows Sub-Menu

Check the windows you want visible and uncheck those you want hidden. You can select from:

- [Element Browser](#)
- [Relationships](#) [[Ctrl+Shift+2](#)]
- [Rules](#) [[Ctrl+Shift+3](#)]
- [Instant Help](#) [[Ctrl+Shift+9](#)]

Note: This sub-menu is a [tear off menu](#).

The Toolbars Sub-Menu

Check the toolbars you want visible and uncheck those you want hidden. You can select from:

- [Default Tools](#)
- [Project](#)
- [Code Generation](#)
- [UML Elements](#)
- [Diagram](#)
- [Current Element](#)
- [Current Connector](#)
- [Format Tool](#)
- [Status Bar](#)

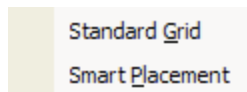
- [Workspace Views](#)

Note: This sub-menu is a [tear off menu](#).

The Snap to Grid Sub-Menu

The Snap to Grid menu offers two possibilities Standard Grid and Smart Placement.

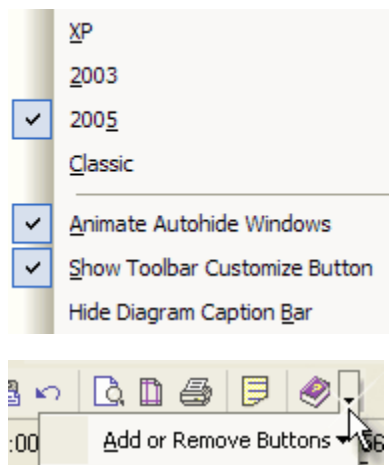
- Standard Grid constrains elements to the grid when they are added to diagrams.
- Smart Placement places elements even distances away from other elements and spaces elements evenly.
- If the Standard Grid or the Smart Placement options are not enabled the elements can be placed freely on the diagram.



The Visual Style Sub-Menu

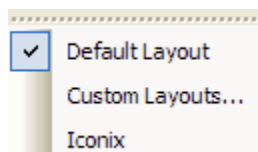
Display windows with the following [visual styles](#):

- A flat XP look and feel.
- The 2003 style look and feel.
- The modern 2005 look and feel.
- Classic Windows look and feel.
- You can also use the Animate Autohide Windows option to animate windows which have been [automatically hidden](#).
- The Show Toolbar Customize Button option toggles the button on the end of the toolbar that allows you to customize the toolbar buttons, as shown below:
- The Always on Top option ensures that the topmost status is preserved between sessions.



The Visual Layouts Sub-Menu

Set the layout of docked windows, toolbars and UML Toolbox to a custom layout. Current options are the default layout, your own user layout or Iconix layout for working with the Iconix process.

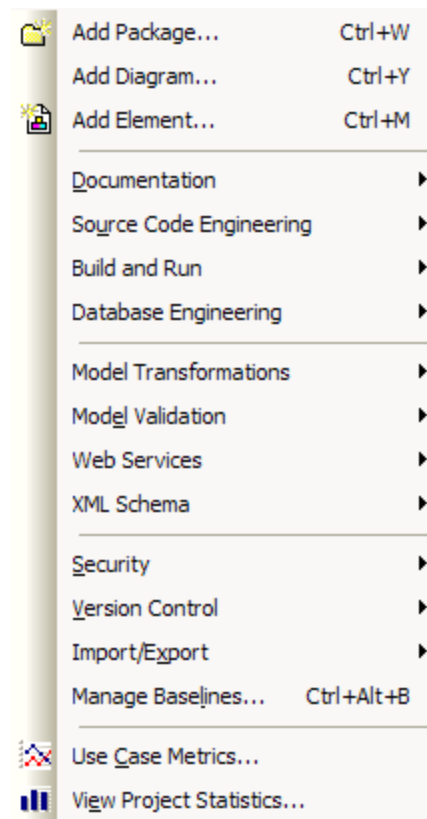


See Also

- [Custom Layouts](#)
- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.4 The Project Menu

Use the *Project Menu* for tasks related to the management of your project - recording issues, setting estimation parameters and compiling a glossary.



| Menu Item | Functionality |
|-----------------------------|--|
| Add Package | Create a new package. [<i>Ctrl+W</i>] |
| Add Diagram | Create a new diagram in the current package [<i>Ctrl+Y</i>] |
| Add Element | Create a new element on the current diagram. [<i>Ctrl+M</i>] |
| Documentation | See below. |

| | |
|----------------------------------|---|
| Source Code Engineering | See below. |
| Build and Run | See below. |
| Database Engineering | See below. |
| Model Transformations | See below. |
| Model Validation | See below. |
| Transformations | See below. |
| Web Services | See below. |
| XML Schema | See below. |
| Security | See below. |
| Version Control | See below. |
| Import/Export | See below. |
| Manage Baselines | Available in the Corporate Edition Only, baselines allow a model branch to be stored as a snapshot. [Ctrl+Alt+B] |
| Use Case Metrics | Set Use Case Metrics to assist in estimating project size. |
| View Project Statistics | View some basic project statistics. |

The Documentation Sub-Menu

Note: This sub-menu is a [tear off menu](#).

Generate documentation of various types.

| Element | Description |
|---|--|
| Rich Text Format Report | Generate a report for the currently selected package in rich text format. [F8] |
| HTML Report | Generate a report for the currently selected package in HTML format. [Shift+F8] |
| Diagrams Only Report | Generate a RTF report containing only a selection of diagrams. [Ctrl+Shift+F8] |
| Testing Report | Generate a RTF report of the model's existing test documentation. |
| Issues | Generate a RTF report of the models Issues. |
| Glossary | Generate a RTF report of the models Glossary. |
| Implementation Details | Generate an implementation report for the currently selected package. |
| Dependency Details | Generate a dependency report for the currently selected package. |
| Testing | Generate test details for the currently selected package. |
| Resource and Tasking Details | View resource details . |

The Source Code Engineering Sub-Menu

Note: This sub-menu is a [tear off menu](#).

Forward and reverse engineer code using the language of your choice.

| Element | Description |
|--|--|
| Generate Package Source Code | Generate source code for the currently selected package. [Ctrl+Alt+K] |
| Synchronize Package Contents | Synchronize selected package with the source code. [Ctrl+Alt+M] |
| Import Directory Structure | Reverse engineer an entire directory structure. [Ctrl+Shift+U] |
| Import Binary Module | Import a binary module (Corporate and Professional Editions only). |
| Import ActionScript Files | Import code written in ActionScript with the file extension .as. |

| | |
|---|--|
| Import C# Files | Import code written in the C# programming language with the file extension .CS . |
| Import C++ Files | Import code written in the C++ programming language with the file extension .H, .HPP, or .HH. |
| Import Delphi Files | Import code written in the Delphi programming language with the file extension .PAS. |
| Import Java Files | Import code written in the Java programming language with the file extension .JAVA. |
| Import PHP Files | Import code written in PHP with the file extension .PHP, .PHP4, .INC. |
| Import Python Files | Import code written in Python with the file extension .py |
| Import Visual Basic Files | Import code written in the Visual Basic programming language with the file extension .FRM, .CLS, .BAS or .CTL. |
| Import VB.Net Files | Import code written in the VB.Net programming language with the file extension .VB. |

The Build and Run Sub-Menu

Link your project with a compiler for building, running and debugging.

| Element | Description |
|---|---|
| Package Build Scripts | Create and configure compiler scripts. [<i>Shift+F12</i>] |
| Build | Build the application for your current script. Executes the Build script in the Source Code Configuration dialog [<i>Ctrl+Shift+F12</i>] |
| Test | Executes your Test script you configured in the Source Code Configuration dialog. [<i>Ctrl+Alt+F12</i>] |
| Run | [<i>Ctrl+F12</i>] |
| Deploy | [<i>Ctrl+Shift+Alt+F12</i>] |
| Debug Run | Run the application. Executes the Run script in Source Code Configuration dialog [<i>Ctrl+Alt+F5</i>] |
| Step Into | Steps into the current function. [<i>Ctrl+Alt+F6</i>] |
| Step Over | Steps over the current function. [<i>Ctrl+Alt+F7</i>] |
| Step Out | Steps out of the current function. [<i>Ctrl+Alt+F8</i>] |
| Debug Stop | Stops the current debug session. [<i>Ctrl+Alt+F9</i>] |
| Start Debug Recording | Start recording your trace for a debug session. |
| Stop Debug Recording | Stop the recording. |
| Create Sequence Diagram | Create a sequence diagram from the Stack Trace History . |

The Database Engineering Sub-Menu

| Element | Description |
|--|---|
| Import DB Schema from ODBC | Import a database schema from an ODBC data source. |
| Generate Package DDL | Generate a DDL script to create the tables in the currently selected package. |

The Transformations Sub-Menu

| Element | Description |
|---------|-------------|
|---------|-------------|

| | |
|---|---|
| Transform Selected Elements | Perform a MDA-Style transformation to the currently selected elements. [<i>Ctrl+Alt+F</i>] |
| Transform Current Package | Perform a MDA-Style transformation to the currently selected package. [<i>Ctrl+Shift+H</i>] |

The Model Validation Sub-Menu

Note: This sub-menu is a [tear off menu](#).

Perform Model Validation.

| Element | Description |
|-------------------|--|
| Validate Selected | Validate a select Element, Diagram or Package from the project view. [<i>Ctrl+Alt+V</i>] |
| Cancel Validation | Cancel the Validation Process. |
| Configure | Configure the Validation Rules from the list of available Rules. |

The Web Services Sub-Menu

| Element | Description |
|---------------|---|
| Import WSDL | Use the WSDL Import function. |
| Generate WSDL | Use the Import WSDL function. |

The XML Schema Sub-Menu

| Element | Description |
|---------------------|---|
| Import XML Schema | Use the XML Schema Import facility. |
| Generate XML Schema | Use the XML Schema Generation facility. |

The Security Sub-Menu

Note: This feature is available in the Corporate Edition only.

Note: This sub-menu is a [tear off menu](#).

Configure security settings for your project.

| Element | Description |
|---------------------------------------|---|
| Maintain Users | Add, modify and remove users, including maintaining permissions. |
| Maintain Groups | Add, modify and remove security groups, including maintaining permissions. |
| View and Manage Locks | View and manage element locks. |
| Change Password | Change current security password. |
| Login as Another User | Switch login to a different user. |
| Manage My Locks | View and delete user level locks. [<i>Ctrl+Shift+L</i>] |
| Enable Security | Enable or disable user security to limit access to update functions in the model. |
| Require User Lock to Edit | Allows you to control the security policy . |
| Encrypt Password | Add encryption to your password. |

The Version Control Sub-Menu

| Element | Description |
|---------|-------------|
|---------|-------------|

| | |
|--|--|
| Package Control | Specify whether this package (and its children) is controlled, and if so, which file it is controlled through. [Ctrl+Alt+P] |
| Set Version Control Options | Allows you to specify the options required to connect to a Source Code Control (SCC) provider. |
| Work Offline | Disconnects version control from the network. |
| Start Version Control Explorer | Asks the SCC provider to display an interface allowing you to review all SCC projects and their contents directly. |

The Import/Export Sub-Menu

Note: This sub-menu is a [tear off menu](#).

Perform import and export to XMI and CSV.

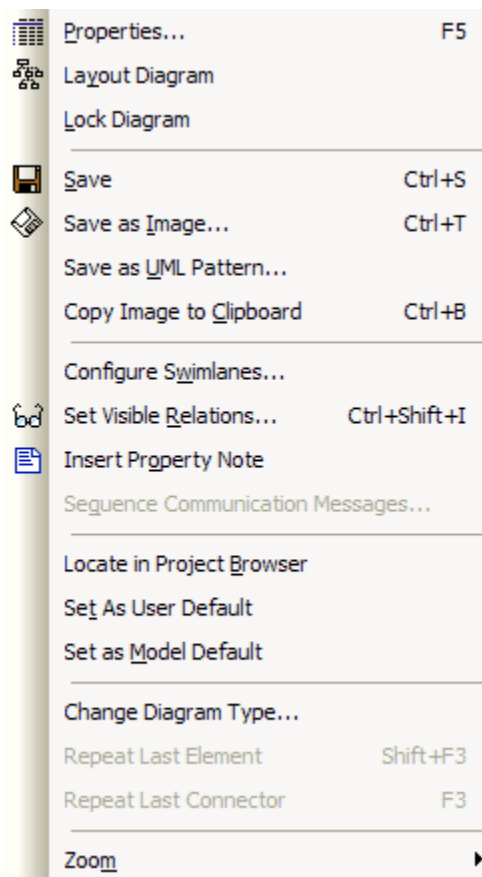
| Element | Description |
|--|---|
| Import Package from XMI | Import a package from an XMI (XML based) file. [Ctrl+Alt+I] |
| Export Package to XMI | Export the currently selected package to an XMI (XML based) file. [Ctrl+Alt+E] |
| CSV Import/Export | Import or Export information about EA elements in CSV format. [Ctrl+Alt+C] |
| CSV Import/Export Specifications | Set up CSV import Export Specifications. |
| Batch XMI Export | Export a group of controlled packages in one action. |
| Batch XMI Import | Run a batch import of multiple packages. |

See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)
- [The View Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.5 The Diagram Menu

The *Diagram Menu* allows the user to save diagram images to file as well as configure diagram properties and options.



| Element | Description |
|---|---|
| Properties | View and edit the property page for the current diagram. [F5] |
| Layout Diagram | Configure automatic diagram layout settings. |
| Lock Diagram | Prevents the diagram from being edited. |
| Save | Save the current position of all diagram elements. [Ctrl+S] |
| Save Image to File | Save the diagram as a bitmap (.BMP), GIF (.GIF) or Windows Metafile (.WMF). [Ctrl+T] |
| Save Image to Clipboard | Copy the current diagram to clipboard. [Ctrl+B] |
| Save as UML Pattern | Save the current diagram as a UML pattern. |
| Configure Swimlanes | Add, modify and delete swimlanes for the current diagram. |
| Set Visible Relations | Hide or show individual links for the current diagram. [Ctrl+Shift+I] |
| Insert Property Note | Display the properties for the current diagram on screen. |
| Sequence Communication Messages | Change the order of the communication messages in the current diagram. |
| Locate in Browser | Locate the current diagram in the Project Browser. |
| Set as User Default | If security is enabled, the User Default diagram overrides the Model Default diagram (see below). |
| Set as Model Default | Set the current diagram as the default diagram opened when the currently loaded model is opened. |
| Change Diagram Type | Change the type of the current diagram. |
| Repeat Last Element | Create an instance of the same type as the last element created. [Shift+T] |
| Repeat Last Connector | Create an instance of the same type as the last connector created. [F3] |
| Zoom | Change the zoom factor on the current diagram - see below. |

See: [Diagram Tabs](#)

The Zoom Sub-Menu

Note: This sub-menu is a [tear off menu](#).

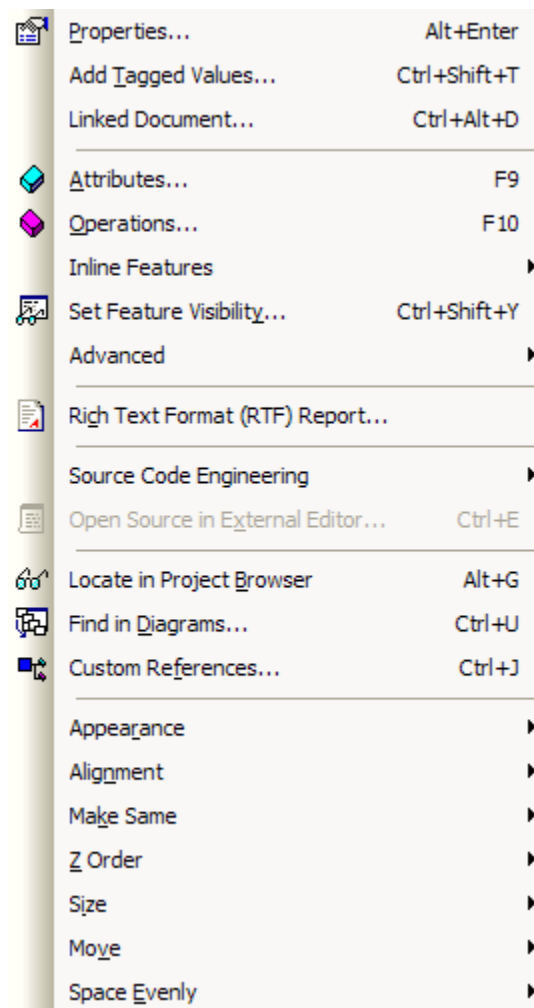
[Zoom](#) in 10%, Zoom Out 10%, Zoom to 100% and Fit to Window.

See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.6 The Element Menu

Element details can be configured and accessed using the *Element Menu*. Element layout can be controlled, documentation can be generated and project resources can be managed.



| Element | Description |
|--------------------------------|---|
| Properties | View the properties window of the selected element. [<i>Alt+Enter</i>] |
| Tagged Value | Add a tagged value to the currently selected element. [<i>Ctrl+Shift+T</i>] |
| Linked Document | Link any element to a rich text document. |
| Attributes | View and edit the attributes for the selected element. [<i>F9</i>] |
| Operations | View and edit the operations for the selected element. [<i>F10</i>] |
| Inline Features | See below. |
| Set Feature Visibility | Set various states relating to the selected element(s) visibility. [<i>Ctrl+Shift+Y</i>] |
| Advanced | See below. |
| Documentation | Generate a report for the currently selected package in rich text format . |
| Source Code Engineering | See below. |
| Open Source in External Editor | Open the source code of the selected class in the default external editor for that language. (Source code must have been generated , and the selected element must be a class.) [<i>Ctrl+E or F12</i>]. |
| Locate in Browser | Locate the currently selected element in the Project Browser. [<i>Alt+G</i>] |
| Find in Diagrams | Display all occurrences of the currently selected element. [<i>Ctrl+U</i>] |
| Show Other References | Show model element cross references . [<i>Ctrl+J</i>] |
| Appearance | See below. |

| | |
|--------------|------------|
| Alignment | See below. |
| Make Same | See below. |
| Z Order | See below. |
| Size | See below. |
| Move | See below. |
| Space Evenly | See below. |

The Inline Features Sub-Menu

The Inline Features sub-menu gives you various options to directly edit class diagram model elements from the class diagram.

| Menu Option | Description |
|----------------------------|---|
| Edit Selected | Attach a note or attach a constraint to the element. [F2] |
| View Properties | Opens up the dialog window containing details of the selected element feature, or the element if no feature is selected. |
| Insert New After Selected | Inserts a new item to the element. [Ctrl+Shift+Insert] |
| Add Attribute | Adds an attribute to the element. [Ctrl+Shift+F9] |
| Add Operation | Adds an operation to the element. [Ctrl+Shift+F10] |
| Add Other | Allows the user to insert a feature on the specific element item, such as Maintenance features and Testing features. [Ctrl+F11] |
| Delete Selected from Model | Deletes the selected item from the model. [Ctrl+Shift+Delete] |

Other options that are available while in editing elements mode in a diagram (when an attribute or operation is highlighted):

| Hotkey | Description |
|---------------------|--|
| <i>Enter</i> | Accept current changes |
| <i>Ctrl + Enter</i> | Accept current changes and open a new slot to add a new item |
| <i>Esc</i> | Abort edit, without save |
| <i>Shift + F10</i> | Context menu for in-place editing |
| <i>Ctrl + Space</i> | Invoke Classifier Dialog |

The Advanced Sub-Menu

The Appearance sub-menu gives you various options to choose from to customize the appearance of model elements.

| Element | Description |
|--|---|
| Override Implementation | Automatically override methods from parent classes and from realized interfaces. [Ctrl+Shift+O] |
| Set Parents and Interfaces | Manually set an element's parent or an interface it realizes. [Ctrl+I] |
| Embedded Elements | Enable you the attach elements to currently selected element. [Ctrl+Shift+B] |
| Change Type | Change the element type of the selected element. |

The Source Code Engineering Sub-Menu

Note: This sub-menu is a [tear off menu](#).

Forward and reverse engineer code using the language of your choice.

| Element | Description |
|--------------------------|--|
| Generate Current Element | Generate source code for the currently selected element. [Ctrl+G or F11] |

| | |
|---------------------------------------|---|
| Synchronize Current Element | Synchronize selected class with source code. [<i>Ctrl+R</i> or <i>F7</i>] |
| Batch Generate Selected Element(s) | Batch generate source code for the currently selected element(s). [<i>Shift+F11</i>] |
| Batch Synchronize Selected Element(s) | Batch synchronize the currently selected element(s) with source code. [<i>Ctrl+R</i>] |

The Appearance Sub-Menu

The Appearance sub-menu gives you various options to choose from to customize the appearance of model elements.

| Element | Description |
|--|---|
| Autosize Selected Elements | Auto-size a group of elements in a diagram to a best fit. [<i>Alt+Z</i>] |
| Configure Default Appearance | Set border, font, background color and border thickness for the selected element. [<i>Ctrl+Shift+E</i>] |
| Select Alternate Image | Select an alternative image for the selected element. [<i>Ctrl+Shift+W</i>] |

The Alignment Sub-Menu

Use the Alignment sub-menu to align the selected element(s) to each other.

| Element | Description |
|---------|---|
| Left | Align left edges of elements [<i>Ctrl+Alt+Left</i>] |
| Right | Align right edges of elements [<i>Ctrl+Alt+Right</i>] |
| Top | Align top edges of elements [<i>Ctrl+Alt+Up</i>] |
| Bottom | Align bottom edges of elements [<i>Ctrl+Alt+Down</i>] |
| Centres | Align centres of elements, horizontally or vertically |

The Make Same Sub-Menu

Use the Make Same sub-menu to make the selected elements the same width, height or both.

The Z Order Sub-Menu

Use the Z Order sub-menu to bring the selected element(s) back, forward, to the top or bottom.

The Size Sub-Menu

Use the Size sub-menu to make the selected element(s) wider, narrower, taller or shorter by one increment.

The Move Sub-Menu

Use the Move sub-menu to move the selected element(s) left, right, up or down by one increment.

The Space Evenly Sub-Menu

Use the Space sub-menu to distribute the selected elements evenly.

| Element | Description |
|---------|---|
| Across | Space elements evenly, horizontally. [<i>Alt+=</i>] |
| Down | Space elements evenly, vertically. [<i>Alt+=</i>] |

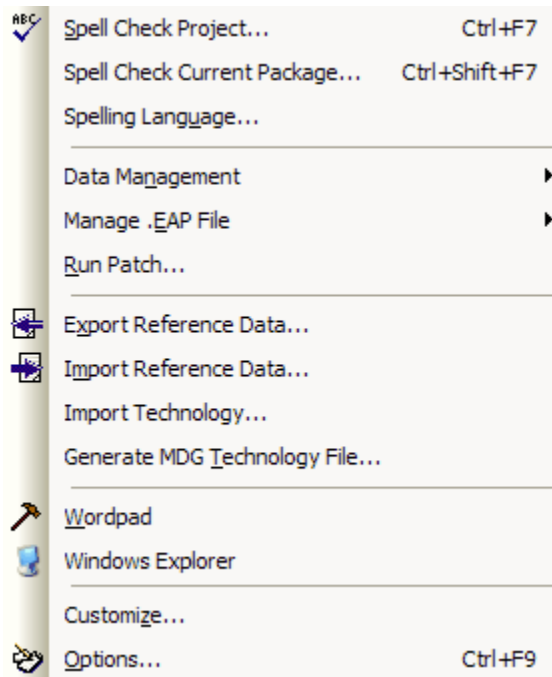
See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)

- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.7 The Tools Menu

The *Tools Menu* provides access to various tools including those related to code engineering, managing .EAP files, spelling options, external resources and customization of features such as configuring shortcuts.



| Element | Description |
|---------------------------------------|---|
| Spell Check Model | Spell check the current model. [<i>Ctrl+F7</i>] |
| Spell Check Current Package | Spell check the current package. [<i>Ctrl+Shift+F7</i>] |
| Spelling Language | Specify language to use for spell checking. |
| Data Management | See below. |
| Manage .EAP File | See below. |
| Run Patch | Execute a SQL patch . |
| Export Reference Data | Export reference data to XML files for convenient model updating. |
| Import Reference Data | Import reference data from XML files for convenient model updating. |
| Import Technology | Import Technology file. |
| Generate MDG Technology File | See Creating MDG Technologies . |
| Wordpad | Open Wordpad. |
| Windows Explorer | Open Windows Explorer. |
| AutInt | Open the Automation Interface program. |
| Customize | Customize the operation of EA. |

| | |
|---------|---|
| Options | Customize your general settings through the local options dialog . [<i>Ctrl+F9</i>] |
|---------|---|

The Data Management Sub-Menu

Manage your project's data.

| Element | Description |
|--------------------------------|--|
| Data Transfer | Move a complete model from one repository to another. |
| Data Compare | Compare the total model size of one model and another. |
| Data Integrity | Check data integrity. [<i>Shift+F9</i>] |

The Manage .EAP File Sub-Menu

Repair, compact or replicate your .EAP file.

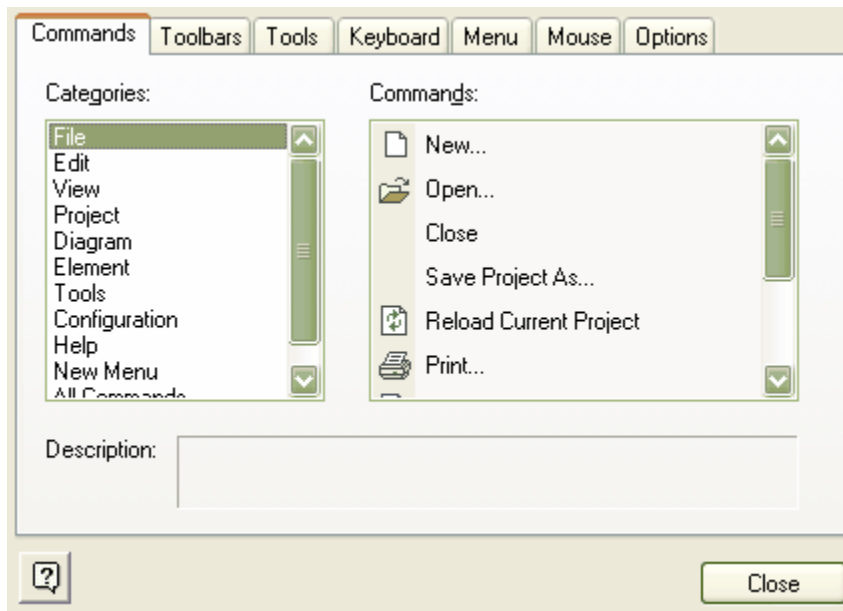
| Element | Description |
|---|---|
| Repair .EAP File | Repair an Enterprise Architect project. If a project has not been closed properly, in rare cases it may not open correctly. This option attempts to repair such projects. Note: <i>All users must be logged off the project while it is being repaired.</i> |
| Compact .EAP File | Compact an EA project. Eventually projects may benefit from compacting to conserve space. Note: <i>Ensure everyone is logged off the target project, then select this option to compact it.</i> |
| Make Design Master | Make a design master project - this is the master project for creating replicas. |
| Create New Replica | Create a new replica from the Design Master . |
| Synchronize Replicas | Copy changes from one replica set member to another. |
| Remove Replication | Remove all replication features if you no longer require a model to be replicable. |
| Resolve Replication Conflicts | Resolve any conflicts caused when multiple users have changed the same element between synchronization points. |

See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.7.1 The Customize Window

The *Customize* window allows you to customize [Commands](#), [Toolbars](#), [Tools](#), [Keyboard](#), [Menu](#) and [Options](#) within EA.



4.5.7.1.1 Customize Commands

The *Customize* window *Commands* tab provides access to many of EA's functions, for the purpose of placing them into a toolbar. To add a command to a toolbar, find the category in the left scrollbox and the available commands for each category in the right. When you have located a command you wish to add, simply select it from the list on the right and drag it on top of the toolbar you wish to add it to.

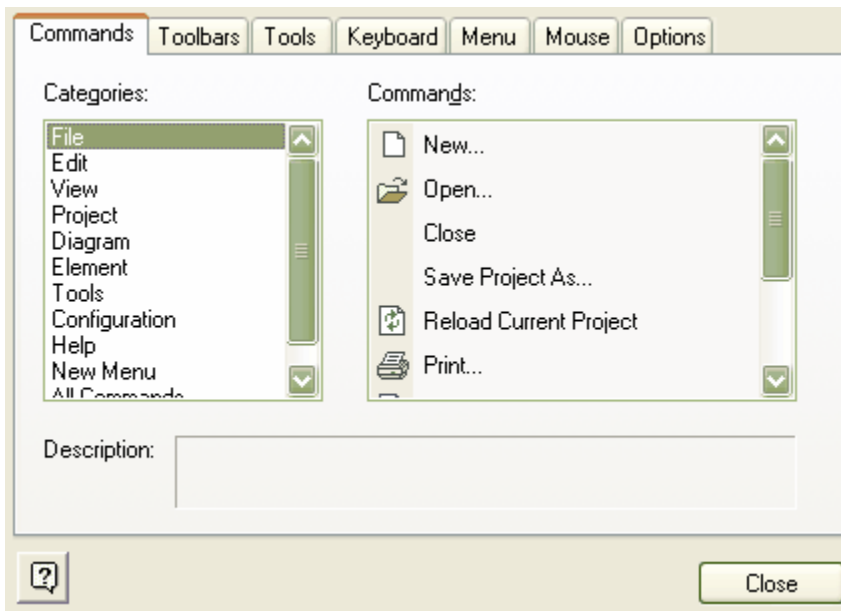
Right-clicking on the command icon in the toolbar while the customize window is open will display a context-sensitive menu. This menu offers options for deleting commands from a toolbar, and for changing the appearance of commands.

To remove a tool, right-click on the tool graphic or text and select *Delete*.

To change the appearance of a tool, right click on the tool graphic or text and select *Button Appearance...*. This way you can add graphical icons for commands that do not have them by default.

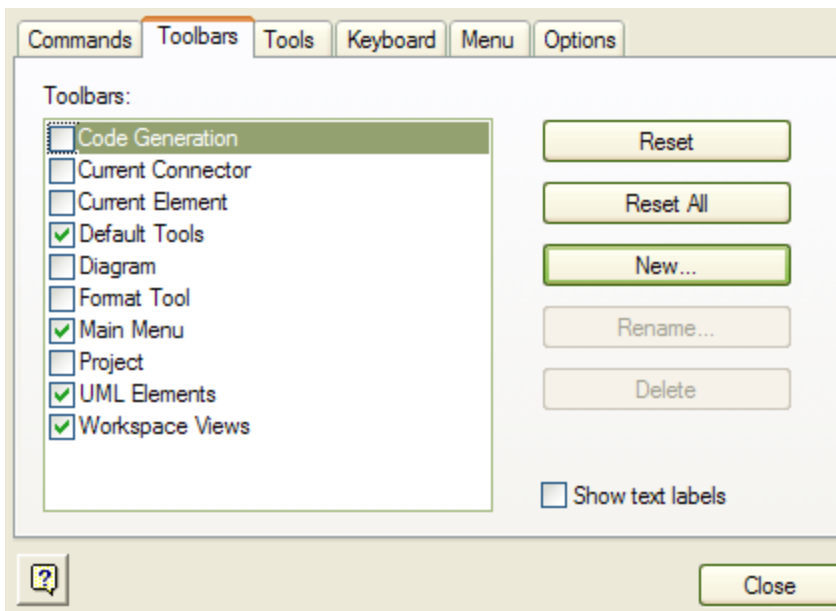
Note: Some commands do not come with a convenient icon, which will result in an empty toolbar button. Either avoid placing these commands on toolbars or use the context-sensitive menu to select an appropriate icon for the command.

Tip: Read the [Create a New Toolbar and Populate it with Commands](#) section of the *Customize Toolbars* topic.



4.5.7.1.2 Customize Toolbars

The **Toolbars** tab on the **Customize** window allows you New toolbars may also be created and configured as required.



To access the **Toolbars** tab, from the **Tools** menu select **Customize** -OR- click on the small drop arrow at the far right of a toolbar and select the **Customize** option.

Using the **Toolbars** tab you can:

- Hide or show toolbars by checking the appropriate check box.
- Rename toolbars.
- Create new toolbars.

- Delete toolbars.
- Modify toolbar contents by dragging commands from the [Command](#) tab onto a visible toolbar.
- Reset a toolbar to its default contents and position.

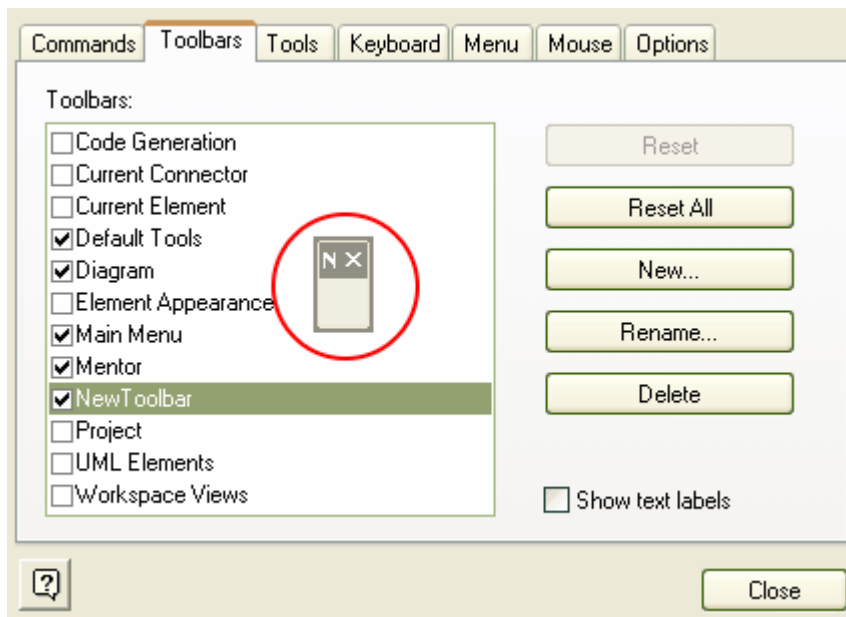
Create a New Toolbar and Populate it with Commands

To create a new toolbar and populate it with commands:

1. Go to the **Tools** menu and select the **Customize** menu option.
2. Select the **Toolbars** tab.
3. Press **New**.
4. The dialog will appear. Enter a name for your new toolbar and press **OK**.

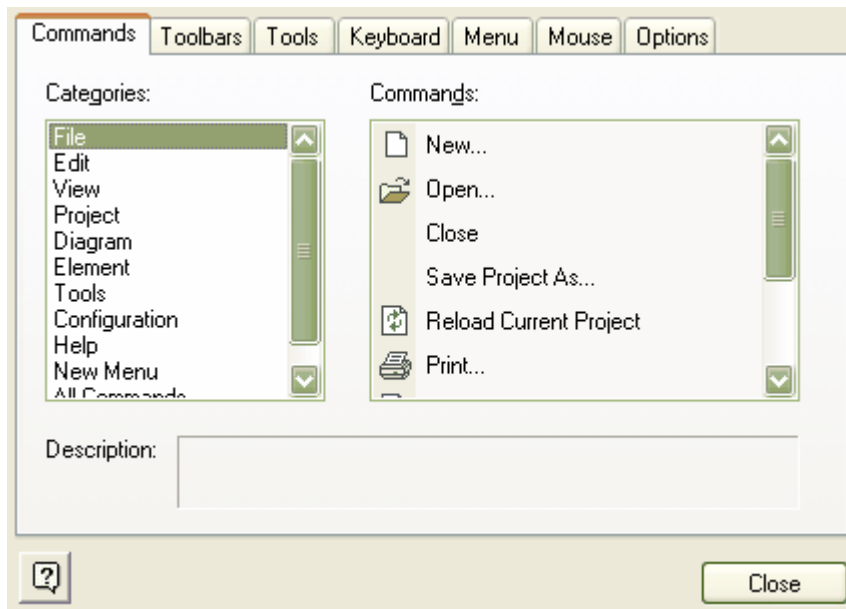


5. Your new toolbar will be created and shown at the front of the **Customize** dialog (see the red circle below).



Note: You can check the *Show text labels* check box to display textual descriptions of toolbar items.

6. Now add commands to your toolbar. Go to the **Commands** tab. This will force your new toolbar behind the **Customize** dialog, so you will need to drag the **Customize** dialog to the side to find your new toolbar.



7. Find the command you wish to add to your toolbar in the *Commands* list. The *Categories* list on the left represents the EA menu structure and the *Commands* list updates each time you click on a different category.
8. When you have located a command, **drag it from the list into the new toolbar**.
9. Your toolbar should look like this, if you checked the *Show Text Labels* check box:



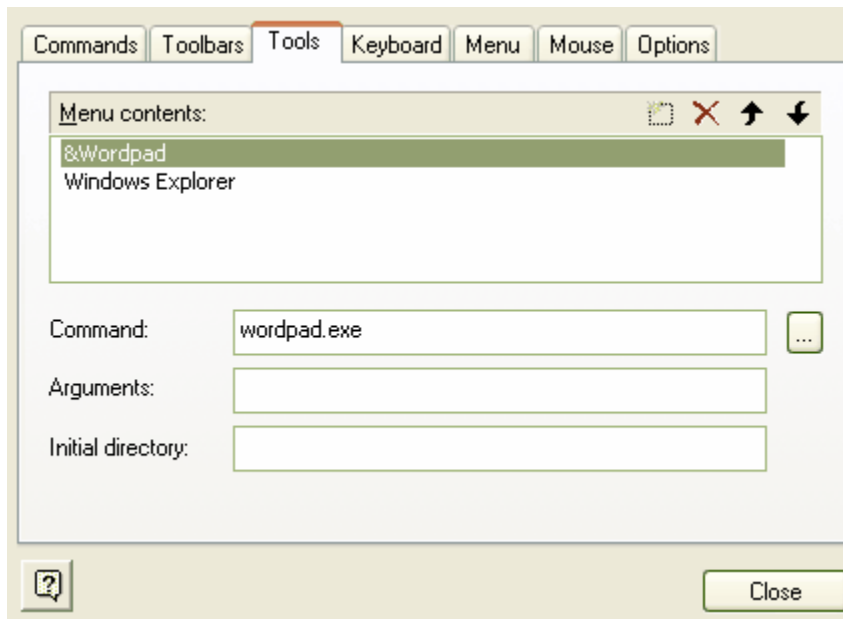
or this - if you didn't select the *Show Text Labels* check box:



You can add as many commands to your toolbar as you need. Your new toolbar behaves the same as other toolbars - it can be positioned next to the other toolbars at the top of the application workspace, docked to the side of the workspace, or closed.

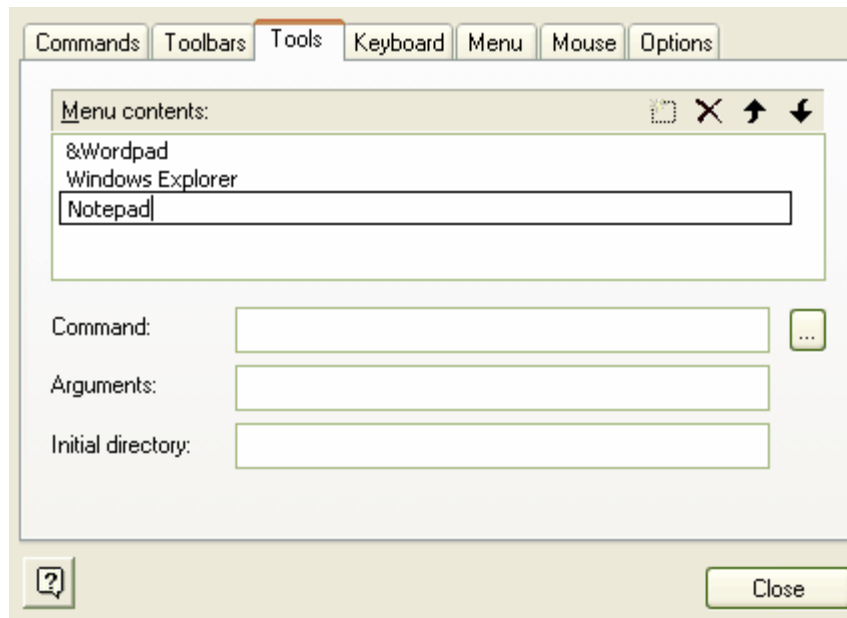
4.5.7.1.3 Custom Tools

The *Tools* tab on the *Customize* window provides a means of extending the power of the EA desktop. From here you can configure custom tools and make them accessible from the Main Menu. You can create menu options that link to different applications, compilers, batch scripts, automation scripts, URLs or documentation.



Add and Configure Custom Tools

1. From the **Tools** menu, select the **Customize** option.
2. Select the **Tools** tab.
3. Press the **New** symbol (at left of the red 'X') and type in the name of the tool as you want it to appear in the menu.

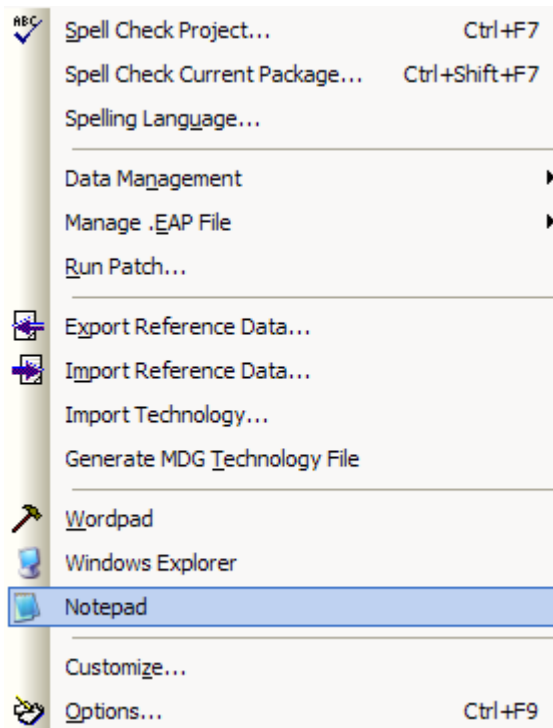


4. Enter the tool you wish to use in the **Command** field - the tool must be a valid filename.

Note: Programs which were installed with your operating system (eg. Notepad, Wordpad) do not require a

full file path. Programs installed separately (eg. Microsoft Visual Studio) require the full file path in the **Command** textbox. Use the **Browse [...]** button to locate the tool in the file system (see [Using Parameters](#)).

5. Press **OK** in the **Open** dialog when you have located the tool. The correct file path will appear in the **Command** field.
6. Add any arguments required by the tool (see [Opening External Tools](#) and [Passing Parameters to External Applications](#)), and specify an initial directory if you wish.
7. Close the **Customize** window. Your tool should have now been added to the **Tools** menu as shown below.



4.5.7.1.3.1 Opening External Tools

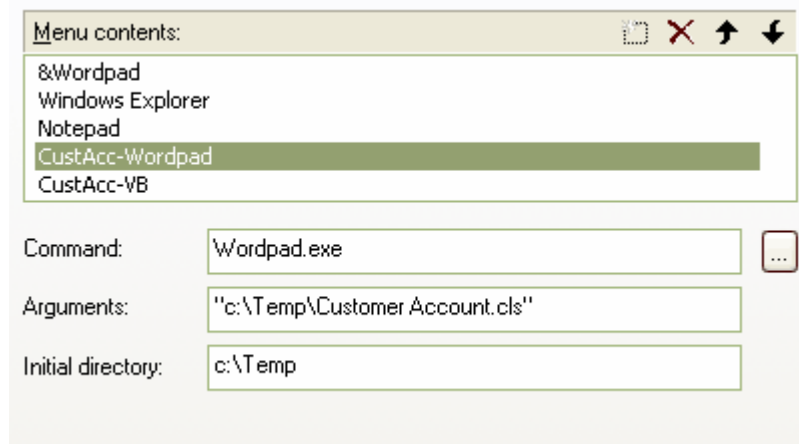
When configuring custom tools in EA, you may specify a file to be opened by the external application.

From the main menu select the **Tools | Customize** option. Select the **Tools** tab in the **Customize** dialog. Now you can:

- Specify a [custom tool](#) (application) using the **Command** field.
- Define file to open or [parameters to pass](#) to this application using the **Arguments** field.

Example 1

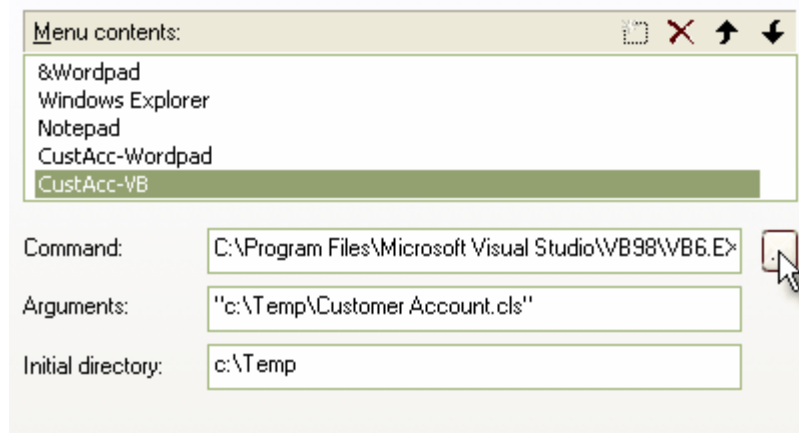
This example opens the file `c:\Temp\Customer Account.cls` using Wordpad. If you save from within Wordpad the initial directory will be `c:\Temp`.



Tip: If there are any spaces in the paths for **Command**, **Argument** or **Initial Directory**, you must enclose the whole path in double quotes. For example: "c:\Temp\Customer Account.cls" needs quotes but c:\Temp\CustomerAccount.cls does not.

Example 2

This example opens the file `c:\Temp\Customer Account.cls` using VB. As VB is not installed with the operating system, the whole file path for VB must be included in the **Command** field - this can be entered using the Browse [...] button to locate the VB executable. If you save from within VB the initial directory will be `c:\Temp`.

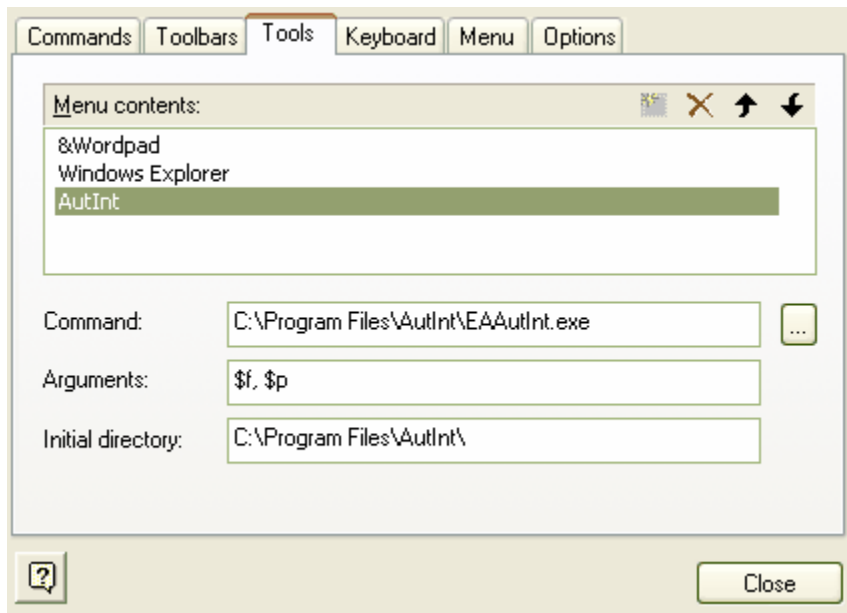


4.5.7.1.3.2 Passing Parameters to External Applications

When configuring custom tools in EA, you may pass parameters to the application.

From the main menu select the **Tools | Customize** option. Select the **Tools** tab in the **Customize** dialog. Now you can:

- Specify a [custom tool](#) (application) using the **Command** field.
- Define [file to open](#) or parameters to pass to this application using the **Arguments** field.



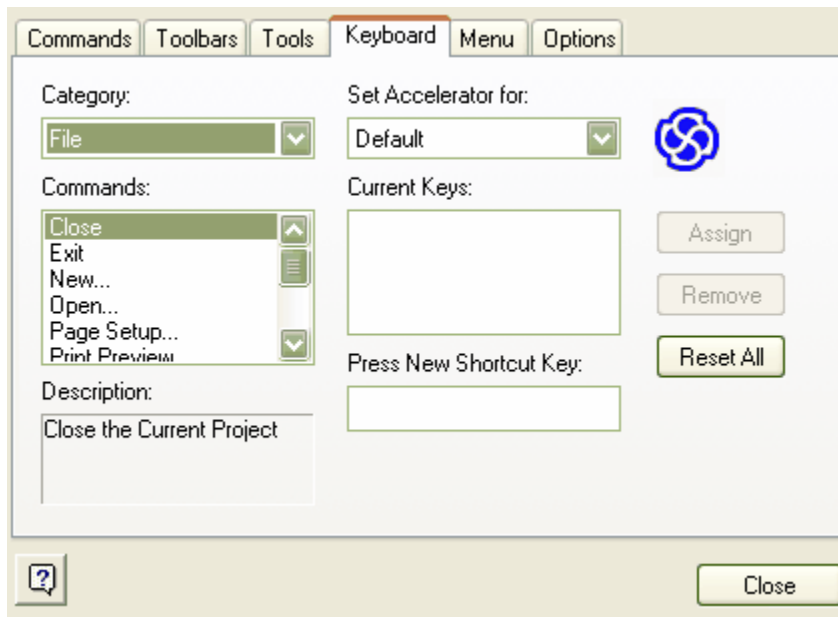
The available parameters for passing information to external applications are:

| Parameter | Description | Notes |
|-----------|---------------------------------------|--|
| \$f | Project Name | i.e.: c:\projects\EAexample.eap |
| \$F | Calling Application (EA) | 'EA' |
| \$p | Current Package Id | ie 144 |
| \$P | Package GUID | GUID for accessing this package |
| \$d | Diagram ID | Id for accessing associated diagram |
| \$D | Diagram GUID | GUID for accessing associated diagram |
| \$e | Comma separated list of element IDs | All elements selected in the current diagram |
| \$E | Comma separated list of element GUIDs | All elements selected in the current diagram |

Tip: For more information on using the Automation Interface visit www.sparxsystems.com.au/AutIntVB.htm.

4.5.7.1.4 Customize Keyboard

The **Keyboard** tab on the **Customize** window allows you to configure shortcuts used to access main menu options. This is convenient for creating additional shortcut keys, or for changing the current configuration to match your work habits or other applications.

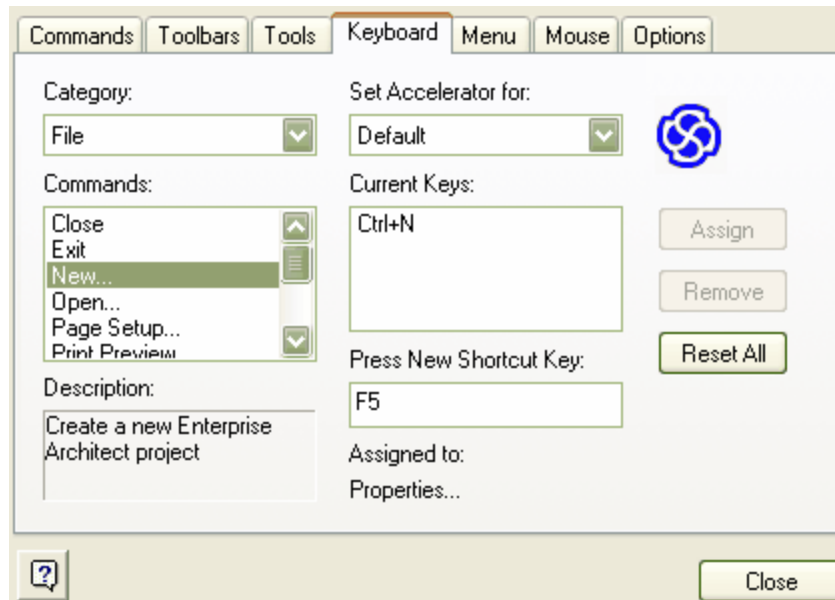


Modifying Keyboard Shortcuts

To modify a keyboard shortcut, follow these steps:

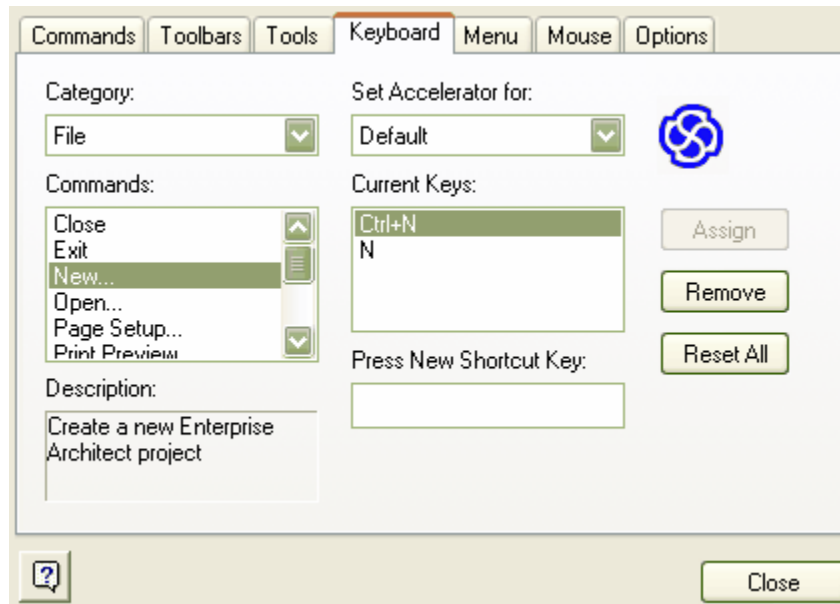
1. From the main menu select the **Tools | Customize** option to open the **Customize** dialog. Select the **Keyboard** tab.
2. Select the command you wish to modify: first select the menu it belongs to from the **Category** drop down list, then select the command from the **Commands** list. The current shortcut key for the selected command will appear in the **Current Keys** list.
3. Place the cursor in the **Press New Shortcut Key** textbox and press your desired shortcut key(s) for this command.

Note: If you want to use the **F5** key for example, press the actual **F5** key, don't type **F** then **5**.



Note: In the example above, the *Assign* option is disabled. This is because the *F5* key is already a shortcut to view diagram properties. If this occurs you will need to select a different shortcut key.

- Once you have selected an available shortcut, press *Assign* to apply the change. In the example below, the new shortcut is *N*.



- This has added a new shortcut so that both *N* and *Ctrl+N* create a new EA project. If you want *N* to be the only shortcut for this action, select *Ctrl+N* in the *Current Keys* list and press *Remove*.

Note: Remember that you can always revert to the default shortcut keys by pressing *Reset All*.

Tip: Modified shortcut keys are stored in the registry, so will only affect the current user.

Warning: About Custom Layouts and Keyboard Shortcuts

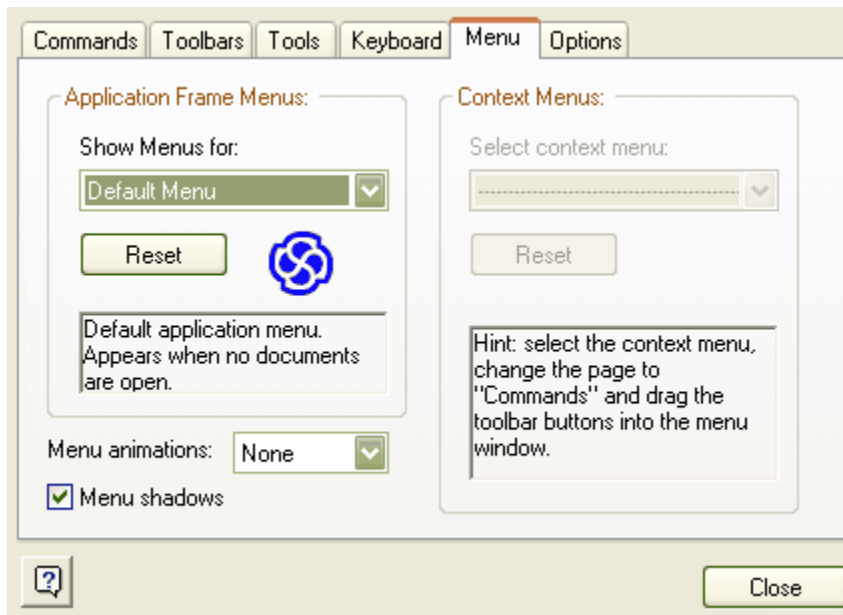
If you have set keyboard shortcuts, these will not be overridden if you switch to the Default Layout or the Iconix Layout option.

However, if you have set keyboard shortcuts and you switch to a User Layout, your keyboard shortcuts will be overridden, unless you have saved them as part of the User Layout you have switched to.

For more information about custom layouts, see the [Custom Layouts](#) topic.

4.5.7.1.5 Customize Menu

The *Menu* tab on the *Customize* window allows you to customize the appearance of your menus.



Application Frame Menus

Currently the *Show Menus For* feature is disabled as EA is not an MDI application.

Context Menus

Currently this feature is disabled.

Menu Animations

The following menu animations can be selected from the *Menu animations* drop down menu:

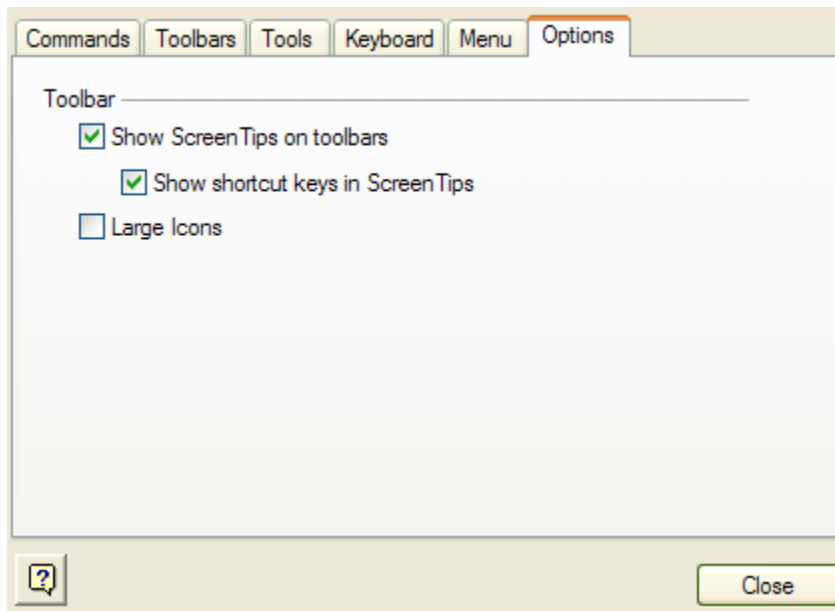
- None
- Unfold
- Slide
- Fade

Menu Shadows

Menu shadows can be toggled on or off by checking or clearing the *Menu shadows* check box.

4.5.7.1.6 Customize Options

The *Options* tab on the *Customize* window allows you to customize the appearance of toolbar items.

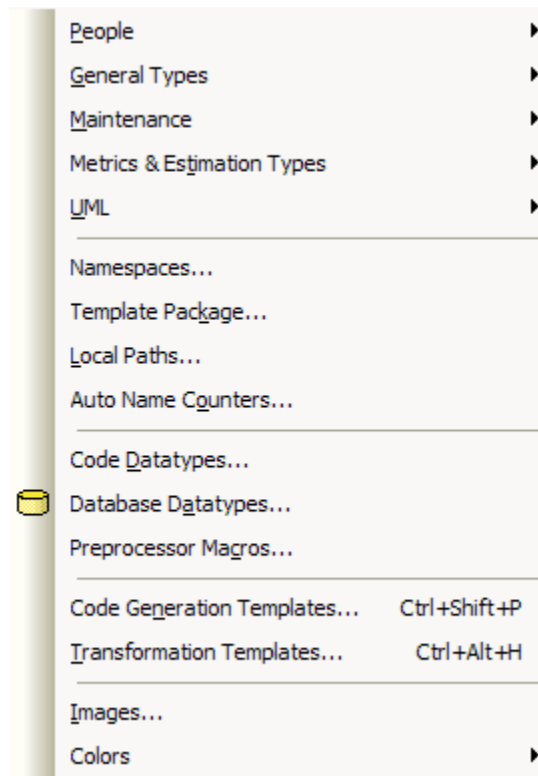


You can toggle the following options by checking or clearing the check boxes:

- Show screen tips on toolbars.
- Show shortcut keys in-screen tips.
- large icons.

4.5.8 The Settings Menu

The **Settings Menu** allows you to configure various settings for your overall project. Configure the resources involved, general types, maintenance types, metrics and estimation types, stereotypes, tagged values, cardinality values, datatypes, language macros, local directories, image management, CSV import and export specifications, and reference data export/import.



| Element | Description |
|-------------------------------|--|
| People | See below. |
| General Types | See below. |
| Maintenance | See below. |
| Metrics & Estimation Types | See below. |
| UML | See below. |
| Model Namespaces | Locate and delete model namespaces. |
| Template Package | Configure or change the default template directory. |
| Local Paths | Configure local directories and paths. |
| Auto Counters | Configure automatic naming for elements. |
| Code Datatypes | Add, modify and delete programming languages datatypes . |
| Database Datatypes | Add, modify and delete database datatypes . |
| Preprocessor Macros | Add and delete preprocessor macros . |
| Code Generation Templates | Modify code generation templates using the Code Templates Editor . [<i>Ctrl+Shift+P</i>] |
| Transformation Templates | Modify transformation templates using the Transformation Templates Editor . |
| Images | Open Image Manager . |
| Colors | See below. |

The People Sub-Menu

Configure the authors, clients, resources and roles for your project.

| Element | Description |
|-------------------------------|---|
| Model Authors | Add, modify and delete project author(s). |
| Clients | Add, modify and delete project client(s). |

| | |
|---------------------------|---|
| Resources | Add, modify and delete project resource(s). |
| Roles | Add, modify and delete project role(s). |

The General Types Sub-Menu

Configure requirements, status types, constraints and scenarios for your project.

| Element | Description |
|--------------------------|--|
| Requirements | Add, modify and delete requirement types . |
| Status Types | Add, modify and delete status types . |
| Constraints Status Types | Add, modify and delete constraint status types. |
| Constraints | Add, modify and delete constraint types . |
| Scenarios | Add, modify and delete scenario types . |

The Maintenance Sub-Menu

Keep track of problem types and test types.

| Element | Description |
|----------|--|
| Problems | Add, modify and delete problem types . |
| Testing | Add, modify and delete test types . |

The Metrics and Estimation Types Sub-Menu

Configure metrics and estimation types for your project.

| Element | Description |
|-----------------------------------|--|
| Risks | Set up the risk types to appear in the project dialog. |
| Metrics | Create and edit metric types . |
| Effort | Create and edit effort types . |
| TCF Values | Configure Technical Complexity Factors. |
| ECF Values | Configure Environment Complexity Factors. |
| Default Hour Rate | Set the Default hour rate for estimation. |

The UML Sub-Menu

Configure stereotypes, tagged values and the cardinality list for your project.

| Element | Description |
|-------------------------------|--|
| Stereotypes | Add, modify and delete stereotypes. |
| Tagged Values | Add, modify and delete tagged values. |
| Cardinality | Add, modify and delete values in the default cardinality list. |

The Color Sub-Menu

Configure the custom colors for the project.

| Element | Description |
|---------------------------|---|
| Get Project Custom Colors | Get the custom colors used in the Appearance dialog . |
| Set Project Custom Colors | Set the custom colors used in the Appearance dialog . |

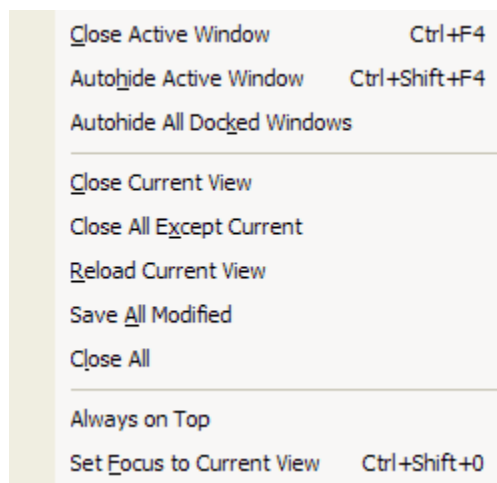
See Also

- [The Main Menu](#)
- [The File Menu](#)

- [The Edit Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.5.9 The Window Menu

The *Window Menu* provides access to various actions related to the configuring of open windows.



| Element | Description |
|-----------------------------|---|
| Close Active Window | Close the window which currently has focus. [<i>Ctrl+F4</i>] |
| Autohide Active Window | Autohide the window which currently has focus. [<i>Ctrl+Shift+F4</i>] |
| Autohide All Docked Windows | Autohide all windows that are docked. |
| Close Current View | Close Current, closes the current view |
| Close All Except Current | Close All Except Current, closes all except the currently selected view. |
| Reload Current View | Reload Current, refresh the current view. |
| Save All Modified | Save All Modified |
| Close All | Close all opened windows in the main tab view. |
| Always on Top | Forces the main EA window to be on top of all other window. |
| Set Focus to Current View | [<i>Ctrl+Shift+0</i>] |

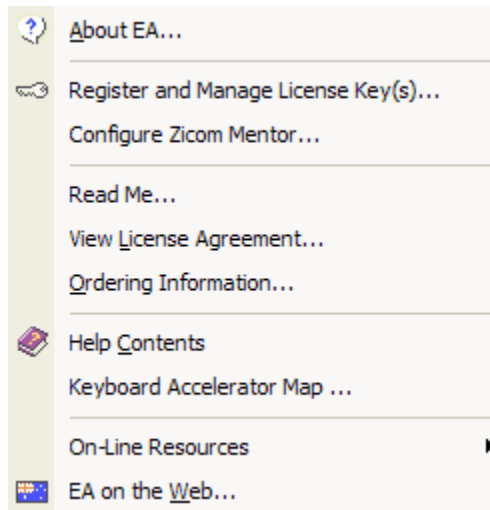
See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)

- [The Settings Menu](#)
- [The Help Menu](#)

4.5.10 The Help Menu

The *Help Menu* provides access to the EA help files, the Read Me, the EA License Agreement and various features on the [Sparx Systems website](#).



| Element | Description |
|---------------------------------------|--|
| About EA | View information about Enterprise Architect, including your registration details. |
| Register and Manage License Key(s)... | Allows the user to configure and manage the license keys used to register the name and keys for EA and its Add-Ins. For more information see the Licence Management section. |
| Configure Zicom Mentor... | Register a third party add-in for EA. For more information see the Register Zicom Mentor section. |
| Read Me | View the readme.txt file which details the changes and enhancements in EA, build by build. |
| View License Agreement | View the Enterprise Architect End User License Agreement. |
| Ordering Information | View information on how to purchase EA. |
| Help Contents | View the EA help files. |
| Keyboard Accelerator Map | View the keyboard accelerator map. You can customize your keyboard shortcuts if you wish. |
| On-Line Resources | See below. |
| EA on the Web | Visit the Sparx Systems website . |

The On-Line Resources Sub-Menu

Access help and resources on-line at the [Sparx Systems website](#).

| Element | Description |
|------------------------|---|
| User Forum and News | Visit the EA user discussion forum . |
| Request-a-Feature | Request a feature you would like to see in EA. |
| Bug Report Page | Report the details of a bug you have found in EA. |
| Latest Version Details | Find out the details of the latest EA build . |
| Automation Interface | Access the EA Automation Interface guide. |

| | |
|------------------------------|---|
| Introducing UML | Access the Sparx Systems online UML tutorials . |
| Pricing and Purchase Options | Purchase or upgrade EA over the internet. |

See Also

- [The Main Menu](#)
- [The File Menu](#)
- [The Edit Menu](#)
- [The View Menu](#)
- [The Project Menu](#)
- [The Diagram Menu](#)
- [The Element Menu](#)
- [The Tools Menu](#)
- [The Settings Menu](#)
- [The Window Menu](#)
- [The Help Menu](#)

4.6 View Options

Models in EA are viewed in ways in the application workspace - either in [Diagram View](#) or [Report View](#). The *Report View* has two modes, the Report View mode and the Search View mode.

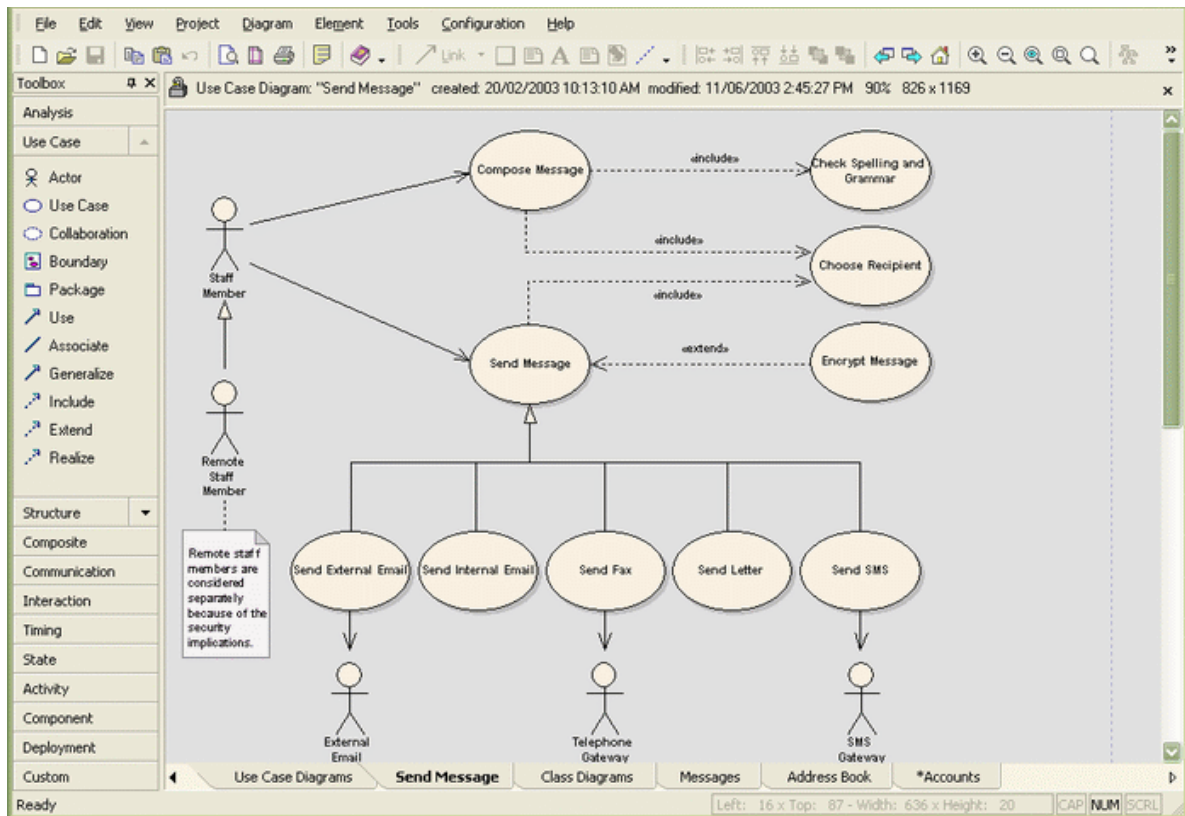
See Also

- [Diagram View](#)
- [Report View](#)
- [Search View](#)

4.6.1 Diagram View

The *Diagram View* is the main workspace window and displays the currently selected diagram. Only one diagram can be open at a time. This view is used to build the model relationships and elements. You may create new elements for the diagram, drag existing elements into the diagram and generally arrange and work with model elements.

The *Diagram View* is the main work area. Most work is carried out on elements in the diagram view, so understanding how it works and how to manipulate elements is essential. Use the example project supplied with EA to explore the capabilities and behavior of the diagram view.



Typical diagram activities include:

- Print and print preview diagrams
- Add new elements to the diagram using the UML Toolbox
- Add connectors between elements using connectors from the UML Toolbox
- Set diagram properties
- Save the diagram image to file
- Save the diagram image to the clipboard
- Copy elements in a diagram to link or copy elsewhere
- Zoom diagram to different magnifications
- Use the toolbar forward and back arrows to load previous and next diagrams
- Align and size multiple elements
- Delete elements from the diagram (but not the project)
- Double click with the left mouse button on the diagram background to open the diagram property dialog
- Right click on the diagram background to open the context menu

Tip: You can also use [Report View](#).

See Also

- [Report View](#)
- [Search View](#)

4.6.2 Report View

The **Report View** is an Outlook style report list that can be displayed in the main workspace. When visible, it lists each element that corresponds with the package selected in the Project View and the main details associated with that element.

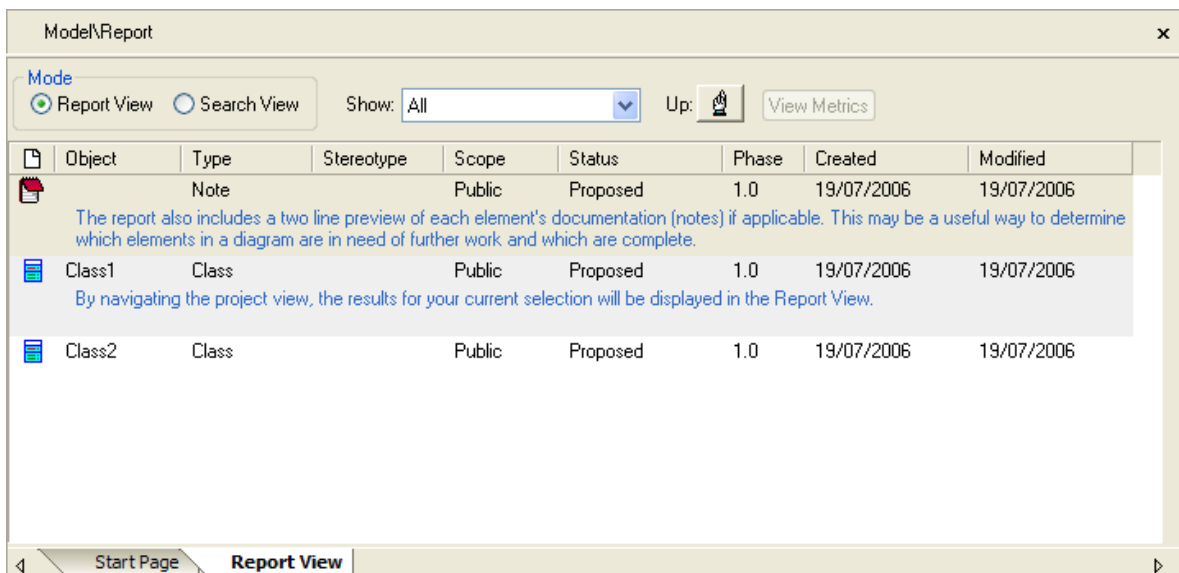
To access the report view, select *View | Report and Search View* from the main menu. Alternatively select a diagram or package in the *Project View* and press the *Report View* button on the [Default Tools toolbar](#).



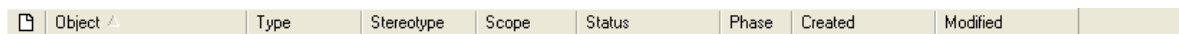
Note: By navigating the project view, the results for your current selection will be displayed in the Report View.

The Report View may be sorted by any column (ascending or descending) by clicking in the column header. Columns may be moved left or right, however the order you place them in will not be retained in subsequent sessions. It is possible to navigate down the package hierarchy by clicking on package icons in the Report View.

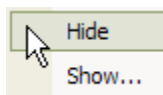
Tip: The report also includes a two line preview of each element's documentation (notes) if applicable. This may be a useful way to determine which elements in a diagram are in need of further work and which are complete.



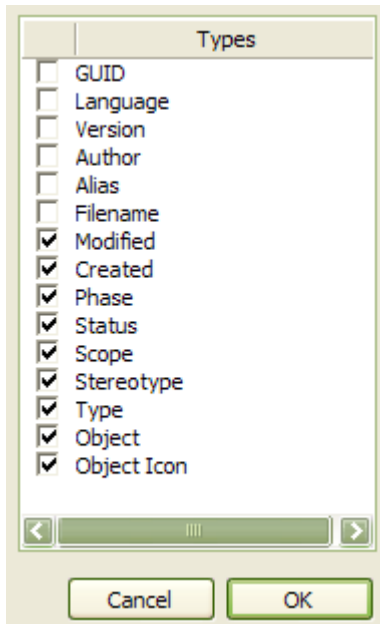
The View Toolbar



By right clicking on a field from the toolbar you can Hide or Show that field. Fields can also be dragged and dropped into position.



By selecting *Show*, the following dialog will appear. Simply check off the fields you wish to be displayed in the toolbar.



The View Controls

The following table contains a description of the view controls.

| Mode | Description |
|-------------|--|
| Report View | This mode enables you to navigate the project view and your selection results will be displayed in the Report View window. |
| Search View | This mode enables you to run pre-defined or a user defined search. See Searching Report View |

The Show drop down menu is visible in the Report View mode. It is used to filter the items displayed by a single element type.

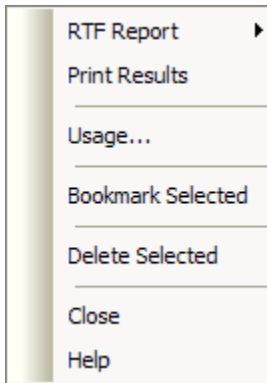
| Show | Description |
|-----------------|--|
| All | By selecting All in the Show drop down menu, all element types will be displayed. |
| Other Selection | You may also filter a particular element type. Only the type selected in the drop down menu will be displayed. |

| Control | Description |
|---------|---|
| Up | Move up one level. This corresponds to moving up one level in the Project View. |

Element Selection in the View

By clicking on any element in the view, you will navigate down a level. This corresponds to moving down one level in the Project View.

By right clicking on any element in the view, you can access the following context menu.



| Control | Description |
|-------------------|---|
| RTF Report | Generate an RTF report, refer to topic RTF Dialog Options |
| Print Results | Prints out the filtered results. |
| Usage | Take you to the diagram that uses the element or if the element is used in multiple diagrams, presents a list of diagrams to choose from. |
| Clear Results | Clears the view. |
| Bookmark Selected | Bookmark the element. |
| Delete Selected | Allows you to delete the element. |
| Close | Close the Report or Search view. |
| Help | A link to the help topic. |

See Also

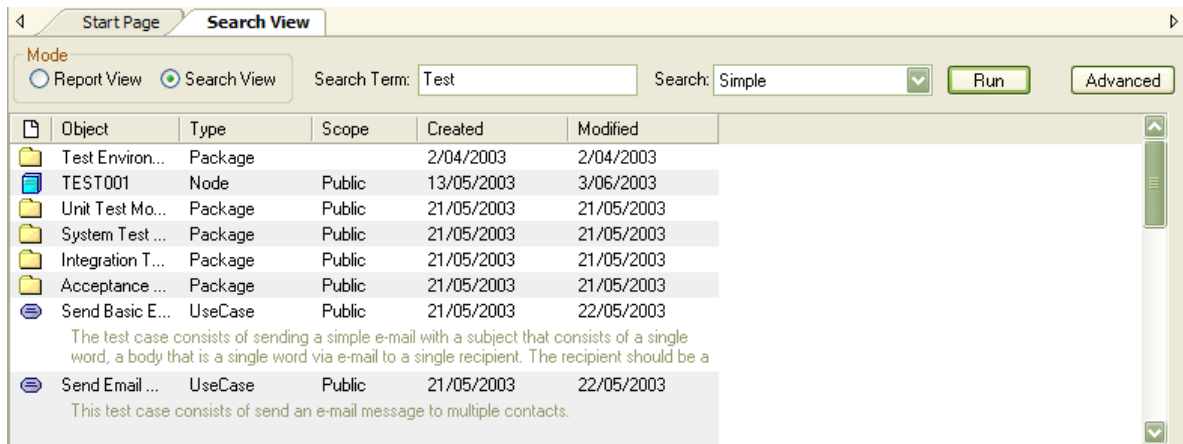
- [Searching a Project](#)
- [Searching the Search View](#)
- [Search Definitions](#)
- [Adding Filters](#)
- [Fields and Conditions](#)
- [Search View](#)

4.6.3 Search View

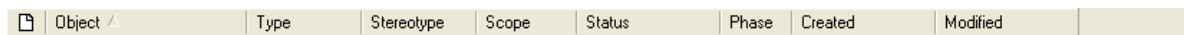
To access the search view, select *View | Report and Search View* from the main menu. Alternatively select a diagram or package in the *Project View* and press the *Report View* button on the [Default Tools toolbar](#).



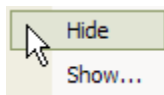
Select the *Search View* radio button under the *Mode* group. For more information on searching the *Search View* refer to help topic [Searching the Search View](#).



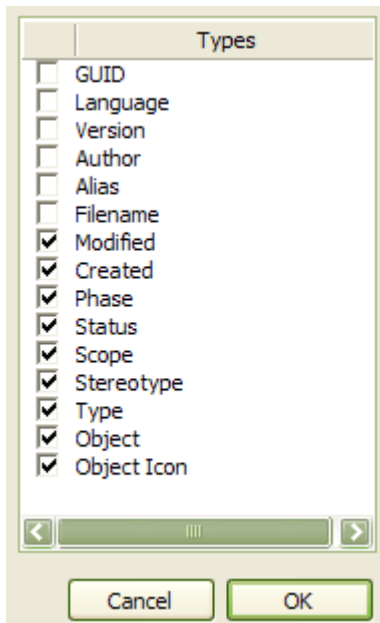
The View Toolbar



By right clicking on a field from the toolbar you can Hide or Show that field. Fields can also be dragged and dropped into position.



By selecting **Show**, the following dialog will appear. Simply check off the fields you wish to be displayed in the toolbar.



The View Controls

The following table contains a description of the view controls.

| Mode | Description |
|-------------|--|
| Report View | This mode enables you to navigate the project view and your selection results will be displayed in the Report View window. |
| Search View | This mode enables you to run pre-defined or a user defined search. See Searching Report View |

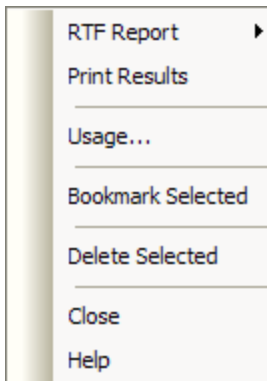
The Show drop down menu is visible in the Report View mode. It is used to filter the items displayed by a single element type.

| Control | Description |
|-------------|--|
| Search Term | The search term is the term you wish to search for. |
| Search | A selection of pre-defined and user defined search definitions. The default being the simple search. |
| Run | Run the selected search. |
| Advanced | The Advanced button enables you to create Search Definitions . |

Element Selection in the View

By clicking on any element in the view, you will navigate down a level. This corresponds to moving down one level in the Project View.

By right clicking on any element in the view, you can access the following context menu.



| Control | Description |
|-------------------|---|
| RTF Report | Generate an RTF report, refer to topic RTF Dialog Options |
| Print Results | Prints out the filtered results. |
| Usage | Take you to the diagram that uses the element or if the element is used in multiple diagrams, presents a list of diagrams to choose from. |
| Clear Results | Clears the view. |
| Bookmark Selected | Bookmark the element. |
| Delete Selected | Allows you to delete the element. |
| Close | Close the Report or Search view. |
| Help | A link to the help topic. |

See Also

- [Searching a Project](#)
- [Searching the Search View](#)
- [Search Definitions](#)
- [Pre-defined Search Definitions](#)

- [Adding Filters](#)
- [Fields and Conditions](#)
- [Report View](#)

4.7 Searching a Project

In EA, it is possible to search elements for a particular phrase or words over an entire project. To open the *Model Search* dialog, select *Edit | Find* from the main menu. This will bring up the search dialog. The search dialog allows the user to enter a search term and has the search parameters selected from a defined search filter, the default being a simple search. The search filter can be one of the default filters or can be defined by the user. For more details on defining a search see [Search Definitions](#). Search results are displayed in the [Report View](#).

| Model Search | Functionality |
|--------------|--|
| Search Term | Type the search term into this field. |
| Search | You can define your own custom search filters in EA. They will appear within the drop down list. |
| Advanced | The Advanced button enables you to create Search Definitions . |
| Run Search | Run the search. |
| Close | Close the Model Search dialog. |

See Also

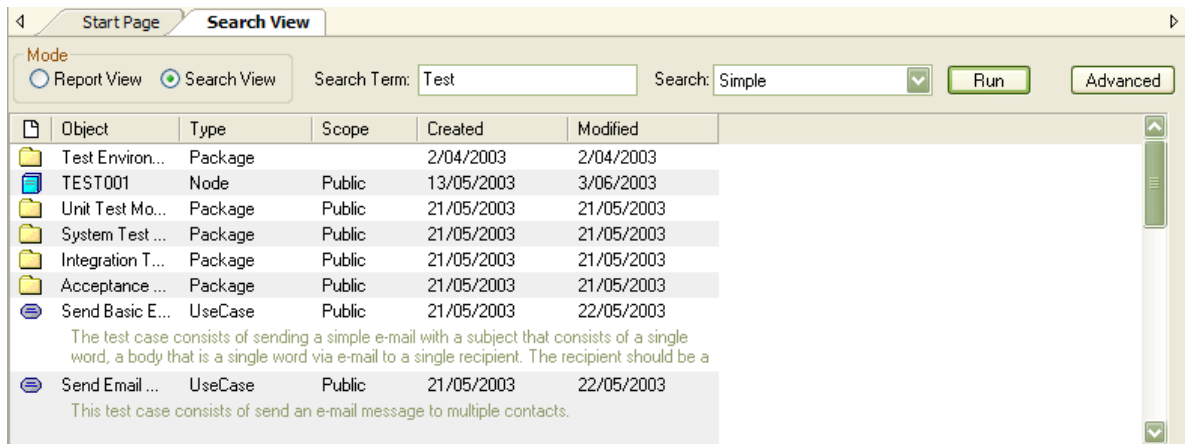
- [Searching the Search View](#)
- [Search Definitions](#)
- [Creating Search Definitions](#)
- [Adding Filters](#)
- [Fields and Conditions](#)
- [Report View](#)
- [Search View](#)

4.7.1 Searching the Search View

In order to search the [Search View](#), select the *Search View* radio button. The Search Term and Search drop down list perform the same function as in the [Model Search](#) dialog. Type in a search term and select a defined search type, the default being *Simple*. By pressing the *Run* button, your results will be displayed.

Note: Navigating the project view has no effect here, you need to press the *Run* button.

The Advanced button enables you to create [Search Definitions](#).

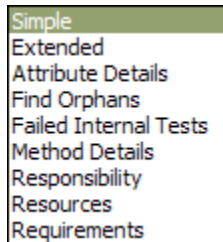


See Also

- [Searching a Project](#)
- [Search Definitions](#)
- [Creating Search Definitions](#)
- [Adding Filters](#)
- [Fields and Conditions](#)
- [Report View](#)
- [Search View](#)

4.7.2 Search Definitions

Search Filters allow the user to perform customized searches on terms in order to locate model elements. There are a number of [pre-defined search definitions](#) available in the *Search List* drop down. The default search is the *Simple* search.



The simple search shown below will search all elements only looking at the **Name** and **Notes** fields. If the search term is found in the **Names** field **OR** the **Notes** field, then those elements will be displayed.

IMPORTANT NOTE: The fields that are listed below take on an **OR** relationship when **no Required** check box's are ticked. i.e. If the search term is found in any one of those fields, then the element will be displayed.

The **Simple** configuration below **OR's** the Name and Notes field.

Search Term: Search List:

Search On

| Search In | Condition | Look For | Required |
|---|-----------|---------------|--------------------------|
| <input type="checkbox"/> Element | | | |
| <input checked="" type="checkbox"/> Name | Contains | <Search Term> | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> Notes | Contains | <Search Term> | <input type="checkbox"/> |

Element Features
 Optional Required

Search In:

Take a look at the following configuration below, the fields, **Name** and **Notes**, both have the **Required** check box's ticked. The two fields now take an **AND** relationship. This configuration will only display elements that contain our search term in both fields. i.e The search is required to be found in both the **Name** and **Notes** Fields.

The **Simple** configuration below **AND's** the Name and Notes field.

Search Term: Search List:

Search On

| Search In | Condition | Look For | Required |
|---|-----------|---------------|-------------------------------------|
| <input type="checkbox"/> Element | | | |
| <input checked="" type="checkbox"/> Name | Contains | <Search Term> | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> Notes | Contains | <Search Term> | <input checked="" type="checkbox"/> |

Element Features
 Optional Required

Search In:

NOTE: Any field having the **Required** check box ticked will over ride fields with the **Required** check box not ticked.

The following configuration will find elements that must have the search term in the **Name** **AND** may or may not have the search term in the **Notes**.

| Element | Description |
|--------------------------|--|
| Search Term | Input the term to search for into this field. |
| Search List | Select previously defined search definitions. |
| Run Search | Runs the selected search. |
| New Search | Create a new search, allowing the user to define a search. see Creating a Search Definition |
| Save Search | Saves a modified or new search |
| Copy Search | Copy an existing search. |
| Delete Search | Delete the search from the search list. |
| Help | Opens the Help topic related to searching |
| Close | Closes the Dialog. |
| Search On Element | |
| Search In | Search In displays the element search filters that are contained in a defined search. The format is the Element name, the conditions placed on the element, the search term on the element condition and whether the filter element is required. |
| <u>Condition</u> | The condition of the search parameter, the available options are contains, equal to, not equal to and one of. Contains Equal To Not Equals One Of.. |

| | |
|---|---|
| Look for | The search term to perform the conditional search, this option can be filled with value pertaining to the selected element type. For example, the value may be a date for DateCreated or may be a text value for other element types. The search term can be appended to by placing a value and a comma separator to allow for multiple search terms. |
| Required | Setting the Required checkbox for a particular field will give you a result set that must contain your search term in that field. |
| Element Features <ul style="list-style-type: none"> • Optional • Required | <p>Element Features appear as a new branch underneath the root Element Term in the Advanced Model Search Dialog. The <Extended> search is a good example - hit CTRL F to bring up the basic search dialog and choose the <Extended> search from the Search combo box. Then push the advanced button and the "Advanced Search dialog" should appear. If you scroll down the list box you will see sub branches like Attribute, Change, Custom Property. These are called element Features. You can Add these Features by pushing the Add Button. This will bring up the Add Filters Dialog with a list of all the filters you can choose for an element or element feature. Push the Search On Element combo box and you will see a list of the Element features you can search on. Each Feature has its own set of filters like Name, Notes and Alias, which you can add to your search. If you wanted to search on an Element Attributes Name, you would need to add The Attribute Feature with a Name filter to your search.</p> <p>The optional radio button inside the Element features group box, allows you to get a list of Elements that will meet one of the Element filters (Element Type = Object), or One Of the Features filters (Attribute Name = Class1). For example if your search was Element Name = Class11, Attribute Name = m_Att1 or Scope = Public, if you had optional checked, you would get all the Elements that have a Name = Class11 and all the elements that have a Attribute Name = m_Att1, or scope equal to public.</p> <p>If you have the Required Radio button ticked inside the Element Features Group Box, the result set will contain elements that must have the element features you have added. For example, if your search was Element Name = Class, Attribute Name = m_Att1 or Scope = Public. You would get Elements that Must have a name = Class AND have an Attribute with a Name of m_att1 or a Scope of Public.</p> |
| Add | Add a new element to filter the search on. |
| Edit Filter | Edit Filter allows the user to change the search parameters, this opens the edit filters dialog. |
| Remove Filter | Removes the selected filter from the search. |
| Search In | Allows you to choose between Entire Model or Current Tree Selection which will run a search in the currently selected package in the tree. |

See Also

- [Searching a Project](#)
- [Searching the Search View](#)
- [Creating Search Definitions](#)
- [Pre-defined Search Definitions](#)
- [Adding Filters](#)
- [Fields and Conditions](#)
- [Report View](#)
- [Search View](#)

4.7.2.1 Creating Search Definitions

Search definitions are created via the *Find in Project* dialog.

To access the *Find in Project* dialog:

- press the *Advanced* button from the *Search View*, or

- open the dialog from the *Main Menu*. Click on *Edit | Find in Project....* Then press the *Advanced* button.

To create a new search definition:

1. Press the *New Search* button.

2. Enter a name for your new search into the *Search Name* field.
3. Select the type of your Search
 - The *Query Builder* provides an interface that allows you to design your own search.
 - The *SQL Editor* allows advanced users to directly write SQL Select statements.
 - The *Add-In Search* allows you to supply the name of your add-in and a method (eg. MyAddin.RunThisMethod). This method will be called whenever the search is run. This search can be exported and distributed as a part of your add-in. See [Add-in Search](#) for more information.
4. Click *OK*

Query Builder

Your search definition will now appear as being selected in the *Search List* drop down. The main window will read 'There are no items to show in this view'. You can now press the *Add* button in order to [Add Filters](#).

SQL Editor

A special dialog, the *Custom SQL* dialog, will appear allowing you to input your Select statement. When done, click *Save* to save this search. The search will then be available from the *Search List*.

Add-In Search

Enter into the field the name of your add-in, a period (fullstop) and then the name of the method to be called (eg. MyAddin.RunThisMethod). Your search is automatically saved and available from the *Search List*. See [Add-in Search](#) for more information.

See Also

- [Searching a Project](#)
- [Searching the Search View](#)
- [Search Definitions](#)
- [Adding Filters](#)
- [Fields and Conditions](#)
- [Report View](#)
- [Search View](#)
- [Add-In Search](#)

4.7.2.2 Pre-defined Search Definitions

A number of pre-defined searches are provided with Enterprise Architect. These are described below.

Simple - Searches the Name and Notes fields of all Elements for the given search term.

Extended - Searches many additional fields relating to the Element including Attributes, Operations, Tagged Values, Test Cases and more.

Attribute Details - Search for Elements with Attributes relating to the search term. Search includes Tagged Values, Constraints, and common Attribute data fields.

Find Orphans - Search for orphaned Elements throughout the model, with ability to filter on common Element fields using a search term. An 'orphaned' Element is an Element that does not appear on any Diagram in the model.

Failed Internal Tests - Search for Elements containing internal Test Cases where search term is in any common Test Case field, and Status equal to 'Fail'.

Method Details - Search for Elements with Operations/Methods relating to the search term. Search includes Tagged Values, Constraints, and common Operation/Method data fields.

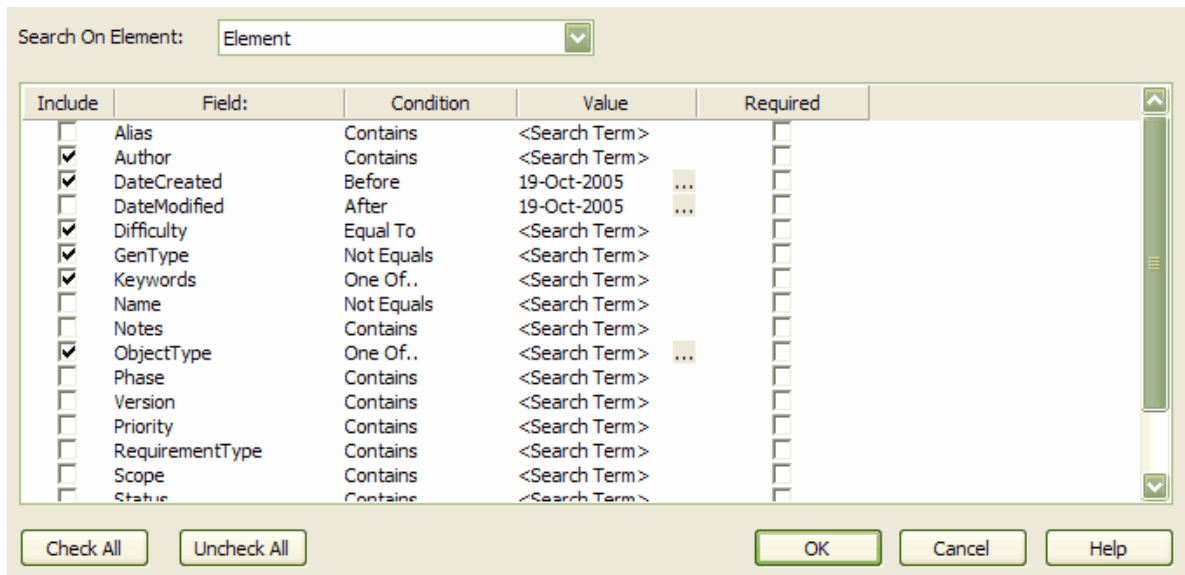
Responsibility - Search for elements with internal Responsibilities/Requirements where search term relates to any common Responsibility/Requirement field.

Resources - Search for elements with assigned Resources where search term relates to any common Resource field.

Requirements - Search for Requirement element types where search term relates to any common Element field.

4.7.2.3 Adding Filters

By pressing the **Add button** in the [Model Search](#) dialog, you will be confronted with the **Add Filters** dialog.



Select an item in the *Search On Element* drop down, a list of available fields to search will be shown. Just tick off the *Include* check box's for the items to search for. The *Required* box may or may not be ticked depending on the search. Press the *Ok* button in order to apply the filter. The fields selected will be added to the search definition. Multiple search definitions may be added as required. See [Fields and Conditions](#) for more info.

| Add Filters | Functionality |
|-------------------|---|
| Search On Element | By selecting from the pull down, you are able to build up search filters on any information about an element. The following is a list of what is available. <ul style="list-style-type: none"> Element Attribute Attribute.AttConstraint Attribute.AttTagValue Change Custom Property Method Method.MethodTagValue Method.Parameter Method.Parameter.ParamTagValue File Issue Scenario TagValue Task Test Responsibility Resource |
| Check All | Check all the items to include. |
| Uncheck All | Uncheck all the items to include. |
| Help | Opens the Help topic related to searching |
| Ok | Once you complete your selection, press Ok to add the selected fields to your search definition. |
| Cancel | Cancel out, don't add fields to your search definition. |

Field Items window

| | |
|---------|--|
| Include | Tick off the field items you wish to include in your search. |
|---------|--|

| | |
|---------------------------|--|
| Field | The name of the field to search. see Fields and Conditions |
| Condition | The condition of the search parameter, see Fields and Conditions |
| Value | This option can be filled with value pertaining to the selected element type. For example, the value may be a date for DateCreated or may be a text value for other element types. The search term can be appended to by placing a value and a comma separator to allow for multiple search terms. see Fields and Conditions |
| Required | Setting the Required checkbox for a particular field will give you a result set that must contain your search term in that field. |



See Also

- [Searching a Project](#)
- [Searching the Search View](#)
- [Search Definitions](#)
- [Creating Search Definitions](#)
- [Fields and Conditions](#)
- [Report View](#)
- [Search View](#)

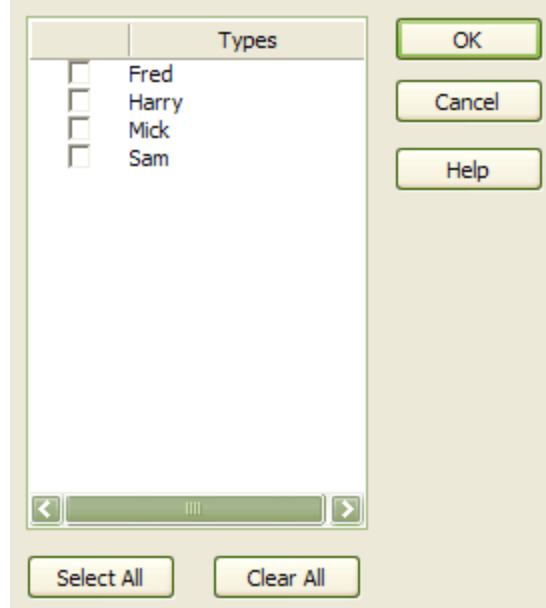
4.7.2.3.1 Fields and Conditions

By clicking on a condition for a particular field, a selection of conditions will be available. The following lists some possible selection conditions.

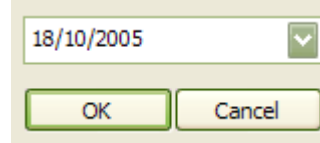
| | | |
|------------|------------|------------|
| Contains | After | Equal To |
| Equal To | Before | Not Equals |
| Not Equals | Equal To | |
| One Of.. | Not Equals | |

For some conditions, the value field will contain a . By clicking on , a selection dialog will appear. Examples of selection dialogs are shown below.

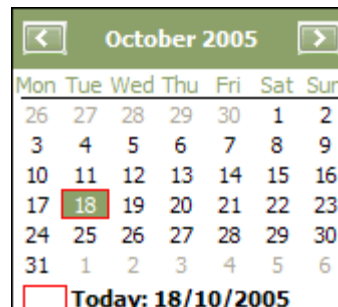
Example Selection Dialog for *One Of* section.



Date Selection Dialog for *Before* or *After* section.



Date selection from the drop down



See Also

- [Searching a Project](#)

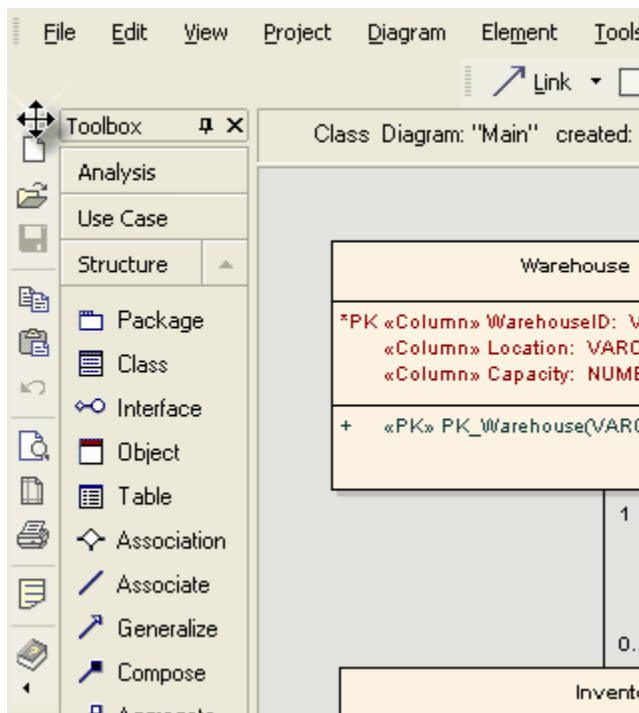
- [Searching the Search View](#)
- [Search Definitions](#)
- [Creating Search Definitions](#)
- [Adding Filters](#)
- [Report View](#)
- [Search View](#)

4.8 Workspace Toolbars

Enterprise Architect provides you with a selection of toolbars which may be dragged and docked within the application frame. These toolbars provide convenient shortcuts to common tasks. Toolbars may also be floated over the application by tearing them off the application toolbar section - this is useful when a certain set of functions is being used a lot in a particular area.

Toolbars may also be customized by deleting and reordering the default button set. See [Customize Toolbars](#) for more information. You can customize which toolbars are active by right clicking on the toolbar background and checking the required items in the context menu.

Note: You can dock toolbars to the edge of EA by dragging them by the title bar and placing them against the edge where you want them. The example below shows the Default Tools toolbar docked to the left side of the EA workspace:



See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)

- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.1 Default Tools Toolbar



The *Default Tools* toolbar provides quick access to the following functions (in order):

- New project [**Ctrl+N**]
- Open a project [**Ctrl+O**]
- Save current diagram
- Copy to clipboard [**Ctrl+Space**]
- Paste from clipboard [**Shift+Insert**]
- Undo last action
- Print Preview
- Page setup
- Print [**Ctrl+P**]
- Show report view of current diagram
- Help [**F1**]

You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.2 Project Toolbar



The *Project* toolbar provides quick access to the following functions (in order):

- Reload current project [**Ctrl+Shift+F11**]
- New diagram
- New package
- New Element [**Ctrl+M**]
- Search Project Browser
- Search entire project [**Ctrl+F**]
- New RTF document [**F8**]
- Project issues

- Project glossary
- Local options (preferences)

You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the [View | Toolbars](#) menu option.

See Also

- [Default Tools Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.3 Code Generation Toolbar

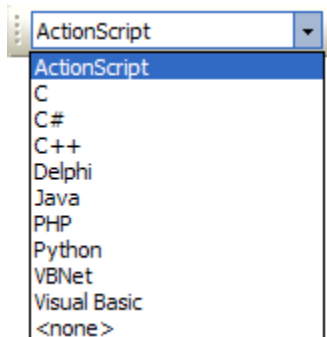


The [Code Generation](#) toolbar provides quick access to the following functions (in order):

- Set the Default Language
- Set the Default Database
- Import classes and interfaces from source files (see menu below)
- Generate code for a single selected class [[Ctrl+G](#)]
- Batch generate code for one or more selected classes [[Shift+F11](#)]
- Synchronize selected classes with source code [[Ctrl+R](#)]
- View code in default editor [[Ctrl+E](#)]

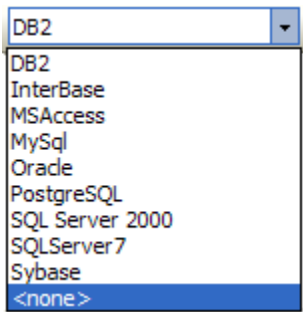
Set Default Code Language

To set the default language for the model use the Default Language dropdown box.



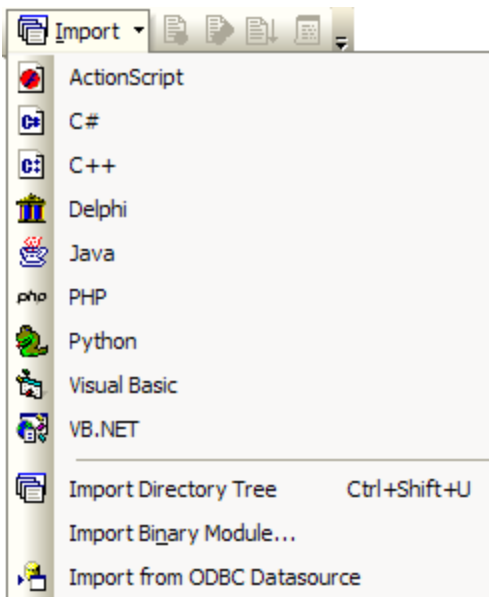
Set Default Database

To set the default database type for modeling use the Default Database dropdown box.



Import Code

If you press the down arrow associated with the *Import* button, you may select a language for code generation from the list below:



Tip: You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.4 UML Elements Toolbar

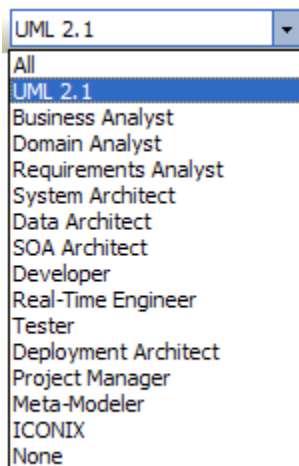


The *UML Elements* toolbar provides quick access to the following functions (in order):

- Select from a list the workspace perspectives
- Repeat last connector [**F3**]
- Access common relationships (see menu below)
- System boundary - swim lanes element
- Note
- Text
- Diagram note
- [Legend](#)
- Hyperlink
- Note link

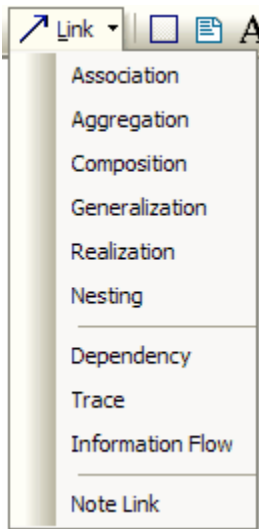
Select Workspace Perspective

By selecting an option from the workspace perspective dropdown list the user may alter the options available in the UML toolbox to suit a specific type of modeling perspective.



Selecting Link type

If you press the down arrow associated with the *Relationship* (arrow) button, you may select a relationship to add to the current diagram from the list below:



You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)
- [Legend](#)

4.8.5 Diagram Toolbar



The *Diagram* toolbar provides quick access to the following functions (in order):

- Align selected elements to the left [*Ctrl+Alt+Left*]
- Align selected elements to the right [*Ctrl+Alt+Right*]
- Align selected elements to the top [*Ctrl+Alt+Up*]
- Align selected elements to the bottom [*Ctrl+Alt+Down*]
- Bring selected element to top of Z order
- Move selected element to bottom of Z order
- Go to previous diagram [*Alt+Left*]
- Go to next diagram [*Alt+Right*]
- Go to default diagram
- Zoom In
- Zoom Out
- Zoom to fit diagram

- Zoom to fit page
- Zoom to 100%
- Auto layout diagram
- Show diagram properties [*F5*]
- Paste appearance
- Delete selected element(s) [*Ctrl+D*]

You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.6 Current Element Toolbar



The *Current Element* toolbar provides quick access to the following functions (in order):

- View and modify element properties [*Alt+Enter*]
- Set an element's parent or implement interfaces [*Ctrl+I*]
- View and modify Operations [*F10*]
- View and modify Attributes [*F9*]
- Specify the visibility of element features and compartments
- Specify the run state of an element [*Ctrl+Shift+R*]
- View usage of element in other diagrams etc. [*Ctrl+U*]
- Locate the element in the Project Browser [*Alt+G*]
- View the cross reference list for this element [*Ctrl+J*]
- Lock or unlock the current element
- Add a tagged value to the current element [*Ctrl+Shift+T*]

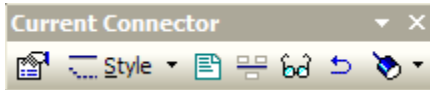
You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)

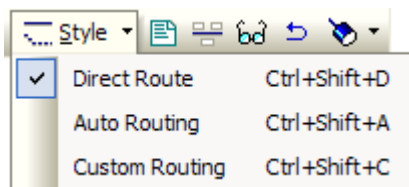
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.7 Current Connector Toolbar

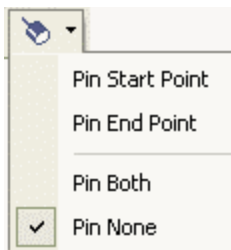


The *Current Connector* toolbar provides quick access to the following functions (in order):

- View and modify properties for the current connector
- Set the connector line style - see below:



- Attach a note to the currently selected connector
- Set the visibility for labels of the connector
- Set the visible or hidden relations in the current diagram [*Ctrl+Shift+I*]
- Reverse the direction of the currently selected connector
- Pin the start and/or connector ends to a position on the target element (drop menu) - see below:



You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.8 Format Toolbar



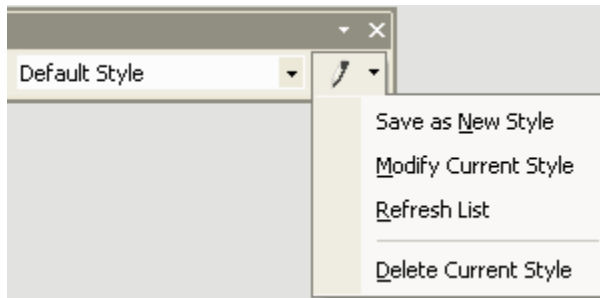
The **Format** toolbar is used to change the appearance of a specific element in the current diagram which does not effect any of the other elements of the same type throughout the model. The **Format** toolbar provides quick access to the following functions (in order):

- Fill Color (drop-down color palette)
- Text Color (drop-down color palette)
- Line Color (drop-down color palette)
- Line Width (drop-down option list)
- Apply Style to Element(s)
- Copy Style from Element
- Style list for selecting saved styles
- Save style (see below)

Note: Changes made with the **Format** toolbar will override any of the global changes made.

Note: To make a global change, select **Tools | Options** from the main menu. Select **Standard Colors** from the options tree.

If you press the down arrow associated with the **Save Style** button, you may select an option from the list below:



The **Fill Color** button can be used in conjunction with the Project Custom Colors menu items to allow users to have access to custom defined Project colors. To enable this feature go to **Tools | Options | Standard Colors** menu and ensure that [Get and Set Project Colors](#) the **Show Project Custom Colors in Element Format** option is checked. To define a set of custom colors see the [section](#).

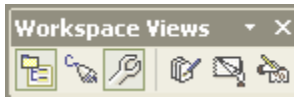
You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the **View | Toolbars** menu option.

See Also

- [Configure an Element's Default Appearance](#)
- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)

- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.9 Workspace Views



The *Workspace Views* toolbar provides a convenient means of turning on or off any of the following docked workspace windows:

- The Project Browser [*Alt+0*]
- The Properties windows [*Alt+1*]
- The UML Toolbox [*Alt+5*]
- The Glossary and System lists [*Alt+2*]
- The Maintenance lists [*Alt+4*]
- The Testing window [*Alt+3*]

Click on any of these buttons to toggle the associated window on/off.

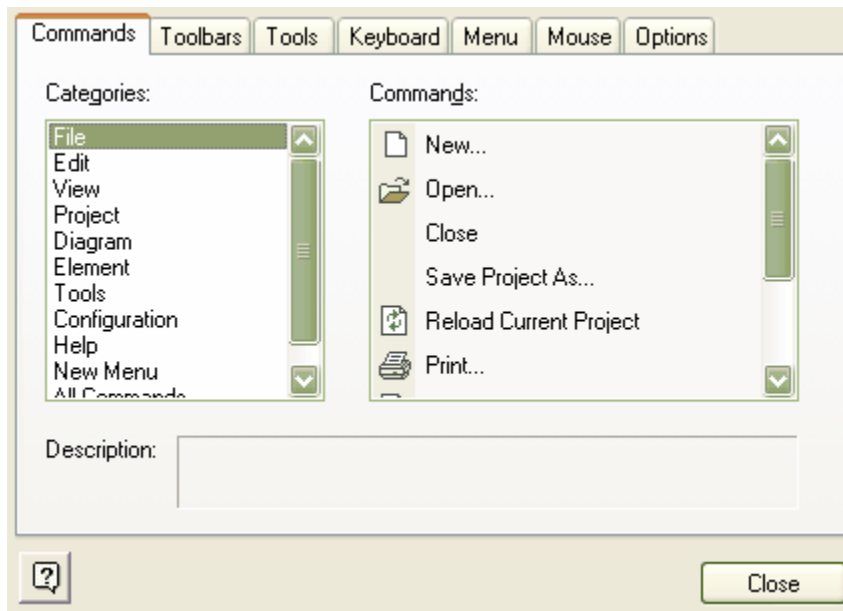
You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.10 Customize Toolbars

Using the *Customize* window, you can modify the content of toolbars by hiding buttons and by adding existing commands from within EA to a toolbar. You can also add one or more new toolbars and configure them exactly as required.



The Customize window allows you to customize the following.

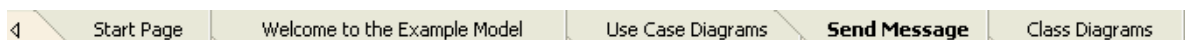
- [Commands](#)
- [Toolbars](#)
- [Tools](#)
- [Keyboard](#)
- [Menu](#)
- [Options](#)

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.11 Diagram Tabs

Diagram Tabs are located at either the bottom or the top of the diagram area, above the status bar. Each time you open a diagram, it is listed in the tab for easy access. The default location is at the bottom of the diagram area - for details explaining the process to place the diagram tabs at the top, refer to [Configure Local Options | General section](#).

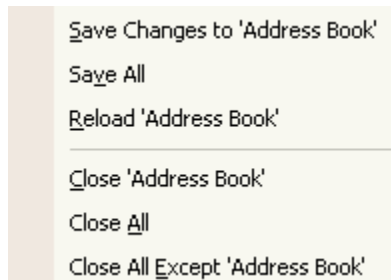


Notice that the Send Message tab is bold - this means that the current diagram is the Send Message diagram.

Also notice that the Address Book tab has an asterisk. This means that there are unsaved changes in the Address Book diagram. To save the changes see below.

The Diagram Tabs Menu

To access the diagrams tab menu, right click on any of the tabs. In the example below, the Address Book tab was right clicked.



The table below explains each menu option - note that 'Address Book' will be substituted in the menu depending on the name of the diagram clicked.

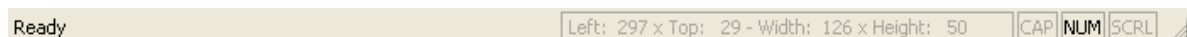
| Menu Item | Description |
|---------------------------------|--|
| Save Changes to 'Address Book' | Save the unsaved changes made to 'Address Book'. |
| Save All | Save the model. |
| Reload 'Address Book' | Reopen 'Address Book' without the unsaved changes - revert back to the state before any changes were made. |
| Close 'Address Book' | Close 'Address Book' - you will be prompted to save changes. |
| Close All | Close all open diagrams - you will be prompted to save any diagrams with unsaved changes. |
| Close All Except 'Address Book' | Close all diagrams except for 'Address Book' - you will be prompted to save any diagrams with unsaved changes. |

See Also

- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.8.12 Status Bar

The *Status* bar is located at the bottom of the Enterprise Architect workspace and provides feedback on current operations, menu hints and other status information.



In particular it lists the currently selected element, the state of the *CAP*, *NUM* and *SCRL* keys as well as showing the current coordinates of the selected element.

You may hide or show this toolbar from the *View | Toolbars* menu option. The Status bar cannot be docked in another position.

See Also

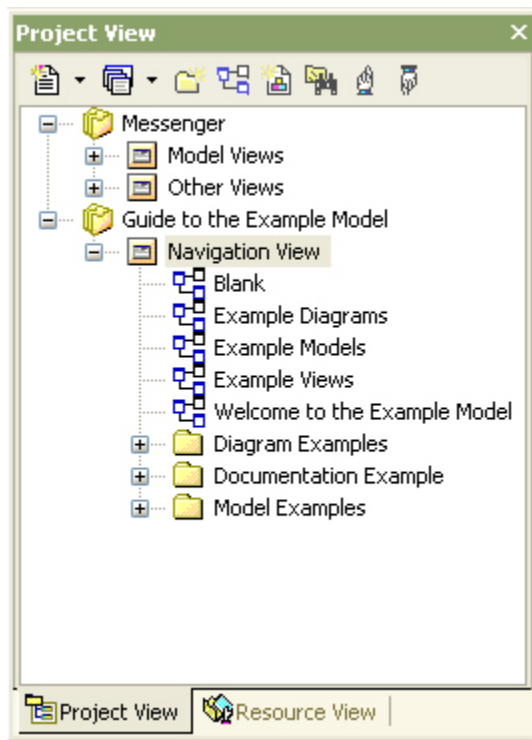
- [Default Tools Toolbar](#)
- [Project Toolbar](#)
- [Code Generation Toolbar](#)
- [UML Elements Toolbar](#)
- [Diagram Toolbar](#)
- [Current Element Toolbar](#)
- [Current Connector Toolbar](#)
- [Format Toolbar](#)
- [Workspace Views](#)
- [Customize Toolbars](#)
- [Diagram Tabs](#)
- [Status Bar](#)

4.9 The Project Browser

The Project Browser allows you to navigate through the Enterprise Architect project space. It displays *Packages*, *Diagrams*, *Elements* and element properties.

You can drag and drop elements between folders, or even drop elements from the *Project View* directly into the current diagram.

If you right click with the mouse on an item in the *Project View*, you may perform additional actions, such as adding new packages, creating diagrams, renaming items, creating documentation and other reports, and deleting model elements. It is also possible to edit the name of any item in the *Project View* by selecting an item and pressing the [*F2*] shortcut key.



Tip: The Project Browser is the main view of all model elements in your model - use the mouse to navigate the model.

Views

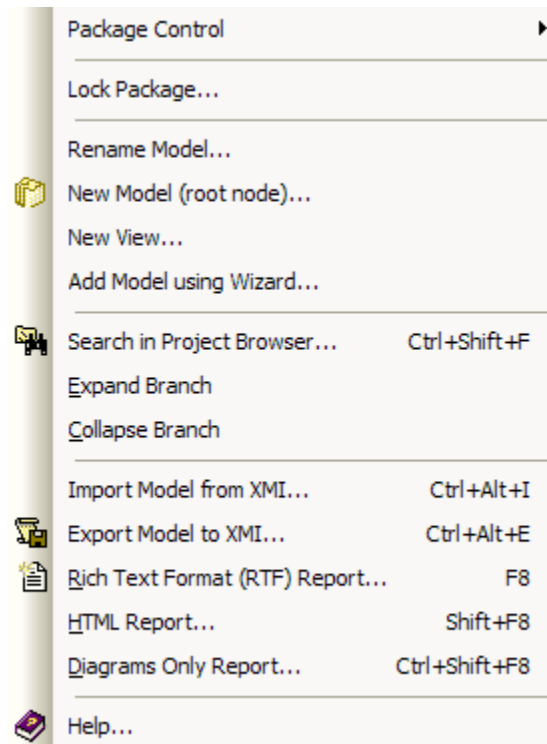
The **Project View** is divided into **Views**, each of which contains packages and other elements. The View hierarchy is described below:

| View | Description |
|-----------------|--|
| Views | The root view and base of your project model. |
| Use Case View | The functional and early analysis view. Contains business process and use case models. |
| Dynamic View | Contains state charts, activity and interaction diagrams. The dynamics of your system. |
| Logical View | The class model and domain model view. |
| Component View | A view for your system components. The high level view of what software will be built (executables, DLLs, components etc). |
| Deployment View | The physical model - what hardware will be deployed and what software will run on it. |
| Custom View | A work area for other views - eg. Formal requirements, recycle bin, interview notes, non-functional requirements, etc. |

Note: You can hide and show the Project Browser by pressing the Hide/Show Project Browser button on the toolbar, or use the Keyboard shortcut Alt+0.

4.9.1 Model Context Menu

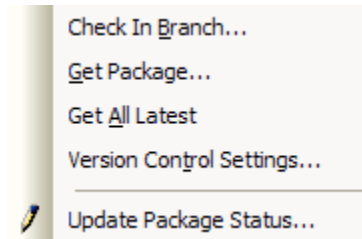
The **Root Node** in the Project Browser is the **Model** element. You may have more than one model element. The first level packages beneath the Model node are sometimes referred to as Views as they commonly divide a model into categories - Use Case Model, Logical Model etc.



| Menu Item | Description |
|--|--|
| Package Control Sub Menu | |
| Rename Project Root | Rename the current model. |
| New Project Root Node | Create a new project root (model). |
| New View | Create a new View (package). |
| Add Model from Pattern | Add additional models using the Model Wizard |
| Search Tree | Search the Project Browser. |
| Expand All | Expand all items. |
| Collapse All | Collapse all items. |
| Import Model from XMI | Import a model from an XMI file. |
| Export Model to XMI | Export a model to XMI. |
| Rich Text Documentation | Produce RTF documentation for the model. |
| HTML Documentation | Produce HTML documentation for the model. |
| Diagrams Only | Produce a diagrams only report (in RTF) for the model. |
| Help | Get additional help. |

4.9.1.1 Root Node Package Control Sub-Menu

The *Root Node Package Control* sub-menu is available from the *Root Package Node* context menu.

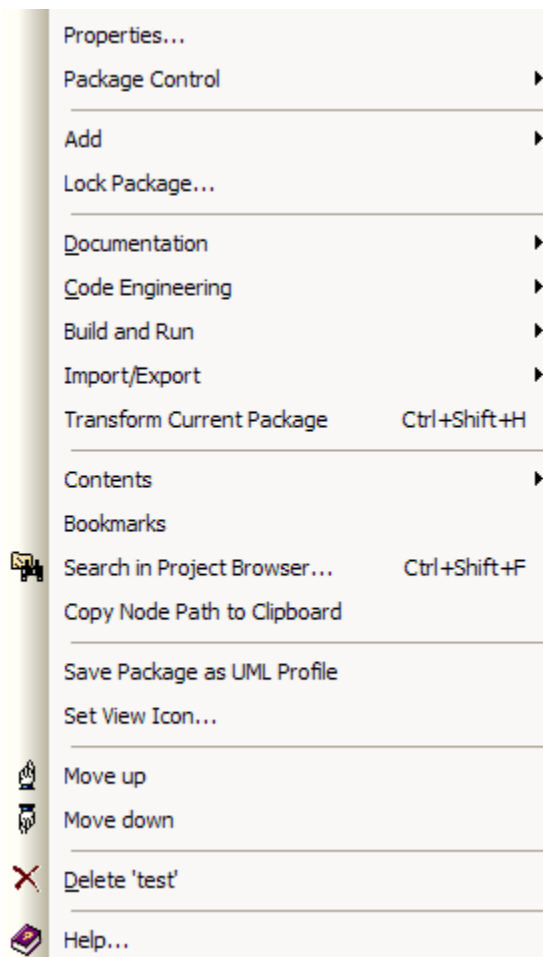


From this menu you can:

- Configure various settings for the package
- Retrieves the latest version of the package from the repository. Available only for packages which are checked in. Get Latest is not intended for sharing .EAP files and should only be used when people have their own individual databases.
- Displays the [Version Control Options dialog](#).
- Allows the user to provide a bulk update on the status of a package, this includes status options such as Proposed, Validate, Mandatory etc.
- Sets the namespace root for languages which support namespaces, for more information see the [Namespaces](#) section.

4.9.2 Package Context Menu

Right clicking on a *View* or *Package* element in the *Project View* opens the context menu below.

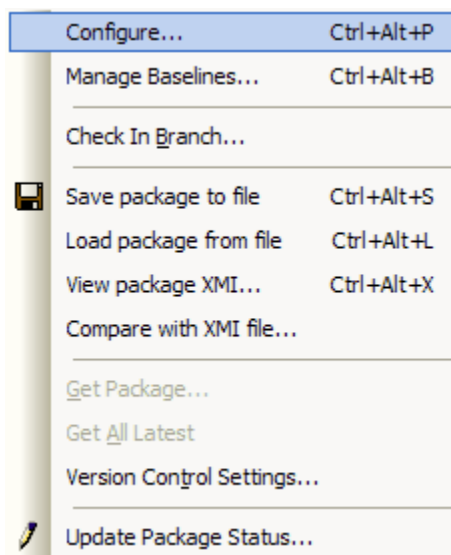


| Menu Item | Description |
|---|---|
| Properties | Add new packages to the model. |
| Set View Icon | Change the display icon for the selected package. |
| Package Control Sub Menu | See Also Version Control . |
| Add Sub Menu | Add a new diagram, element or another package to the current package. |
| Documentation Sub Menu | Produce a variety of documentation in RTF format. |
| Code Engineering Sub Menu | Perform Code Engineering functions |
| Build and Run Sub Menu | Build, run and debug functions. |
| Import/Export Sub Menu | Import and Export using XML text files. |
| Transform Current Package | [Ctrl+Shift+H] |
| Contents Sub Menu | |
| Bookmarks | |
| Search Tree | Search the View for specific elements. [Ctrl+Shift+F] |
| Copy Node Path to Clipboard | |
| Save Package as UML Profile | |

| | |
|----------------|---------------------------------|
| Move up | Move the View Up in the list. |
| Move down | Move the View Down in the list. |
| DeleteMyShapes | |
| Help | Get additional help. |

4.9.2.1 Package Control Sub-Menu

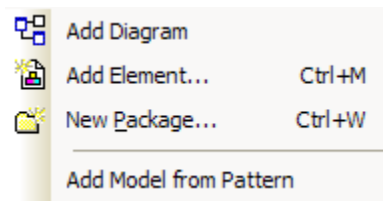
The *Package Control* sub-menu is available from the *Package* context menu.



| Menu Item |
|--|
| Configure various settings for the package. [Ctrl+Alt+P] |
| Manage Baselines. [Ctrl+Alt+B] |
| Save the package to file. [Ctrl+Alt+S] |
| Load a package from a file. [Ctrl+Alt+L] |
| View package XMI. [Ctrl+Alt+X] |
| Compare with XMI file. |
| Get package for version control |
| Get All Latest for version control |
| Version Control Options version control options |
| Update the package's status |

4.9.2.2 Add Sub-Menu

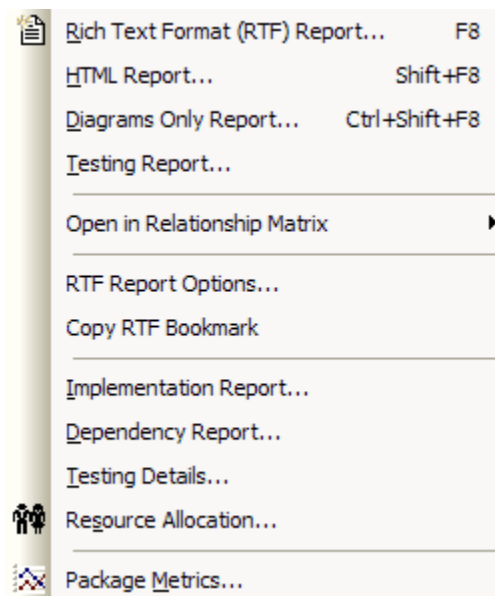
Add a new *Diagram*, *Element* or another *Package* to the current package.



| Menu Item |
|--|
| Add Diagram |
| Add Element [Ctrl+M] |
| New Package [Ctrl+W] |
| Add Model From Pattern |

4.9.2.3 Documentation Sub-Menu

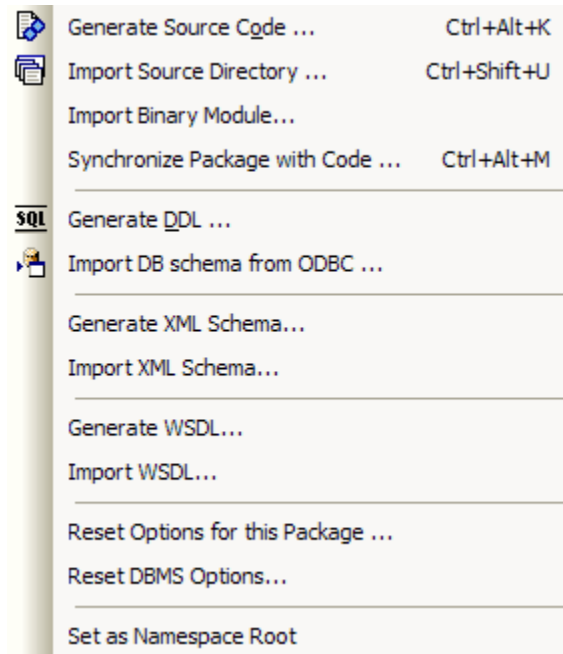
The *Documentation* sub-menu is available from the *Package* context menu.



| Menu Item |
|---|
| HTML documentation [Shift+F8] |
| Rich Text Documentation [F8] |
| Diagrams Only [Ctrl+Shift+F8] |
| Testing Documentation |
| Open in Relationship Matrix |
| RTF Documentation Options |
| Copy RTF Bookmarks to Clipboard |
| Implementation Report |
| Dependency Report |
| Testing Details |
| Resource Allocation |
| Package Metrics |

4.9.2.4 Code Engineering Sub-Menu

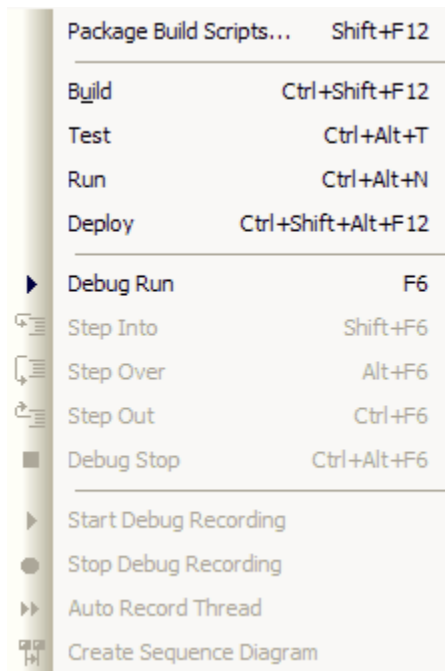
The *Code Engineering* sub-menu is available from the *Package* context menu.



| Menu Item |
|--|
| Generate source code [<i>Ctrl+Alt+K</i>] |
| Import Source Directory [<i>Ctrl+Shift+U</i>] |
| Import Binary Module |
| Synchronize Package Contents [<i>Ctrl+Alt+M</i>] |
| Generate DDL |
| Import DDL schema from ODBC |
| Generate XML Schema |
| Import XML Schema |
| Generate WSDL |
| Reset Options for this Package |
| Reset DBMS Options |
| Set as Namespace Root |

4.9.2.5 Build and Run Sub-Menu

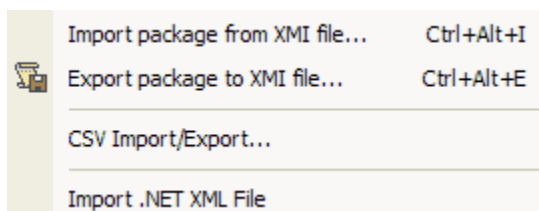
The *Build and Run* sub-menu is available from the *Package* context menu. This menu enables you to access the [Build and Run](#) options.



| Menu Item |
|---|
| Package Build Scripts |
| Build |
| Test |
| Run |
| Deploy |
| Debug Run |
| Step Into |
| Step Over |
| Step Out |
| Debug Stop |
| Start Debug Recording |
| Stop Debug Recording |
| Create Sequence Diagram |

4.9.2.6 Import/Export Sub-Menu

The *Import/Export* sub-menu is available from the *Package* context menu.

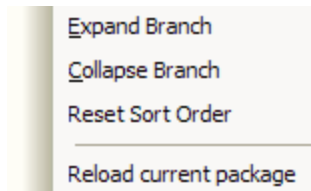


| Menu Item |
|---|
| Import a package from an XMI file |
| Export a package to an XMI file |

| |
|--|
| Perform a CSV import or export |
| Import a .NET XML File |

4.9.2.7 Contents Sub-Menu

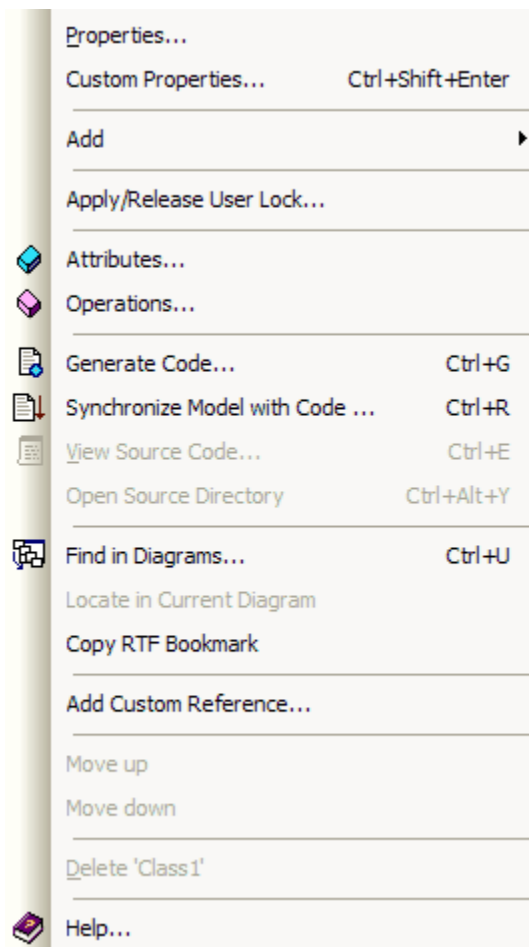
The Contents *sub-menu* is available from the *Package* context menu.



| Menu Item |
|--|
| Expand all of the items in the Project Browser |
| Collapse all of the items in the Project Browser |
| Reset the sort order |
| Reload current package |

4.9.3 Element Context Menu - Project Browser

Right clicking on an *Element* (class, object, activity, state, etc) in the Project Browser opens the element's context menu. The example below illustrates the options available from this menu



| Menu Item | |
|------------------------------|---|
| Properties | View and modify the element properties. |
| Custom Properties | |
| Add Sub-Menu | Create a diagram. |
| Attributes | Brings up the attribute dialog ready to create a new attribute. |
| Operations | Brings up the operations dialog ready to create a new operation |
| Generate Code | Generate the source code for this element. see Generate Source Code |
| Synchronize with Code | Synchronize the source code with the element in the diagram. See Reverse Engineer and Synchronizing |
| View Code | View the source code files. |
| Usage | Takes you to the diagram or displays a list of diagrams where the element is used. |
| Locate in Diagram | Selects the element in the current visible diagram. |
| Copy RTF Bookmark | Copy a bookmark in RTF format to the clipboard. |
| Linked Document | Creates a Linked Document (Corporate Edition Only) see Linked Documents |
| Add element as reference | |
| Move Up | Move the element up in the list of elements within this package. |
| Move Down | Move the element down in the list of elements within this package. |
| Delete (Element Name) | Delete the element. |
| Help | Get additional help. |

4.9.3.1 Add Sub Menu

The Add Sub-Menu

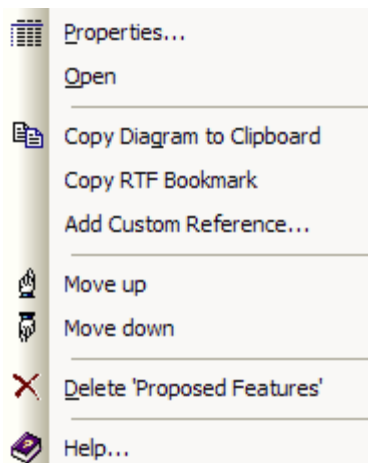
Depending on the kind of element you right click, you will see a different New Child Diagram sub-menu. Some elements do not show this context menu option.

From these menu you can create:

- A [Use Case diagram](#)
- An [Activity diagram](#)
- A [State Machine diagram](#)
- A [Timing diagram](#)
- A [Sequence diagram](#)
- An [Interaction Overview](#) diagram
- A [Communication diagram](#)
- An [Analysis diagram](#)
- A [Package diagram](#)
- A [Class diagram](#)
- An [Object diagram](#)
- A [Composite Structure diagram](#)
- A [Component diagram](#)
- A [Deployment diagram](#)
- A [Custom diagram](#)

4.9.4 Diagram Context Menu - Project Browser

Right click on a Diagram in the Project Browser to open the diagram context menu. The example below illustrates the functions available from this menu:



| Menu Item | |
|---------------------------|--|
| Properties... | View and modify the element properties |
| Open | Open the diagram in the View panel |
| Copy Diagram to Clipboard | Copy the diagram for pasting/copying to another location (see: Duplicate a Diagram) |
| Copy RTF Bookmark | Copy a bookmark in RTF format to the clipboard |
| Add diagram as reference | Add this diagram as a cross reference to other elements |
| Move UP | Move the diagram up in the list of diagrams within this package |
| Move down | Move the diagram down in the list of diagrams within this package |

| | |
|-----------------------|---------------------|
| Delete (Diagram Name) | Delete the diagram |
| Help | Get additional help |

4.9.5 Order Package Contents

EA allows you to change the order of elements in the *Project Browser*.

Elements will always be ordered first by their type, then set position, then alphabetically by default. Using the context menu option to Move Up or Move Down an element, it will be shifted within its type- but not outside of its type. This means you can order Packages or Diagrams or Use Cases but not mix elements up.

Ordering elements is very important when it comes to structuring your model - especially packages. Previous versions of EA only supported alphabetic ordering, but version 3.00 and greater support custom ordering. RTF documents will honor the custom ordering when printing documentation.

See Also

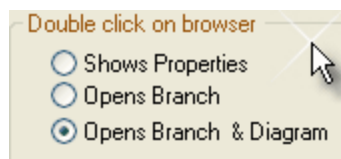
- [Setting the Default Project Browser Behavior.](#)

4.9.6 Setting Default Project Browser Behavior

There are several options for altering the look and behavior of the Project Browser.

Double Click Behavior

1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu. Navigate to the *General* page of the dialog.
2. In the Double click on browser area, you have the following options:



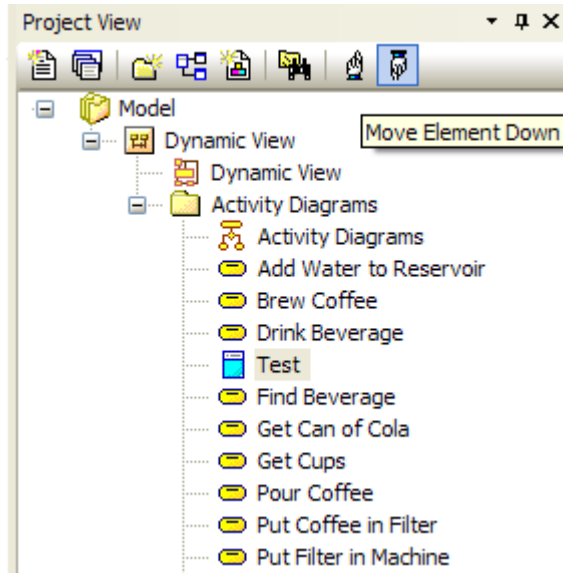
- *Shows Properties* - double clicking an item in the Project Browser will open a property dialog (if available) for that element.
- *Opens Branch* - double clicking an item in the Project Browser will expand the tree to show the item's children. If there are no children, nothing will happen.
- *Opens Branch & Diagram* - as above, plus also opens the first Diagram beneath the item, if applicable.

Allow Free Sorting

1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu. Navigate to the *General* page of the dialog.
2. Check the *Allow Free sorting* check box.

Free sorting of elements in the Project Browser, regardless of element type, is now enabled.

For example, the element Test (below) has been moved from its original position below the activity diagram, to amongst the activity elements. This can be done using the hand icons at the end of the Project Browser.



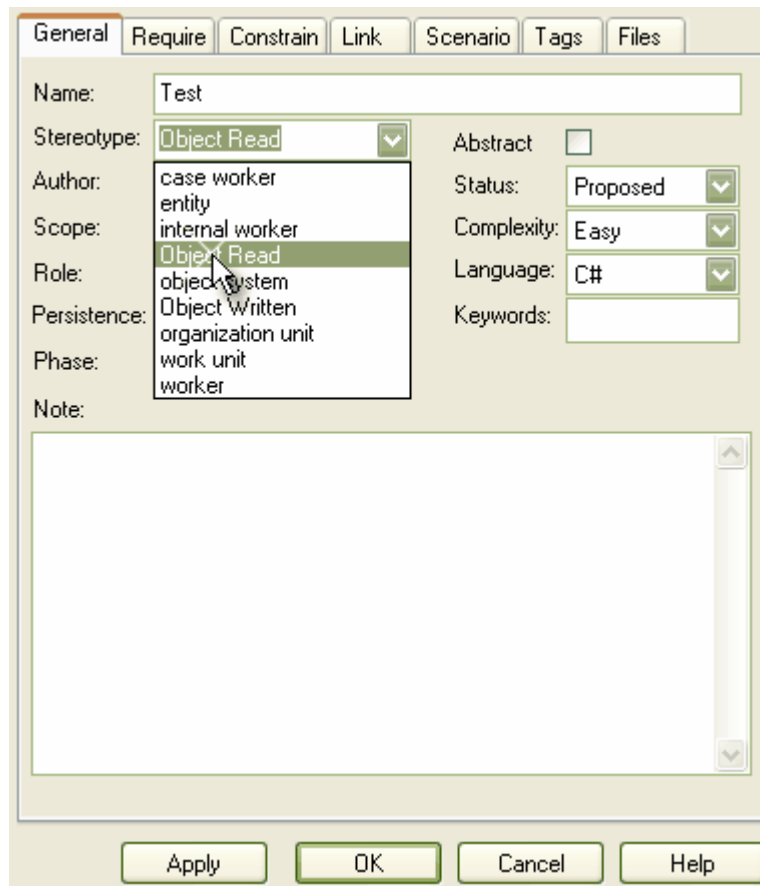
As the tool tip suggests - the *Move Down* icon moves the element further down the tree, and the *Move Up* icon moves the element further up the tree.

Show Stereotypes

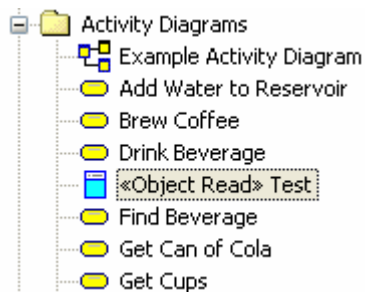
1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu. Navigate to the *General* page of the dialog.
2. Check the *Show Stereotypes* check box.
3. As prompted, shutdown and restart EA to enable this change to take effect.

Element and feature stereotypes will now be shown in the Project Browser.

For example, go to the properties window of an element such as the element Test, and set the stereotype.



The element's stereotype now shows in the Project Browser, as below.



4.10 Dockable Windows

There are several dockable tab windows available to use in Enterprise Architect. These can be accessed through the **View** menu and the **View | Other Windows** sub menu -OR- via the context menu accessed by right clicking on the main menu.

The dockable windows available include:

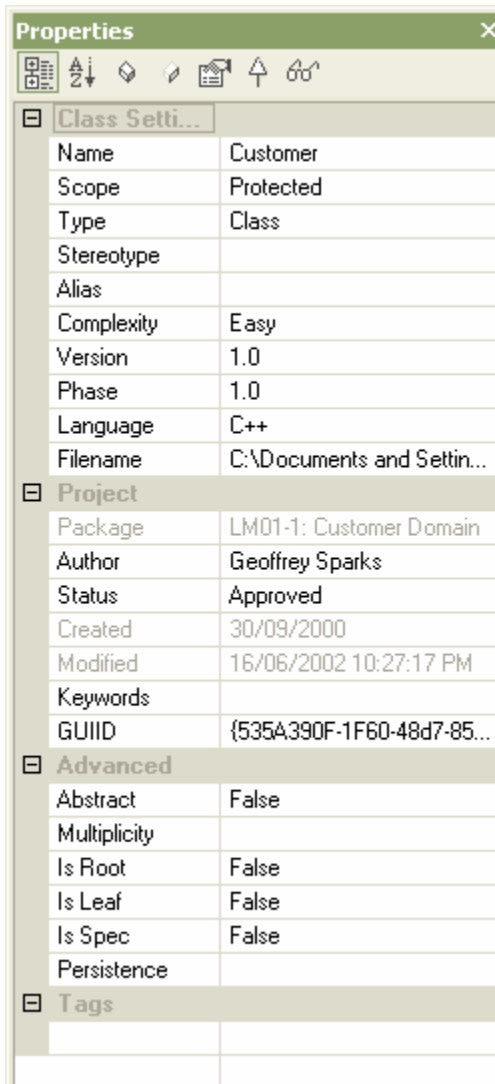
- [Project View](#)
- [Properties window](#)

- [UML Toolbox](#)
- [Resource window](#)
- [Notes window](#)
- [System window](#)
- [Testing window](#)
- [Maintenance window](#)
- [Source Code Viewer](#)
- [Element Browser](#)
- [Relationships window](#)
- [Rules window](#)
- [Hierarchy window](#)
- [Tagged Values window](#)
- [Project Management window](#)
- [Output window](#)
- [Instant Help window](#)

Some of these windows are discussed in this section, others in different areas of the helpfiles.

4.10.1 The Properties Window

The *Properties* window provides a convenient way to view (and in some cases edit) common properties of elements. When an element is selected, the Properties tab will show the element's name, stereotype, version, author, dates, and other pertinent information.



Tip: The properties tab can be a quick method of setting a single property (such as Phase or Status). To access and edit all properties of an element, double click on the element in a diagram or in the Project Browser.

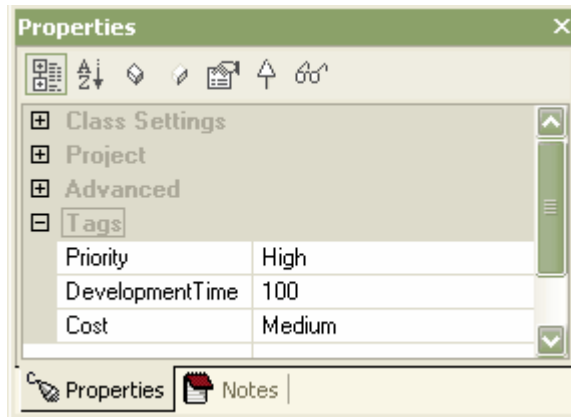
Properties Sections

The Properties tab is divided into three expandable sections:

- **General settings** - for the basic element details
- **Project** - for general housekeeping settings
- **Advanced** - only active for generalizable elements

A fourth section, **Tags**, appears only if the element has any tagged values. Tagged values may be edited but not added or deleted here.

Note: Informative tips appear in the bottom section of the Properties tab, unless the **Hide Properties Info Section** check box is checked in the **Tools | Options** dialog.



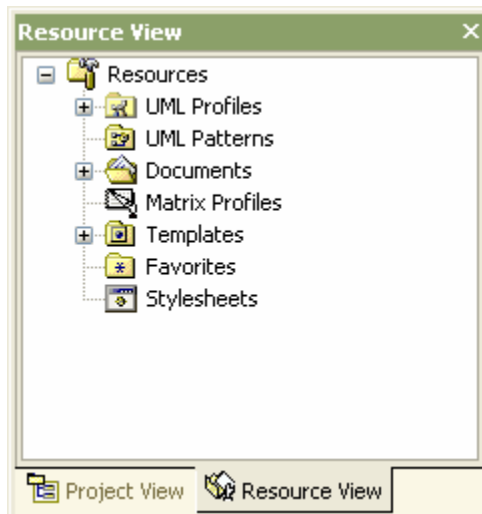
4.10.2 The Resource Window

The *Resource* window contains a collection of useful shortcuts and re-use functions.

[UML Profiles](#), [UML Patterns](#) and [MDG Technologies](#) provide a convenient way to insert complex new elements and features without having to retype or reconfigure each element. [Documents](#) provides a shortcut to the RTF and HTML documentation functions. The [Matrix Profiles](#) provide quick access saved Matrix setups.

Templates allow for customization of the RTF and HTML that make up EA reports. Favorites provides a shortcut to elements that you configure as a shortcut.

Tip: You can add a document to the shortcut list by going to *Project | Documentation | Rich Text Format Report*. Once you have defined your document select *Save as Document* and enter a name. The document will then appear in the Resource window.



See Also

[UML Profiles](#)

[UML Patterns](#)

[MDG Technologies](#)

[Resource Documents](#)

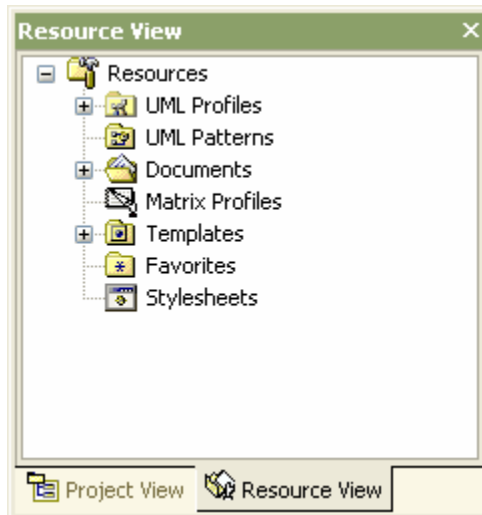
[RTF Templates](#)

[Web Style Templates](#)

[Matrix Profiles](#)
[Favorites](#)

4.10.2.1 The Resource View

The *Resource View* contains a tree of Model Elements, Scripts, Documents, UML Profiles and Patterns, Matrix profiles and more. This view holds elements that may be added to the current model, patterns, elements for re-use and additional information.



[UML Profiles](#) are discussed in the UML Profiles section of this manual.

See [UML Patterns](#) for more information on pattern support in EA.

The Documents node contains shortcuts to RTF documents. To add a document to the shortcut list, from the main menu select *Project | Documentation | Rich Text Format Report*. Once you have defined your document, select *Save as Document* and enter a name. The document will then appear in the Resource window. By right-clicking on the document name you can regenerate documents, or open them directly from EA.

The Templates folder holds templates you can design yourself to apply to your documentation in either [RTF style](#) or [Web style](#).

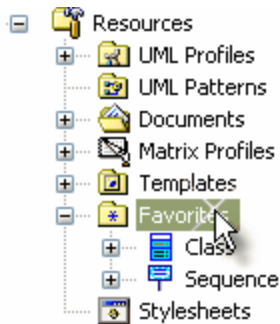
[The Matrix Profiles](#) node will contain a list of all defined Matrix Profiles. Double click on a profile to load the Matrix with the saved settings and source-target packages

The Style sheets section lets you import XSL Style sheets. These are then available in the Drop List on the XML Export dialog... if you select a Style sheets on export, EA will apply that style sheet to the XML generated before saving to file. This makes it convenient to generate other forms of output from the base XML content. Combined with UML Profiles, this provides a powerful means of extending EA to generate almost any content desired.

4.10.2.2 Favorites

The resource view contains a *Favorites* folder. Here you can link to any UML element from the model as a whole, and conveniently drag and drop instances or links to this element into other diagrams. This is particularly useful where certain elements - eg. the list of Actors in a systems - are re-used again and again,

and switching to the Actors folder is not convenient. In cases like this, using the Favorites folder makes managing and creating your model much easier.



Modifying the Favorites Folder

Add to the Favorites Folder

To add an element to the Favorites folder:

- Right click on the element to add in a diagram.
- In the Context Menu select *Add to Favorites*.
- Now switch back to the resource view and check the Favorites folder - the new element should be listed in its category within the favorites.

Delete from the Favorites Folder

To delete a favorite:

- Right click on it within the Favorites folder in the resource view
- Select *Delete* from the context menu.
- Confirm the action by pressing *Yes*.

View Properties of a Favorite

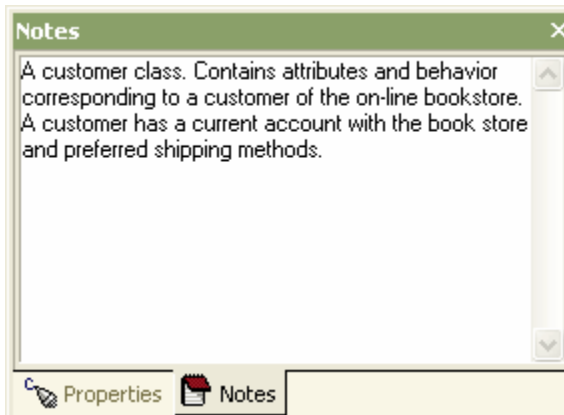
To view a favorite's properties from the Favorites folder:

- Select and right click on the favorite in the resource view.
- Select Properties from the context menu.

4.10.3 The Notes Window

The *Notes* window is used to view and edit all element documentation (notes) associated with any element or connector – either in a diagram (for elements and connectors) or in the Project Browser (elements only). When an element is selected, the note displayed here will change to reflect the current selection. If changes are made to notes in this tab, they will be saved. The notes tab may be used to display and edit notes for elements, diagrams, attributes, operations and connectors.

Notes are the main documentation feature you use to describe an element. In the documentation EA outputs, notes will feature prominently.



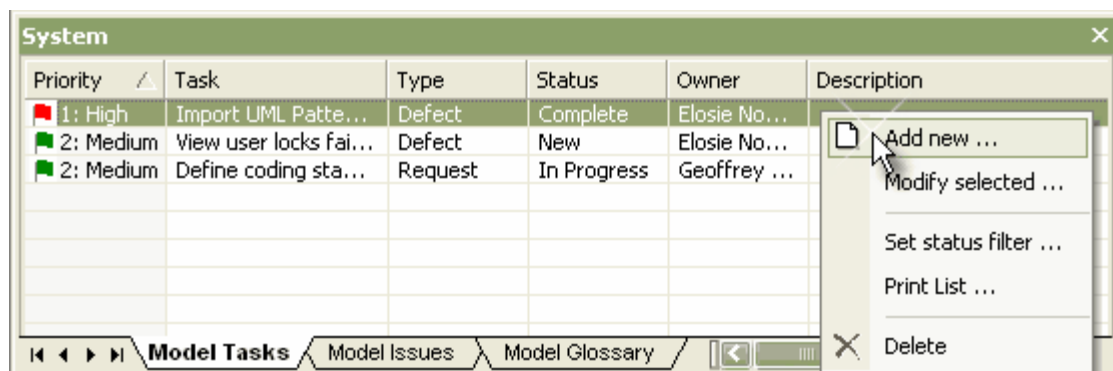
Tip: You can also edit notes by double clicking on an element in a diagram or in the Project Browser to open the property dialog.

4.10.4 The System Window

The **System** window documents tasks and issues which relate directly to the current project.

The System tab is divided into three sections:

- [Model Tasks](#) - a list of major project tasks that require attention. Tasks may be filtered based on their current status. Right click for a popup menu - or double click on a line item to modify details
- [Model Issues](#) - these are events, occurrences and situations that impact on project development and delivery. You can handle Issues using the right click menu or by double clicking on selected issues.
- [Model Glossary](#) - lists all the defined technical and business terms already defined for a model. You can add to the list, delete or change items and filter the list to exclude by type.



Tip: Right-clicking in the system window will bring up a context-sensitive menu, which has options for filtering tasks/issues by status, and glossary by term. You can also rearrange the sort-order by left-clicking in the title bar of the column which you wish the items to be indexed on.

4.10.5 The Source Code Viewer

The Source Code viewer can be used to view any source code you wish to open. If a class is selected, it will show the source code for that class, provided it has already been generated. For C++ a second tab will open to show the implementation file.

There are a number of options that change the way the source code viewer works. They can be altered via the *Local Options* dialog (*Tools | Options*). Select *Source Code Engineering - Code Editors* from the options tree.

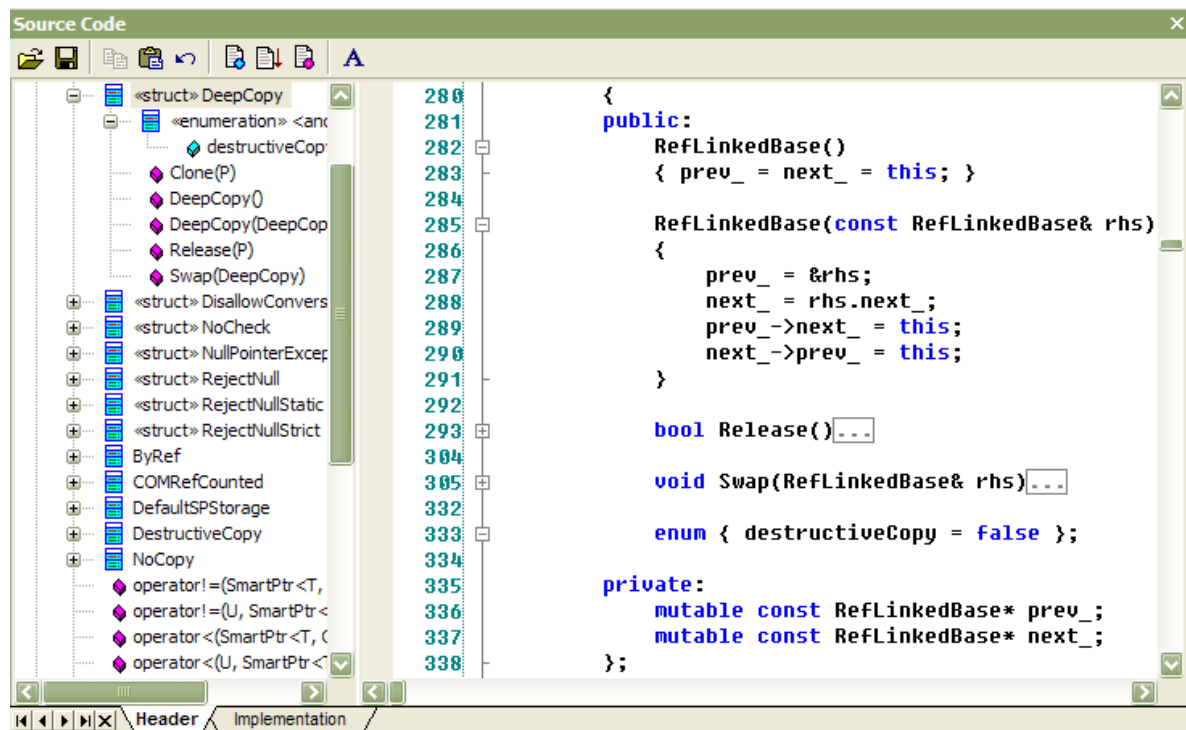
By default the source code viewer is set to:

- Parse all opened files, and show a tree of the results.
- Show line Numbers
- Have outlining enabled.

File Parsing

The source code editor parses files for a number of reasons. The first is to enable it to jump to the location in the file the currently selected item is found.

Additionally the parsing enables a structure tree showing an overview of the file in a similar fashion to the main project view is shown on the left. You can also select anything in that and jump to the appropriate line in the editor.



The Source Code Viewer window menu buttons

The menu buttons in the Source Code window allow the user to edit, view and interact with the code contained in the source code viewer. The table below details each button's functionality.



The Open button opens the source code from an existing file.

The Save Button saves the changes to the currently loaded source code.

The Copy button copies the highlighted text.



The Paste button pastes the text that is currently contained in the buffer to the source code viewer.



The Undo button cancels the previous action



The Generate and Reload button generates and reloads the current object source.



The Save and Resynch button save the source code and re synchronizes the class.



The Code Templates button access [Code Templates Editor](#).



Access a menu that allows quick access to relevant commands. The commands are:

- **Build** - run package build scripts
- **Test** - run package test scripts
- **Run** - run debug
- **Package Build Scripts** - Configure package build scripts



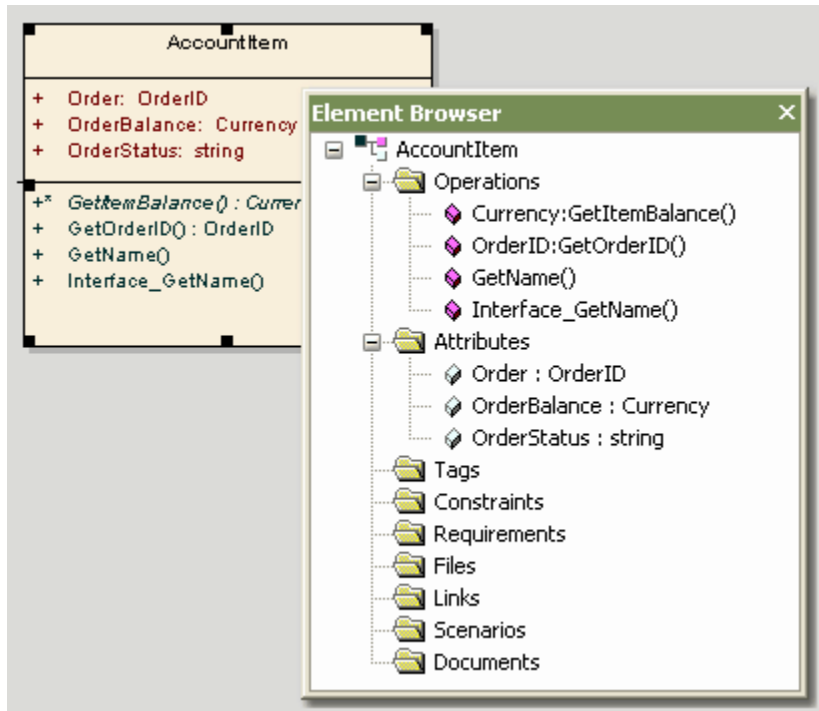
The Set Font button sets the font for the text contained in the Source Viewer.

See Also

- [Generating Source Code](#)

4.10.6 The Element Browser

The *Element Browser* displays all aspects of the selected element as shown below. It can be accessed from the *View | Other Windows | Element Browser* menu item, or via the context menu accessed by right clicking on the main menu.



The following are displayed where available:

- Operations
- Attributes
- Tags
- Constraints

- Requirements
- Files
- Links
- Scenarios
- Documents

4.10.7 The Relationships Window

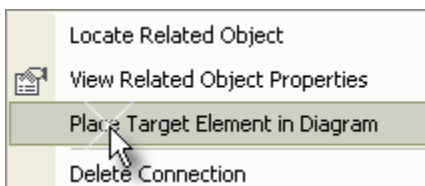
The *Relationships* window displays all connections between currently selected element and other elements. This provides a quick overview of an elements relations in the model.

For each link, the link type and target element are displayed. If an X appears in the 'Visible' column, the target element will be visible in the currently loaded diagram. This is useful when you are dragging related elements from the relations list onto the current diagram.

Double clicking a link in the list opens the link properties dialog where you can edit the link attributes. Right clicking a link opens the context menu - you may locate the related element, view the related element properties or delete the connector. You may also hide certain links from appearing in diagrams.

| Relationships | | | | | |
|---------------|---|------------------|--------|------------|------|
| Relation | ▼ | Target | Type | In Diagram | Role |
| ControlFlow | | Import Options | Object | Yes | |
| ControlFlow | | Import Options | Object | Yes | |
| ControlFlow | | Code Engineering | Object | Yes | |
| ControlFlow | | Code Engineering | Object | Yes | |

If an element is not visible in the current diagram, the context menu will have an option to place the selected element in the current diagram. This is useful when you are building a picture of what an element interacts with - especially when reverse engineering an existing code base.



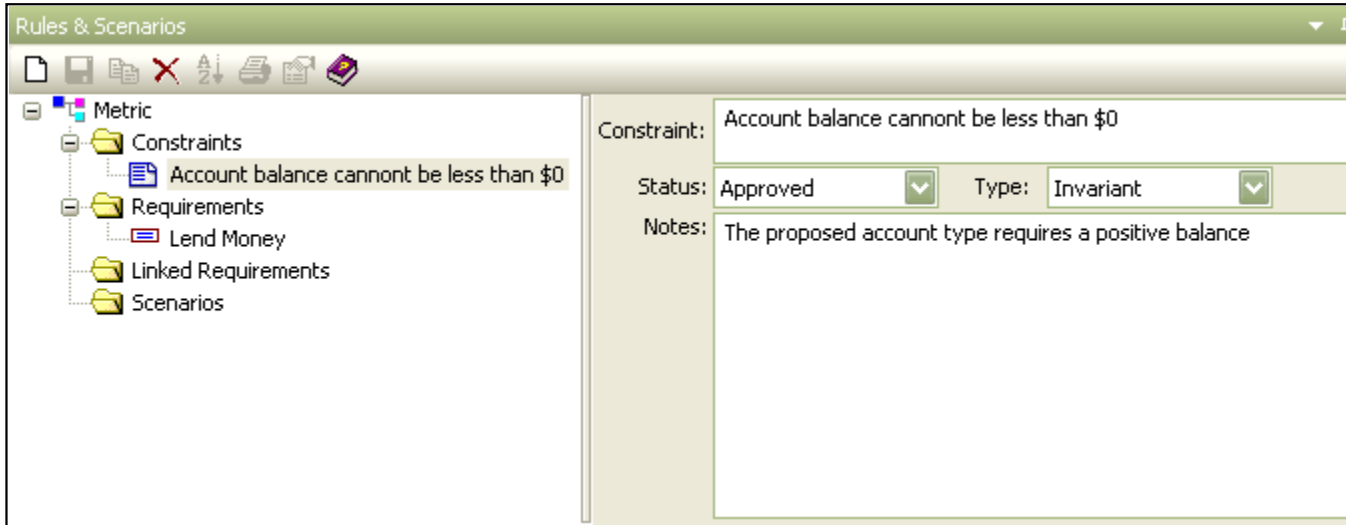
4.10.8 The Rules Window

The Rules window displays all requirements, constraints, Linked Requirements and Scenarios against an element. The Rules window provides a convenient way to quickly view, edit and add rules to an element.

To enter a new Requirement, Constraint or Scenario for an element:

1. Select the element and open the Rules window using **CTRL+SHIFT + 3** or **View | Other Windows | Rules & Scenarios**
2. Click on the desired folder, for example Requirements. You can:
 - add a new rule by pushing **CTRL + n**,
 - save by pushing **CTRL + s**

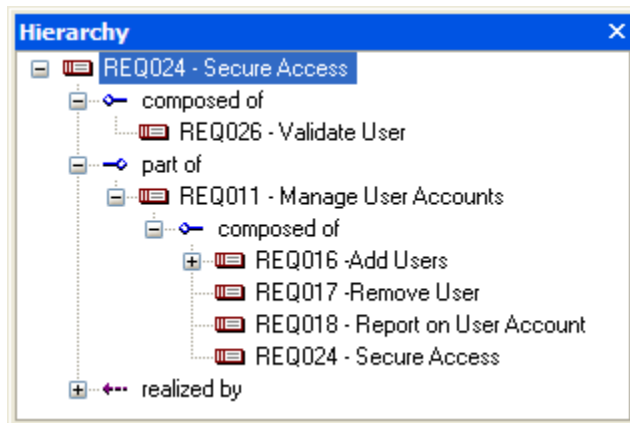
- delete by pushing **CTRL + d**.



4.10.9 The Hierarchy Window

The *Hierarchy* window shows a mini picture of the current element's composition with respect to other elements.

This information is derived from relationships with child or related classes. Relationships shown in the hierarchy include aggregation, inheritance and dependency, and embedded elements are also shown. This helps extend the picture of where an element exists in the model space.



Tip: You can alter the maximum number and the initial number of levels that the hierarchy will open to on the **Tools | Options | General** tab (accessed from the main menu).

4.10.10 The Tagged Values Window

The *Tagged Values* window is used to view and modify tagged values for the currently selected element - either in the current diagram or in the Project Browser.

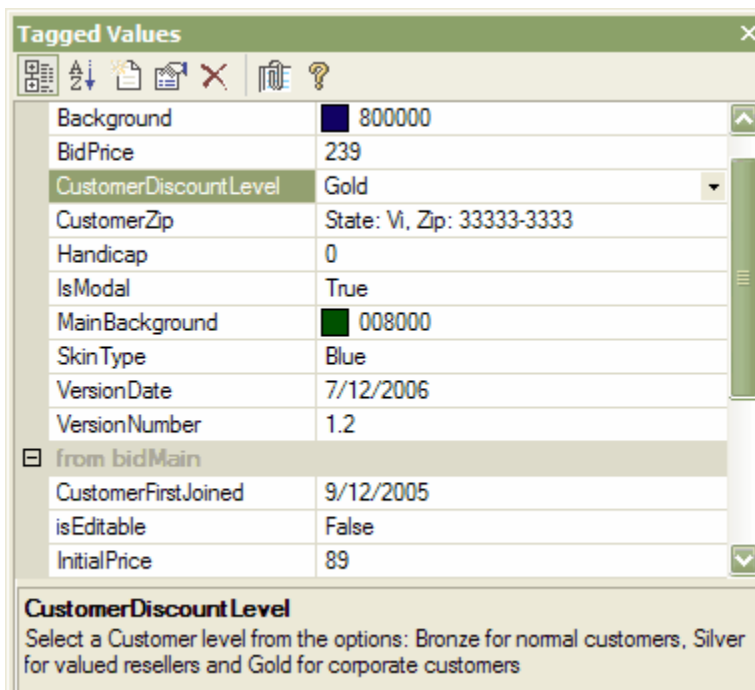
From the Tagged Values dockable window the user may perform the following actions:

- [Create a tagged value from the predefined tagged values type.](#)
- [Create a Custom tagged value type.](#)
- [Assign a defined tagged value to an item.](#)
- [Modifying tagged values with the Tagged Values window.](#)
- [Assign information notes to a tagged value.](#)

Model Elements and features with tagged values

The following model elements may use the Tagged Values window as a convenient way to quickly view and modify tagged values:

| | |
|------------------|---|
| Elements | Elements display their own tagged values along with any inherited values. |
| Object Instances | Object Instances display owned tags and those obtained from their classifier. |
| Ports and Parts | Ports and parts display information similar to objects and displays Port/Part "Type" instead of a classifier. Tags are included for all parents etc. of the Ports type. |
| Attributes | Includes owned tagged values and those received from attribute type classifiers, which the inclusion of any inherited ones. |
| Operations | Owned properties only |
| Connectors | Owned properties only |










When over-riding an inherited property, EA copies the tag from the parent down to the child element and sets the new value, leaving the original tag unchanged.

To edit Tagged Values the following options are available from the Tagged Values tab toolbar:

Tagged Values window menu buttons

The menu buttons in the Tagged Values menu allow the user to add, edit, sort, delete and arrange the tagged values of model features, the table below details each buttons functionality.



-  The Compartments button is used to display the tagged values in compartments.
-  The Sort Alphabetically button sorts the current tagged values for the element alphabetically.
-  The New Tag button adds an new tagged value.
-  The Edit Notes button allows the user to create notes that explain the purpose of the tagged value.
-  The Delete selected button removes the currently selected tagged Value.
-  The Default tagged values type button allows quick access to tag definitions created in the Configuration main menu.
-  The help button is used to allow the user to obtain help relating to the use of the tagged values window.

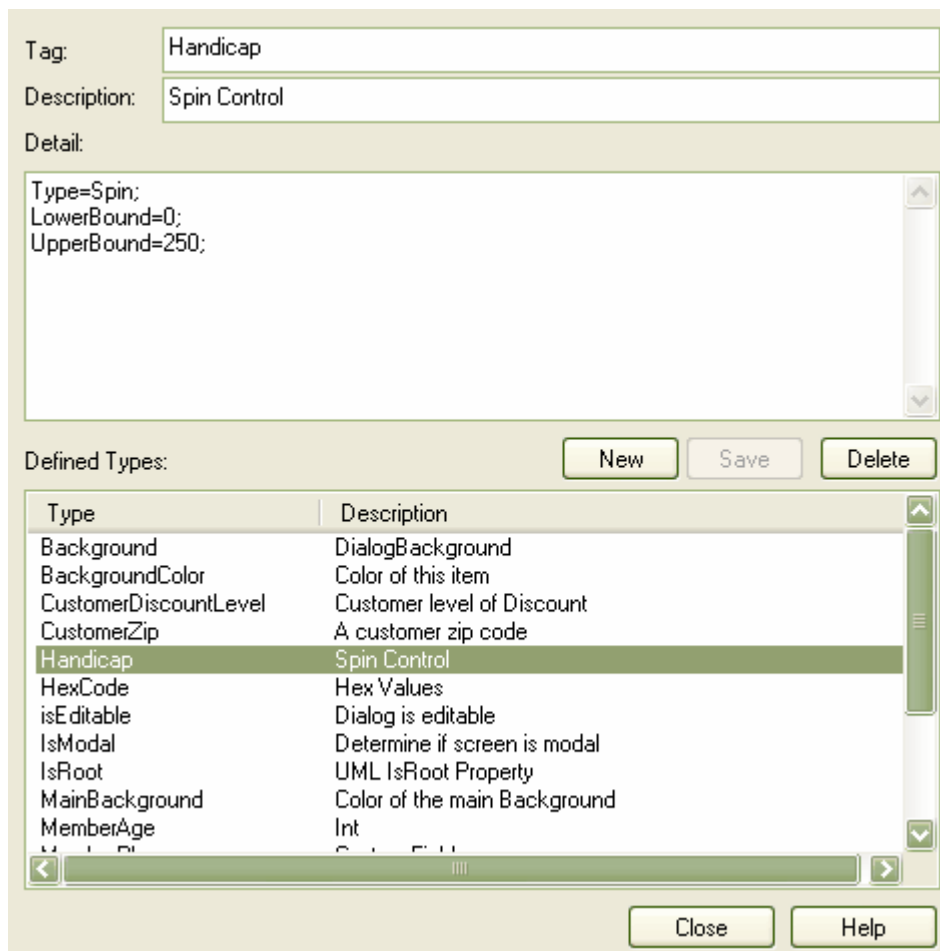
4.10.10.1 Predefined Tagged Value Types

A range of predefined tagged values have been created in EA to allow the user to rapidly create masked tagged values. This allows the user to create structured tags that adhere to a specific format. For example, model features that use the predefined tag "Boolean" may use the tagged values tab to assign a value of true or false and no other value. In addition it is possible to define a Custom masked tag type allowing the user to define an almost unlimited amount of structured tag types.

Creating Structured Tags

To create a structured tag use the following instructions:

1. From the *Settings* menu mouse over the *UML* item and select *Tagged Values* from the sub menu. This will bring up the *Tagged Values* dialog.
2. In the *Tag* field give the Tag an appropriate name.
3. In the *Description* field enter of the purpose of the tag if required.
4. In the *Detail* field enter a value for the type of tagged value and any extra information as required by the predefined type, in the example below the predefined values have been set for a Spin type allonge with the Upper and Lower Bound for the field.



Types and Arguments

This table details the predefined tagged value types along with the format used to create the initial values for their use.

| | | |
|------------------------|---|--|
| Integer | Type=Integer; | This predefined type allows the entry of an Integer value. |
| Float, Decimal, Double | Type=Float; Type=Decimal; Type=Double; | This predefined type allows the entry of a Float, decimal or a Double value. These types all map to the same type of data. |
| String | Type=String; | This predefined type allows the entry of a String value. |
| Enum | Type=Enum; Values=Val1,Val2,Val3; Default=Val2; | The Enum predefined type allows the user to define a comma separated list where Val1, Val2 and Val3 represent values in the list and Default represents the default value of the list. |
| Const | Type=Const; Default=Val; | The Const is a predefined type allows the user to create a read-only constant value. |
| Color | Type=Color; | Using the Color predefined type allows the user to input a color value from a color chooser menu. |

| | | |
|------------|---|--|
| Custom | Type= Custom; | Using the Custom predefined type allows the user to create their own template for predefined types, more information is provided in the Custom tagged vales type section . |
| DateTime | Type=DateTime; | Using the DateTime predefined type allow the user to input the date and time for the tagged value from a calendar menu. |
| Boolean | Type=Boolean; | Using the boolean predefined type allows the input of a true or false value. |
| Memo | Type=Memo; | The Memo predefined type allows the input of large and complex tag values. |
| Spin | Type=Spin; LowerBound=x; UpperBound=x; | The Spin predefined type allows the user to create a spin control with the value of LowerBound being the lowest value and UpperBound being the highest value. |
| File | Type=File; | The File predefined type allows the input of a filename from a file browser dialog. The named file may be launched in its default application. |
| Classifier | Type=Classifier; Values=Type1,Type2; Stereotypes=Stereotype1; | The Classifier predefined type allows the selection of an element from the model where Type1 and Type2 represent the allowed element types and Stereotype1 represents an allowed stereotype. |

Tag Filters

It is also possible to restrict where a predefined tagged value is available. This table details filters that can be applied to restrict where a tagged value may be applied.

| | | |
|----------------|------------------------|--|
| AppliesTo | AppliesTo=Type1,Type2; | The AppliesTo filter restricts the element types this filter can be applied to where Type1 and Type2 are the valid types. Possible values are all element types, all connector types, Attribute, Operation and OperationParameter. |
| BaseStereotype | BaseStereotype=S1,S2; | The BaseStereotype filter restricts the stereotypes that this tag belongs to where S1 and S2 are the allowed stereotypes. |

4.10.10.2 Predefined Reference Data

This table details the predefined tagged value types that is used to return the values held in a relevant table in EA along with the syntax required for their use:

| | | |
|-----------------|---|---|
| Authors | Type=Enum; Type=List; List=Authors; | This predefined type returns a drop down list with a list of the Authors that have been defined for the EA model. |
| Cardinality | Type=Enum; Type=List; List=Cardinality; | This predefined type returns a drop down list with a list of the Cardinality types that have been defined for the EA model. |
| Clients | Type=Enum; Type=List; List=Clients; | This predefined type returns a drop down list with a list of the Clients that have been defined for the EA model. |
| ComplexityTypes | Type=Enum; Type=List; List=ComplexityTypes ; | This predefined type returns a drop down list with a list of the Complexity types that have been defined for the EA model. |

| | | |
|------------------|--|---|
| ConstraintTypes | Type=Enum; Type=List; List=ConstraintTypes; | This predefined type returns a drop down list with a list of the Constraint types that have been defined for the EA model. |
| EffortTypes | Type=Enum; Type=List; List=EffortTypes; | This predefined type returns a drop down list with a list of the Effort types that have been defined for the EA model. |
| MaintenanceTypes | Type=Enum; Type=List; List=MaintenanceTypes; | This predefined type returns a drop down list with a list of the Maintenance types that have been defined for the EA model. |
| ObjectTypes | Type=Enum; Type=List; List=ObjectTypes; | This predefined type returns a drop down list with a list of the Object types that have been defined for the EA model. |
| Phases | Type=Enum; Type=List; List=Phases; | This predefined type returns a drop down list with a list of the Phases that have been defined for the EA model. |
| ProblemTypes | Type=Enum; Type=List; List=ProblemTypes; | This predefined type returns a drop down list with a list of the Problem types that have been defined for the EA model. |
| RoleTypes | Type=Enum; Type=List; List=RoleTypes; | This predefined type returns a drop down list with a list of the Role types that have been defined for the EA model. |
| RequirementTypes | Type=Enum; Type=List; List=RequirementTypes; | This predefined type returns a drop down list with a list of the Requirement types that have been defined for the EA model. |
| Resources | Type=Enum; Type=List; List=Resources; | This predefined type returns a drop down list with a list of the Resources that have been defined for the EA model. |
| RiskTypes | Type=Enum; Type=List; List=RiskTypes; | This predefined type returns a drop down list with a list of the Risk types that have been defined for the EA model. |
| ScenarioTypes | Type=Enum; Type=List; List=ScenarioTypes; | This predefined type returns a drop down list with a list of the Scenario types that have been defined for the EA model. |
| TestTypes | Type=Enum; Type=List; List=TestTypes; | This predefined type returns a drop down list with a list of the Test types that have been defined for the EA model. |

Creating Predefined Reference Data Tagged Value Types

To create a predefined reference data tagged values type use the following instructions:

1. From the *Configuration* menu mouse over the *UML* item and select *Tagged Values* from the sub menu.
2. This will bring up the *Tagged Values* dialog, in the *Tag* field give the Tag an appropriate name.
3. In the *Description* field enter of the purpose of the tag if required.
4. In the *Detail* field enter a value for the type of tagged value and any extra information as required by the predefined type, in the example below the predefined values return values have been set to return the values for all of the Authors in the EA model.

Tag:

Description:

Detail:

```
Type=Enum;
Type=List;
List=Authors;
```

Defined Types:

| Type | Description |
|-----------------------|------------------------------|
| Background | DialogBackground |
| BackgroundColor | Color of this item |
| CustomerDiscountLevel | Customer level of Discount |
| CustomerZip | A customer zip code |
| Handicap | Spin Control |
| HexCode | Hex Values |
| isEditable | Dialog is editable |
| IsModal | Determine if screen is modal |
| IsRoot | UML IsRoot Property |
| MainBackground | Color of the main Background |
| MemberAge | Int |

- This will allow the user to assign any of the previously defined Authors to a model feature (Model features which may have tagged values applied to them are detailed in the [Model Elements and features with tagged values](#) section).

| Tagged Values | |
|------------------|-----------------|
| Club (Class) | |
| Author | Tom O'Reilly |
| Handicap | John Redfem |
| MemberAge | Kim Redfem |
| MemberPhone | Tom O'Reilly |
| MemberPhone | (044) 333-33-33 |
| MembershipExpire | 30/Dec/1899 |

Note: If the values in the reference data are changed after the tagged value type is created, EA will need to be reloaded in order to reflect the changes with the tagged value type.

4.10.10.3 Creating a CustomTagged Value Type

Creating a custom tagged value allows the user a great deal of flexibility for creating their own masked tagged values. To create a user defined masked tagged value follow the procedure detailed below:

1. From the *Configuration* menu mouse over the *UML* item and select *Tagged Values* from the sub menu.
2. This will bring up the *Tagged Values* dialog, in the *Tag* field give the tag an appropriate name.
3. In the *Description* field enter of the purpose of the tag.
4. In the *Detail* field enter Type=Custom;
5. By defining the type as Custom, the it is now possible for the user to set up the appropriate mask using the following characters to define the format of the mask:

| | |
|---|--|
| D | Using this mask allows the tagged value to display digits only. |
| d | Using this mask allows the tagged value to display digits or spaces. |
| + | This mask allows for the use of "+", "-" or spaces. |
| C | This mask allows for the use of Alpha characters only. |
| c | Using this mask allows the tagged value to be an Alpha character or a space. |
| A | This mask allows for the use of Alphanumeric characters. |
| a | This mask allows the tagged value to Alphanumeric values or a space. |

In the diagram detailed below the Mask configuration option shows syntax which first defines seven blank spaces, which will be occupied by characters determined by the template option. The first two visible characters in the Mask option are represented by a lower case "c" indicating that the allowable information may entered as either an Alpha character or as a space, the following blank spaces again indicate space defined by the template option and the remaining characters are defined by the "d" character which represents the allowable characters as digits or spaces. The hyphen will be present in the final output, splitting up the digits.

With the Template configuration option, the syntax defines the template of the masked option by occupying the blank spaces which are present in the Mask option. The template is used to ensure that this information is present with every use of this custom tagged value. The underscored values indicate the area that will be occupied by data inputted by the user and defined in the mask option.

Tag:

Description:

Detail:

```
Type=Custom;
Mask= cc ddddd-dddd;
Template=State: __, Zip: ____-____;
```

Defined Types:

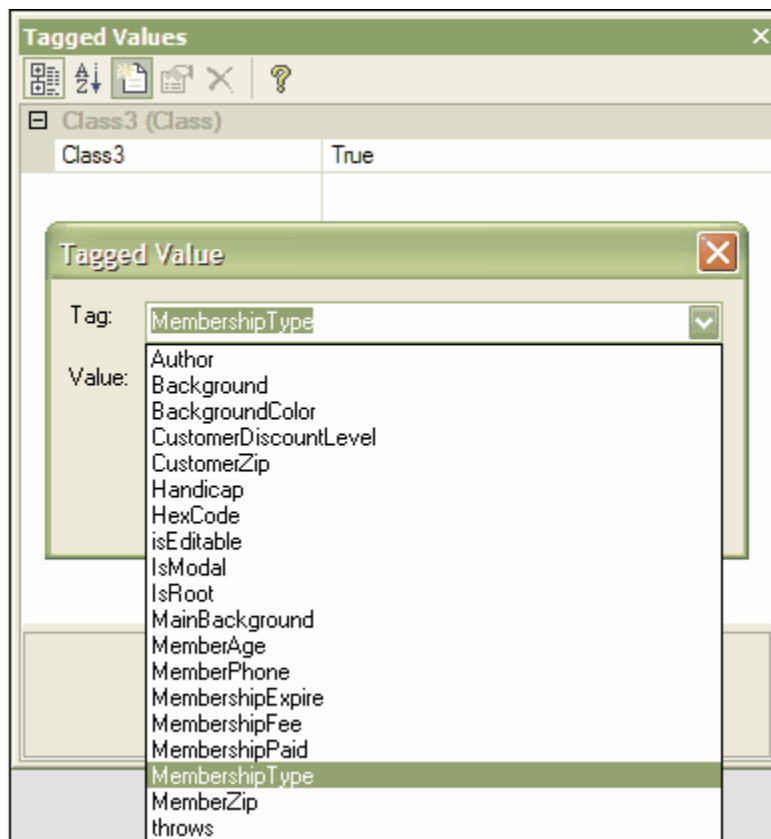
| Type | Description |
|------------------|------------------------------|
| isEditable | Dialog is editable |
| IsModal | Determine if screen is modal |
| IsRoot | UML IsRoot Property |
| MainBackground | Color of the main Background |
| MemberAge | Int |
| MemberPhone | Custom Field |
| MembershipExpire | Date |
| MembershipFee | decimal value |
| MembershipPaid | Boolean |
| MembershipType | Enumerated Values |
| MemberZip | Zip code |

4.10.10.4 Assigning defined Tagged Value to an Item

Tagged Values may be assigned to several model features, to add a tagged value use the following instructions (Model features which may have tagged values applied to them are detailed in the [Model Elements and features with tagged values](#) section):

1. Create user defined tagged values either using a predefined tagged value type or by creating a Custom tagged value type.
2. Select the model feature that you wish to associate the defined tagged value.
3. Ensure that the *Tagged Values* Window is visible by opening the *View* menu, mousing over the *Other Windows* item and selecting *Tagged Values* from the submenu. Alternatively press *Ctrl + Shift +6*.
4. Press either the *New Tags* button or the *Ctrl + N* hotkey combination to bring up the Tagged Values Dialog .
5. From the drop down in the *Tag* field menu select the appropriate defined tagged value that you wish to assign to the item.

Note: The direct entry of predefined tag values is only available for predefined tags of type "string".

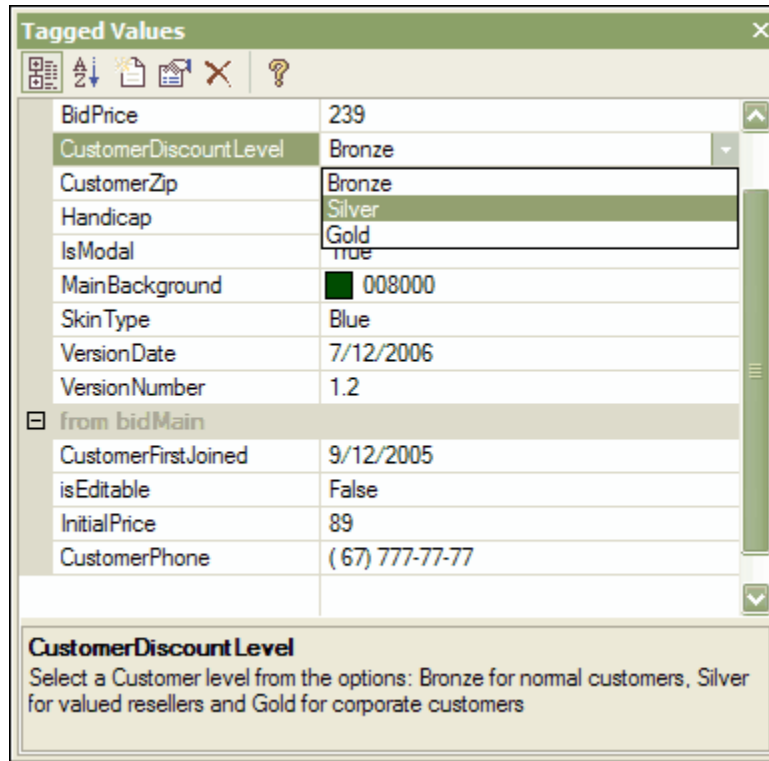


6. To confirm the selection of the tagged value, press the **OK** button.

Modifying Tagged Values with the Tagged Values Window

Once the tagged value has been assigned to the model feature it is possible to edit the values from the Tagged Values dockable window. To do this use the following instructions (Model features which may have tagged values applied to them are detailed in the [Model Elements and features with tagged values](#) section):

1. Ensure that the **Tagged Values** Window is visible by opening the **View** menu, mousing over the **Other Windows** item and selecting **Tagged Values** from the submenu. Alternatively press **Ctrl + Shift + 6**.
2. Select the model feature that you wish to edit the tagged values, this will show all of the tagged values for the selected model feature.
3. Edit the fields as appropriate. The information entered may only reflect the masked value types that have been defined either as a predefined type or in the format defined by the creation of the Custom tagged type.
4. The example below shows a predefined Enum type having its value modified.

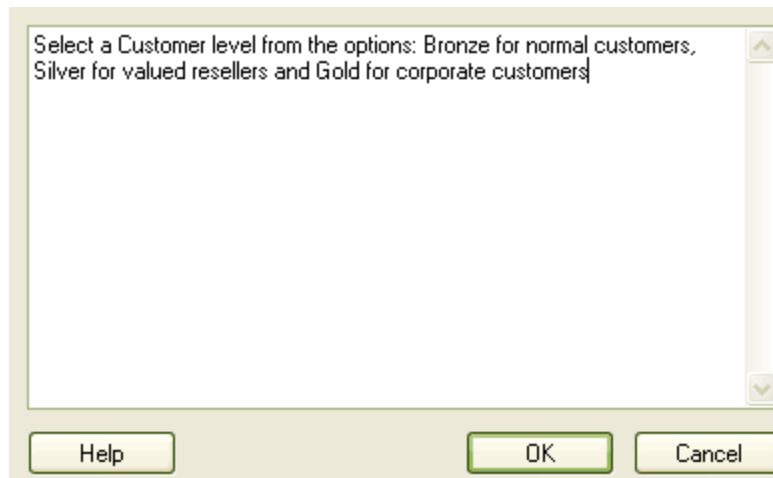


Note: To override a Tagged Value defined in a parent element, edit the value in the from [parentname] compartment of the Tagged Values dockable window, once this has been done the tag will be moved into the selected elements tagged values and this will not effect the tagged values defined in the parent element.

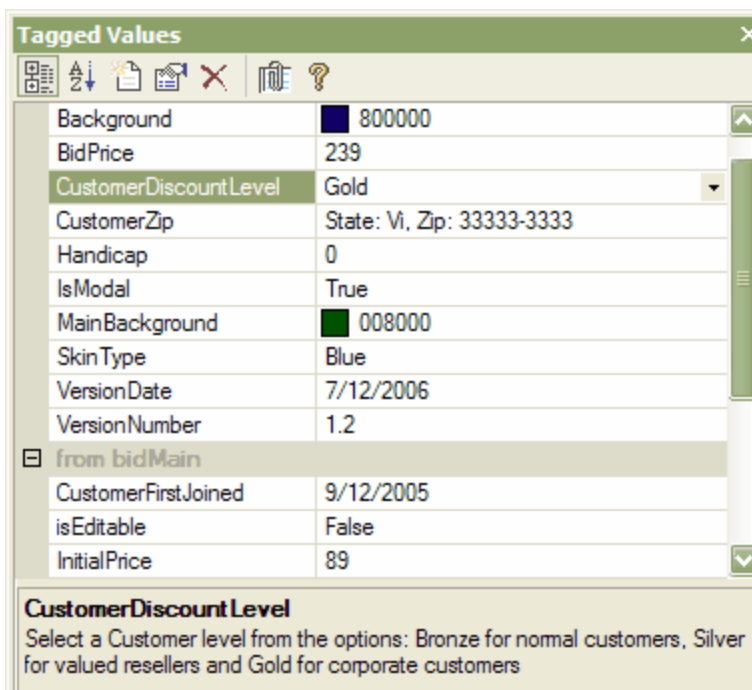
4.10.10.5 Assigning Information to a Tagged Value

Once the tagged value has been assigned to a model feature, it is possible to add information and notes describing the tagged value to the information property of the tagged value (Model features which may have tagged values applied to them are detailed in the [Model Elements and features with tagged values](#) section). To facilitate this from the Tagged Values dockable window use the following instructions:

1. Ensure that the Tagged Values Window is visible by opening the *View* menu, mousing over the *Other Windows* item and selecting *Tagged Values* from the submenu. Alternatively press *Ctrl + Shift + 6*.
2. Select the model feature that you wish to edit the tagged values, this will display the tagged values for the selected model feature.
3. Select the tagged value that you wish to add information to.
4. Then press the *Edit Notes* button or press the *Ctrl + E* hotkey combination. This will bring up the *Tagged Value Note* dialog.



5. Enter the information relating to the tagged value into the *Note* field, this information will be displayed in the lower portion of the *Tagged Values* dockable window whenever the tagged value is selected.

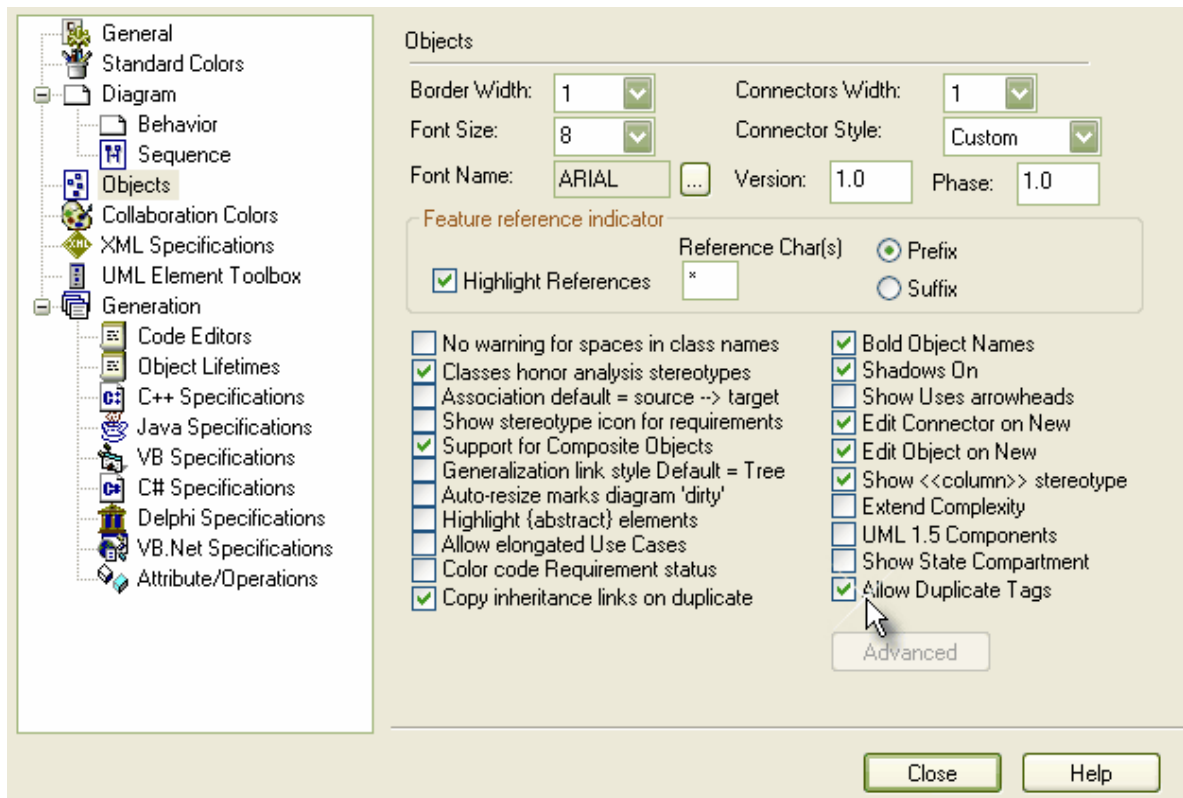


4.10.10.6 Allow Duplicate Tags

Tagged values are by default set to suppress duplicate values, this setting is used to facilitate inherited and overridden tag names. However, if the user wishes to allow duplicate tagged values this can be achieved as follows:

1. Go to the *Tools* menu and select *Options*.
2. This will open the *Local Options* dialog, from the hierarchical tree select the *Objects* item.

- Ensure that the *Allow Duplicate Tags* item is checked.



4.10.11 The Project Management Window

The Project Management window allows for the input of the resources, effort, risk and metrics that may be added to elements contained in the model.

Open the *Project Management* window by selecting *Resourcing Metrics & Risk* from the *Element* menu or open the *Project Management* window from the *View | Other Windows | Project Management* (or press the *Ctrl + Shift + 7* hotkey combination).

The screenshot shows the 'Project Management' window with a table of resource allocation data. The table has the following columns: Resource, Role, Allocated Time, % Completed, Start, End, and Description.

| Resource | Role | Allocated Time | % Completed | Start | End | Description |
|----------|---------------------|----------------|-------------|------------|------------|---|
| Gerrico | C++ Programmer/Help | 5.000000 | 23 | 17/07/2006 | 18/07/2006 | Add client data to invoicing system |
| Kelly | Application Analyst | 5.000000 | 95 | 17/07/2006 | 17/07/2006 | Complete analysis of the evaluation version systems core routines |
| | | | | | | |
| | | | | | | |

The window also shows tabs for 'Resource Allocation', 'Effort', 'Risks', and 'Metrics' at the bottom.

Right click on the list to view the context menu which allows you to add, modify and delete list items. For more information see the [Adding, Modifying and Deleting Tasks](#) topic.

Toolbar



New : Create New item

Save: Save changes to an item

Copy: Copy allows you to duplicate an existing entry. You need to change an items Role for this to become enabled.

Delete: Delete an item from the list

Sort: Sort Items in the list

Print: Print items data from the list

Show/Hide Details : Swap between old and new window style

Help: Show help contents for this window

See Also

- [Resource Management](#)
- [Effort Management](#)
- [Risk Management](#)
- [Metrics](#)

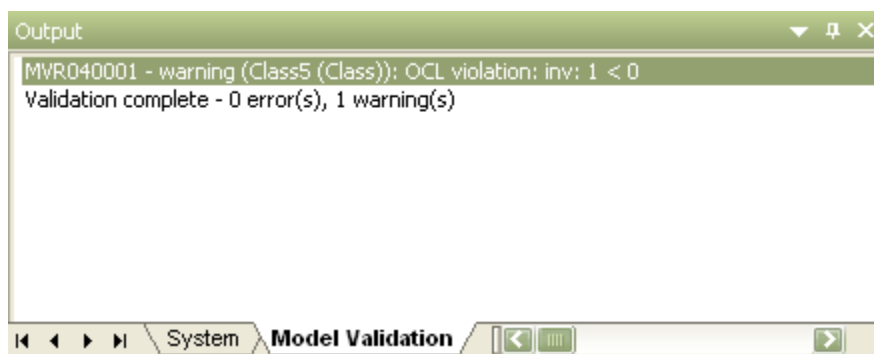
4.10.12 The Output Window

The output window is used to display line items which may be either system generated or Add-In generated. Examples of situations where EA will generate items are:

- Validation Items
- Launch of external processes
- Command line output of Build and Test
- Parse errors generated during import

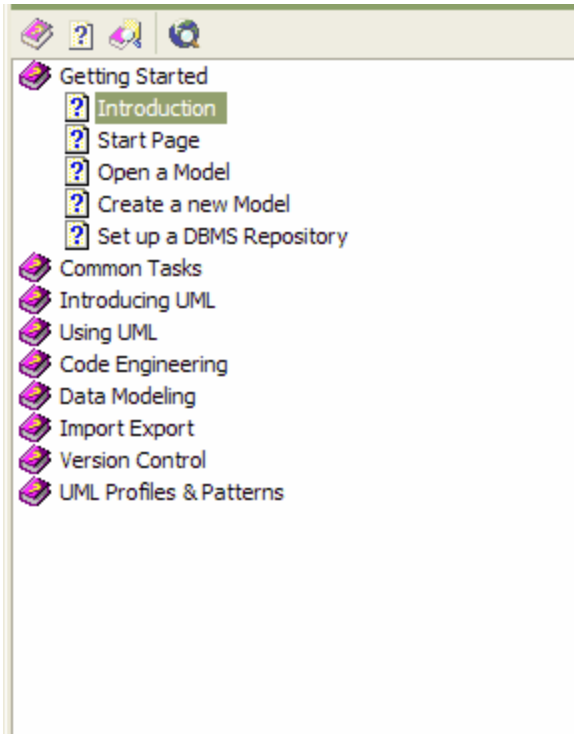
Double clicking on model validation errors or parsing errors will take you to the source of the error.

For more information on using the Automation interface to allow Add-Ins make use the output window see the [Add-Ins | Repository](#) topic.



4.10.13 Instant Help Window





The Instant Help window allows access to the most popular help file reference topics. Double-clicking on a topic opens that topic in the main window of EA. Instant Help allows the user to interact with specific items in the EA help file, allowing access to the contents, search and index page of the EA help file. The Instant Help also allows the user to open an Internet search page within the Instant Help tab of the EA workspace.



Instant Help window menu buttons

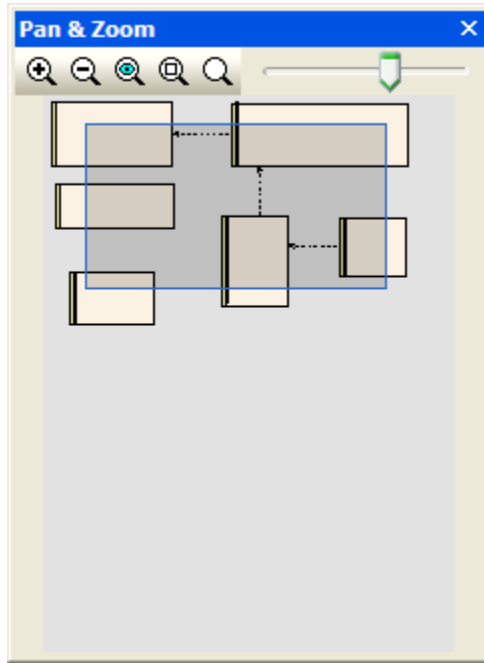
The menu buttons in the Instant Help menu allow the user to interact with specific items in the EA help file and open a browser window from within EA. The table below details each button's functionality.



| | |
|---|---|
|  | The Open Help Contents button opens the help file in a separate window at the Introduction page. |
|  | The Open Help Search button opens the help file in a separate window at the search page. |
|  | The Help Index Page button opens the help file in a separate window at the Index page. |
|  | The open Web Search button allows open a web search page within EA, the default page is www.google.com but the page may be altered to another web page from the Local Options dialog. |

4.10.14 The Pan & Zoom Window

The Pan & Zoom window allows you to navigate quickly around large diagrams. The shaded box represents the viewed area on the open diagram. Holding the *Left Mouse* Button down inside the window allows you to pan the open diagram by moving the shaded box. To zoom use either the *Zoom Slider* or buttons located on the tool bar.

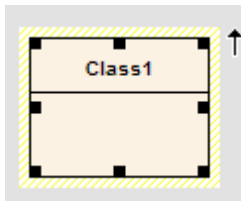


The Toolbar provides the following functions (in order)

- Zoom In
- Zoom Out
- Zoom to fit diagram
- Zoom to fit page
- Zoom to 100%
- Zoom Slider

4.11 The Quick Linker

The Quick Linker provides a simple and fast way to create new elements and connectors on a diagram. When an element is selected in a diagram, the Quick Linker icon is displayed in the upper right corner of a element, as shown below:



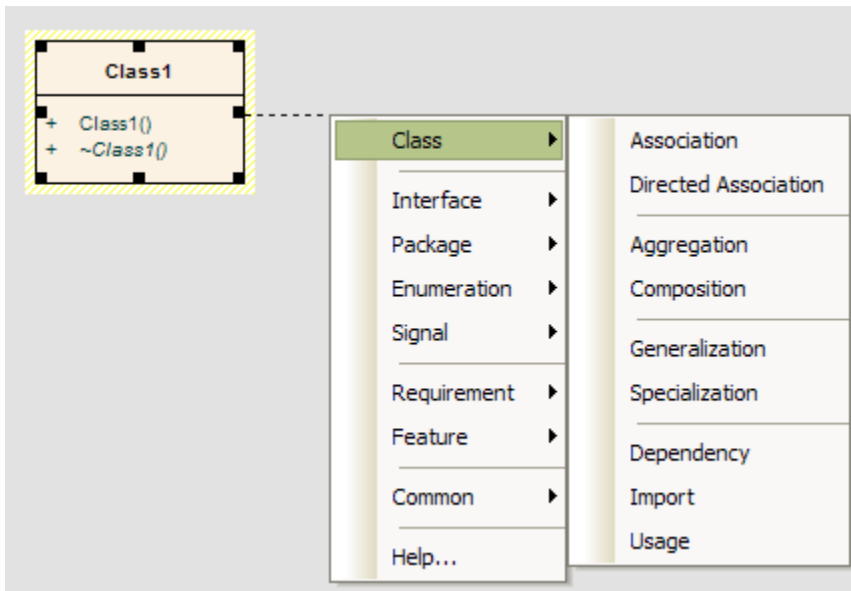
Simply clicking and dragging the icon allows you to create new connectors and elements. The following sections illustrate the use of the Quick Linker to create and link elements on a diagram:

- [Creating New Elements](#)
- [Creating Connections Between Elements](#)

4.11.1 Creating New Elements

Use the following steps to create new elements using the Quick Linker:

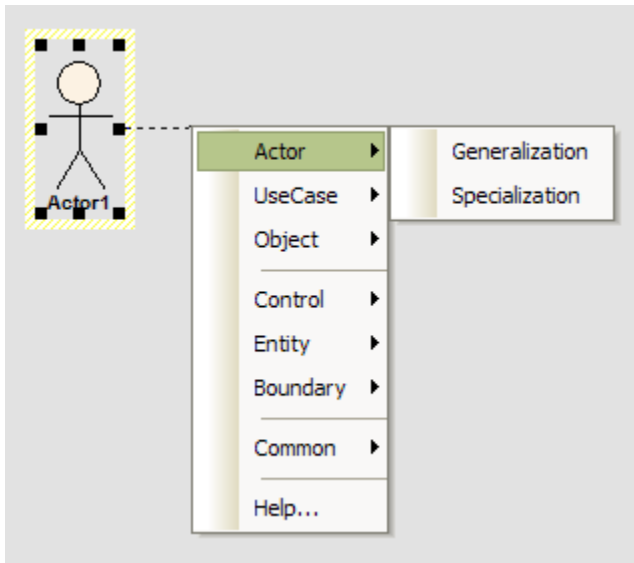
1. Select a start element on the current diagram
2. Drag the Quick Linker icon onto an empty area of the diagram
3. Use the Quick Linker context menu to select the type of element and connector.



Tip: Holding the shift key while selecting the type of connector, allows you to select an existing classifier as the target.

Tip: For rapid modeling, you can suppress the properties dialog when creating new elements. See the option [Tools | Options | Objects | Edit Object on New](#)

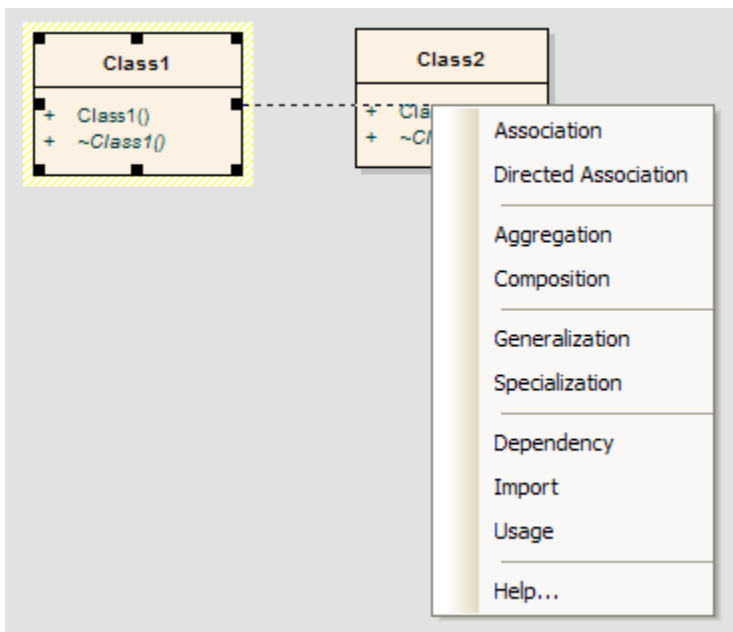
Note: The available Quick Link options depend on the type of element selected. For example, the Quick Link options for a class (above) differ from those of an Actor (below)



4.11.2 Creating Connections Between Elements

Use the following steps to create new connectors using the Quick Linker:

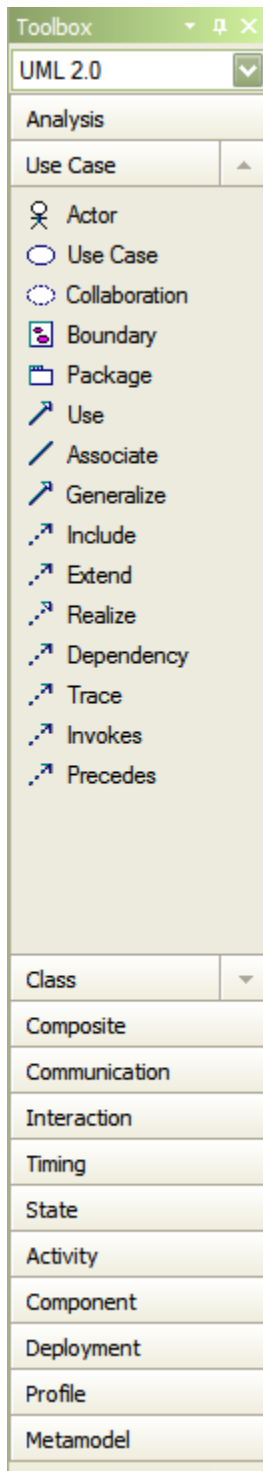
1. Select the start element on the current diagram
2. Drag the Quick Linker icon onto another element in the diagram
3. Use the Quick Linker context menu to select the type of element and connector



4.12 The UML Toolbox

The *UML Toolbox* is used to create elements and connectors on a diagram. Related UML elements are organized into categories within the toolbox, as shown below. The Toolbox categories can be customized via [Workspace Perspectives](#) or by adding [MDG Technologies](#) and [UML Profiles](#) to the toolbox.

The UML Toolbox may be docked on either side of the diagram, or free floated on top of the diagram to expose more surface for editing.




Creating Elements and Connectors:

1. Open a diagram (select the desired diagram from the *Project Browser* if required)
2. Select the appropriate category in the UML Toolbox, for example "Use Case"
3. Select the desired item within the category, for example "Actor" or "Associate"
4. For element items, click anywhere on the diagram to place the new element. For connector items, drag the cursor between the start and end elements on the diagram.
Alternatively, drag from the start element to an empty area of the diagram and the Quick-linker will allow the creation of the end element.
5. Edit the [element properties](#) or [connector properties](#), as required.

Note: The toolbar categories within the toolbar relate to specific UML diagrams. When a given diagram is opened, the toolbox will automatically open the corresponding category. This does not prevent the use of elements and connectors from other categories in a given diagram, though some combinations may not represent valid UML.

Note: Dropping a Package from the toolbox into a diagram, will create a new package in the Project Browser - and a default diagram of the same type as the current diagram.

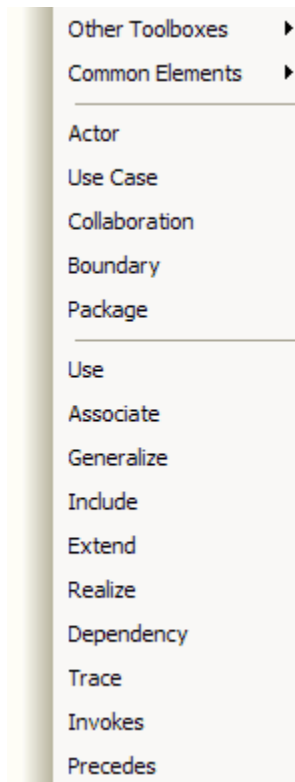
Tip: You can hide and show the UML Toolbox using the  *Hide/Show Toolbox* button on the shortcut toolbar.

4.12.1 UML Toolbox Shortcut Menu

Instead of employing the full UML Toolbox users may access the UML Toolbox shortcut menu to add elements and links into a diagram. The items that are available for use with the Toolbox Shortcut menu are determined initially by the type of diagram being used or the current UML category type selected in the UML Toolbox. The advantage of using the UML Toolbox shortcut menu is that it allows for an increased amount of the workspace to be used for diagramming rather than being used to display menus.

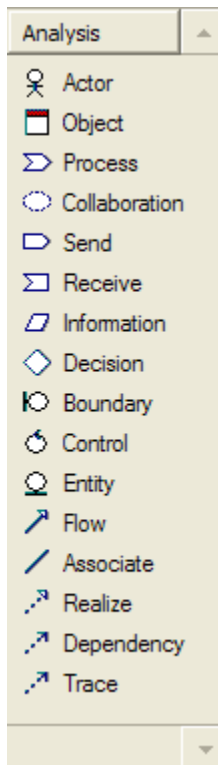
To use the UML Toolbox Shortcut Menu use the following steps:

1. Open a diagram.
2. Press the *Space Bar* or *Ctrl-Right Click* the mouse.
3. Select the element or links from the menu that you wish to include in the diagram and click on the selection. The element or link will be added to the diagram.
4. If you require an element or link from a different UML category, these can be accessed from the *Other Toolboxes* menu item.



4.12.2 Analysis Group

Analysis type elements are used early in modeling to capture business processes, activities and general domain information. They are generally used in [Analysis diagrams](#).



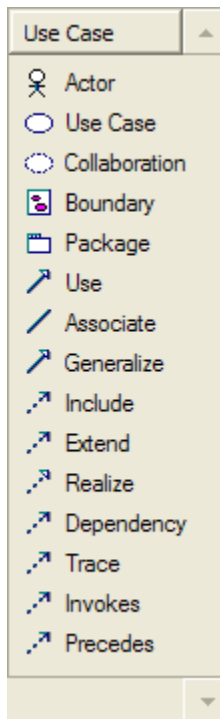
The elements and relationships in the Analysis group are used for early modeling of business processes, activities and collaborations. You can use stereotyped activities to model business processes, or stereotyped elements to capture standard UML business process modeling extensions such as *worker*, *case worker*, *entity*, *controller* and etc.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.3 Use Case Group

Use Case elements are used to build [Use Case models](#). These describe the functionality of the system to be built, the requirements, the constraints and how the user will interact with the system. Often sequence diagrams are associate with Use Cases to capture work flow and system behavior.



The Use Case group is used to model the system functionality from the perspective of a system user. The user is called an Actor and drawn as a stick figure, although the user may be another computer system or similar. A use case is a discrete piece of functionality the system provides that allows the user to perform some piece of work or something of value using the system.

Examples of use cases are: login, open account, transfer funds, check balance and logout; each of these implies some purposeful and discrete functionality the system will provide a user.

The connectors available are: association (an actor uses a use case), extend (one use case can extend another), include (one use case can include another) and realize (this use case may realize some business requirement)

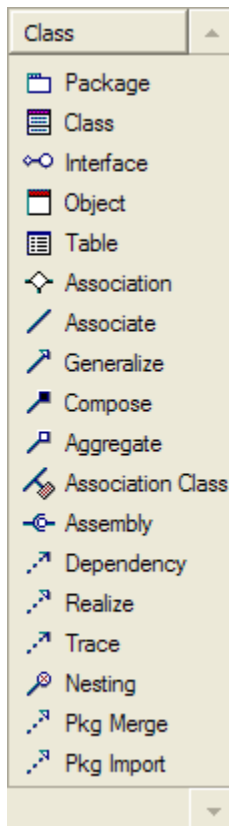
To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

Note: Invokes and Precedes are defined by the Open Modeling Language (OML).

4.12.4 Class Group

The *Class* group can be used for [Package diagrams](#), [Class diagrams](#) and [Object diagrams](#) - those which usually display elements concerned with the logical structure of the system. These include objects, classes and interfaces. Logical models may include domain models (high level business driven object model) to strict development class models (define inheritance, attributes, operations, etc).



The Class group is used for creating class models and database models. Class modeling is done using the Class and Interface elements, as well as occasional use of the Object element to model class instances. You can add association, inheritance or aggregation relationships. See the Class Diagram for an example of this.

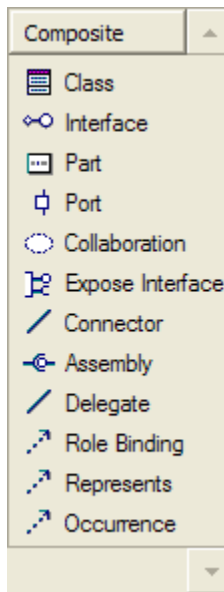
Use the Table element to insert a stereotyped class for use in database modeling. See the section on data modeling for more details.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.5 Composite Group

The *Composite* group is used for [Composite Structure diagrams](#). Composite Structure diagrams reflect the internal collaboration of classes, interfaces, or components to describe a functionality, and to express run-time architectures, usage patterns, and the participating elements' relationships, which static diagrams may not show.

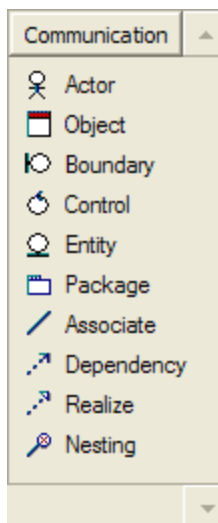


To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.6 Communication Group

The *Communication* group is used to model dynamic interactions between elements at run-time. The actor element models a user of the system, while the other elements model things within the system - including standard elements (rectangular element), user interface component (circle with left positioned vertical bar), controller (circle with arrow head in top most position) and entity (circle with bar at bottom).



[Communication diagrams](#) are used to model work flow and sequential passing of messages between elements in real time. They are often placed beneath Use Case elements to further expand on use case behavior over time.

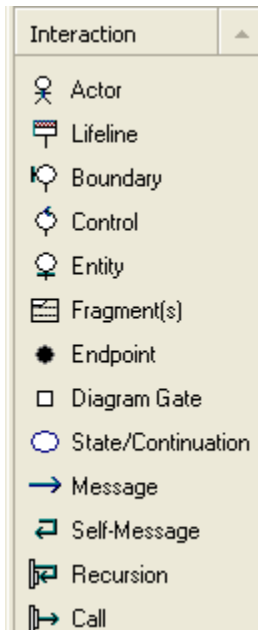
To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

Note: *Communication diagrams* were known as *Collaboration diagrams* in UML 1.4.

4.12.7 Interaction Group

The **Interaction** group is used for [Interaction diagrams](#), which are used to model work flow and sequential passing of messages between elements in real time. They are often placed beneath Use Case elements to further expand on use case behavior over time.



The Interaction group is used to model dynamic interactions between elements at run-time. The actor element models a user of the system, while the other elements model things within the system - including standard elements (rectangular element), user interface component (circle with left positioned vertical bar), controller (circle with arrow head in top most position) and entity (circle with bar at bottom). The meaning of these symbols is discussed further in the section on [Sequence diagrams](#). The message relation is used to model the flow of information and processing between elements.

The following model elements are supported: Actor (sequence element), element (sequence element), Boundary (sequence element), Control (sequence element), Entity (sequence element), Message and Iteration boundary.

Note: Messages may be simple or recursive calls.

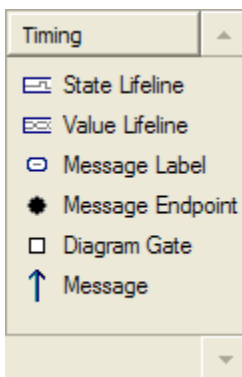
To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.8 Timing Group

The **Timing** group is used solely for [Timing diagrams](#), which use a time-scale to define the behavior of objects. The time-scale visualizes how the objects change state and interact over time. Timing diagrams can be used for defining hardware-driven or embedded software components, also time-driven business processes.

Timing diagrams can be used for defining hardware-driven or embedded software components, also time-driven business processes.



A lifeline is the path an object takes across a measure of time, indicated by the x-axis.

A [State Lifeline](#) follows discrete transitions between states, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations.

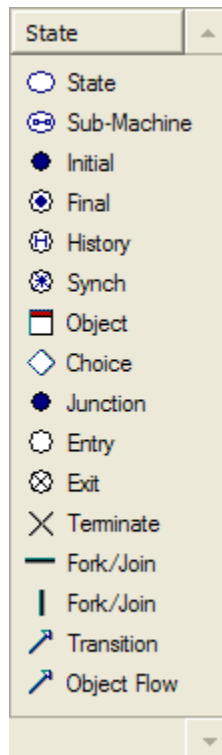
The [Value Lifeline](#) shows the lifeline's state across the diagram, within parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.9 State Group

The **State** group is used by [State Machine diagrams](#) to show the allowable states a class/element may be in and the transitions from one state to another. These diagrams are often placed under a class element in the Project Browser to illustrate how a particular element changes over time.



The State group provides elements common to State Machine diagrams - basically the State, start and end nodes and the Flow relation. State Machine diagrams are used to model the states or conditions that elements at runtime may be in - eg. active, inactive, idle, accelerating, braking, etc.

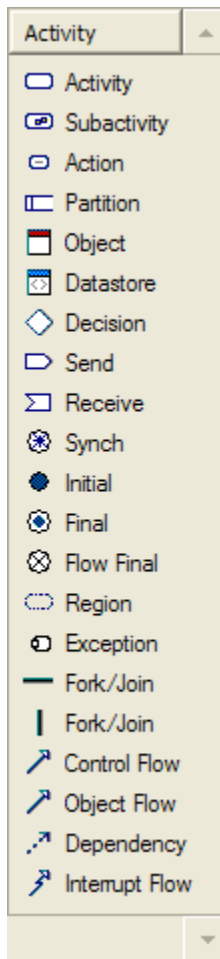
This group of icons allows you to add elements and relations to the current diagram. Click on the required element or connector to add to the current work. An element will be automatically inserted and ready for naming and properties to be set. A connector requires you to click on the start element and drag to the end element.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.10 Activity Group

The **Activity** group is used to model system dynamics from a number of viewpoints in [Activity diagrams](#) and [Interaction Overview diagrams](#). An activity is some work that is carried out - it may overlap several use cases or form only a part of one use case. Send and receive events are included as triggers. A decision element marks a point where processing may split based on some outcome or value. The Flow relation models an active transition and synch points are used to split and rejoined periods of parallel processing.



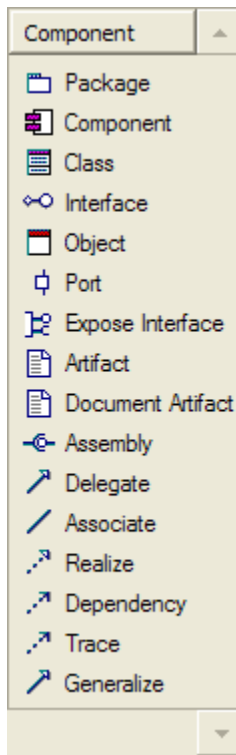
Activity elements allow you to describe the *dynamics* of the system from the point of view of *activities* and *flows* between them. Activities may be stereotypes as a "process" to display a business process icon.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.11 Component Group

The *Component* group allow you to model the physical components of your system in a [Component diagram](#). A component is a piece of hardware or software that makes up the system, for example, a DLL or Web Server are examples of Components that may be deployed on a Windows 2000 Server (Node). See the Deployment Diagram for an example of this.



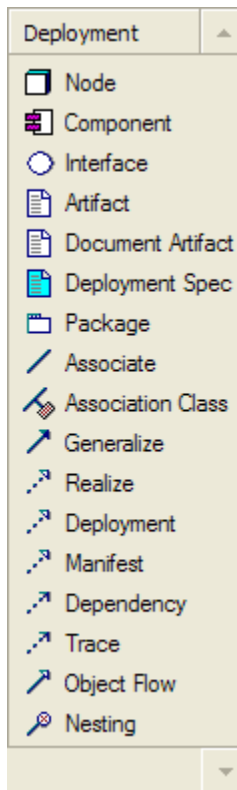
The Component group contains elements related to the actual building of the system – the components that will make up the system (e.g. ActiveX DLL's or Java beans), the Interfaces they expose and the dependencies between those elements.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.12 Deployment Group

The *Deployment* group allow you to model the physical components and deployment structure of your system in a [Deployment diagram](#). A component is a piece of hardware or software that makes up the system, and a Node is a physical platform on which the component will exist. For example, a DLL or Web Server are examples of Components that may be deployed on a Windows 2000 Server (Node). See the Deployment Diagram for an example of this



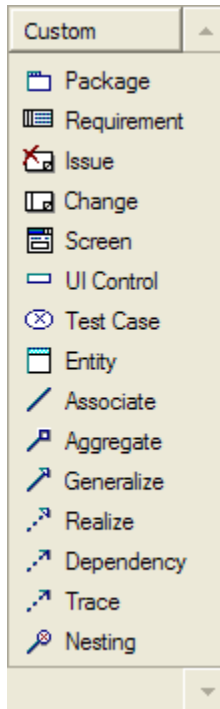
The Deployment group contains elements related to the actual building of the system – the components that will make up the system (e.g. ActiveX DLL's or Java beans) and the nodes those components will run on - including the physical connections between nodes.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.13 Custom Group

The *Custom* group contains a few extended UML elements that may be of use in modeling or designing your system in a [Custom diagram](#).



An **Entity** is a stereotyped element that represents any general thing not captured by the element or class type elements (for example a trading partner). Use of this element is deprecated: it was originally intended to take the role now occupied by a [Table](#) element.

A **Screen** provides a stereotyped class element that displays a GUI type screen – this can be used to express application GUI elements and flows between them.

A **UI control** likewise can be used to express GUI controls.

A **Requirement** is a custom element used to capture requirements outside of standard UML elements. A requirement expresses required system behavior that may cross several use cases. You may link requirements to other elements using the realize link to express the implementation of a requirement and hence the traceability from user needs to what is being built.

An **Issue** is a custom element used to capture model issues (such as defects and bugs).

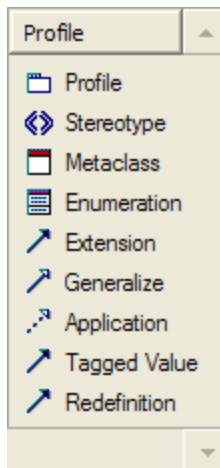
A **Change** is a custom element used to model change requirements

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.14 Profile Group

The **Profile** group contains some extended UML elements that may be of used to create and modify Profiles for the rapid creation of stereotyped and tagged values that may be applied to elements, attributes, methods, links, etc..



A **Profile** is used to provide a generic extension mechanism for building UML models in particular domains. They are based on additional Stereotypes and Tagged values that are applied to Elements, Attributes, Methods, Links, Link Ends, etc.

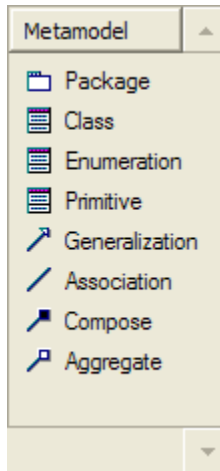
A **Stereotype** provides a mechanism for varying the behavior and type of a model element.

A **Metaclass** is used to create a class whose instances are classes, a metaclass is typically used to construct metamodels.

An **Enumeration** creates a class stereotyped as enumeration, which is used to provide a list of named values as the range of a particular type.

4.12.15 Metamodel Group

The Metamodel group gives you the ability to create metamodel diagrams with support for MOF diagrams.



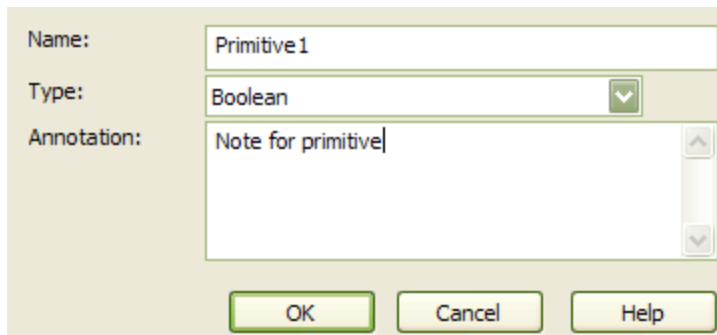
A [Package](#) is a namespace as well as an element that can be contained in other package's namespaces. (see [Package](#))

A [Class](#) is a representation of object(s), that reflects their structure and behavior within the system.

An [Enumeration](#) is a class with an enumeration stereotype.

A [Primitive](#) is new for EA to support the MOF specification.

Primitive New in EA for MOF diagrams. By dragging and dropping a Primitive from the Metamodel section of the toolbox, the following dialogue will appear.

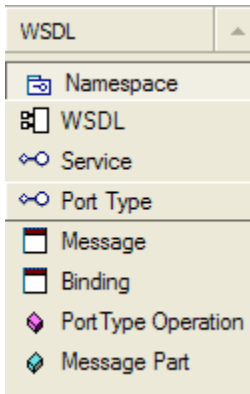


The following types are available. These types are defined in the MOF specification.

- Boolean
- Integer
- Long
- Float
- Double
- String

4.12.16 WSDL Group

The WSDL group gives you the ability to rapidly [model](#) and automatically [generate](#) W3C Web Service Definition Language (WSDL) documents.



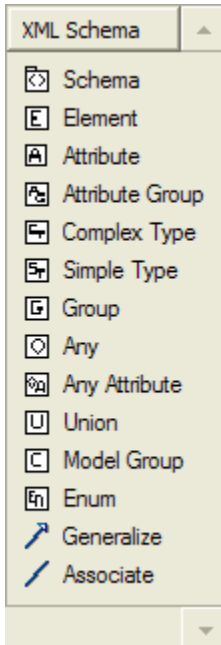
A *Namespace* represents the top-level container for the WSDL model. Drag this element onto an open diagram to create the necessary model structure for WSDL documents.

A physical *WSDL* document is represented as an UML component. Its interfaces represent the WSDL *services*.

A WSDL *Port Type* is modeled as an UML interface. Its *Port Type Operations* are realized by *Binding* elements. Each of the operation parameters are derived from the *Message* classes defined in the Messages package.

4.12.17 XML Schema Group

The XML Schema group provides the ability to [model](#) and automatically [generate](#) W3C XSD schema files. This group implements the constructs provided by the [UML profile for XML Schema](#).



A *Schema* corresponds to an UML package, which contains the type and element definitions for a particular targetNamespace. Drag this item onto to an open diagram to create the package to contain your schema model elements. The package will be stereotyped as XSDschema.

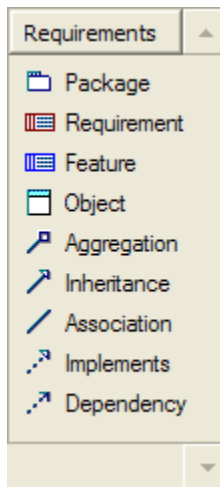
Open the logical diagram created under the XSDschema package and add additional schema elements as required.

To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set the name and other properties as prompted.

To add a relationship, click the required icon, then click on the start element in the diagram and drag to the end element.

4.12.18 Requirement Group

As an analysis step, often it is desirable to capture simple system requirements. These will eventually be realized by use cases. (see [Requirements](#))



A **Package** is a namespace as well as an element that can be contained in other package's namespaces. (see [Package](#))

Specify the **Requirement** of a system. Note, there are a few different requirement types as listed below.

- Display
- Functional
- Performance
- Printing
- Report
- Testing
- Validate

You also have the ability to create your own. To do this, right click on the Requirement class and select **Properties**. Type the name of the requirement into the **Type** field.

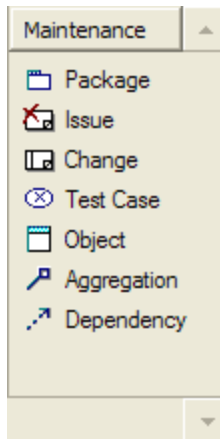
(see [Requirements](#))

A **Feature** is a small client-valued function expressed as a requirement. Features are the primary requirements-gathering artifact of the Feature-Driven Design (FDD) methodology.

An **Object** is an instance of a class. (see [Object](#)).

4.12.19 Maintenance Group

The maintenance elements are defects, changes, issues and tasks. (see [Maintenance](#)).



A **Package** is a namespace as well as an element that can be contained in other package's namespaces. (see [Package](#))

An **Issue** element is a structured comment which contains information about defects and issues that relate to the system/model. (see [Defects Issues](#))

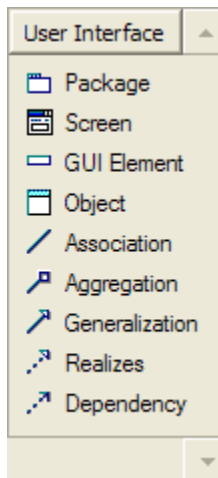
A **Change** element is a structured comment which contains information about changes that have been requested to the system/model. (see [Changes](#))

A **Test Case** describes what is needed to be setup in order to test a particular feature. (see [Package](#))

An **Object** is an instance of a class. (see [Object](#)).

4.12.20 User Interface Group

The User Interface Group enables you to create graphical user interface diagrams.



A **Package** is a namespace as well as an element that can be contained in other package's namespaces. (see [Package](#))

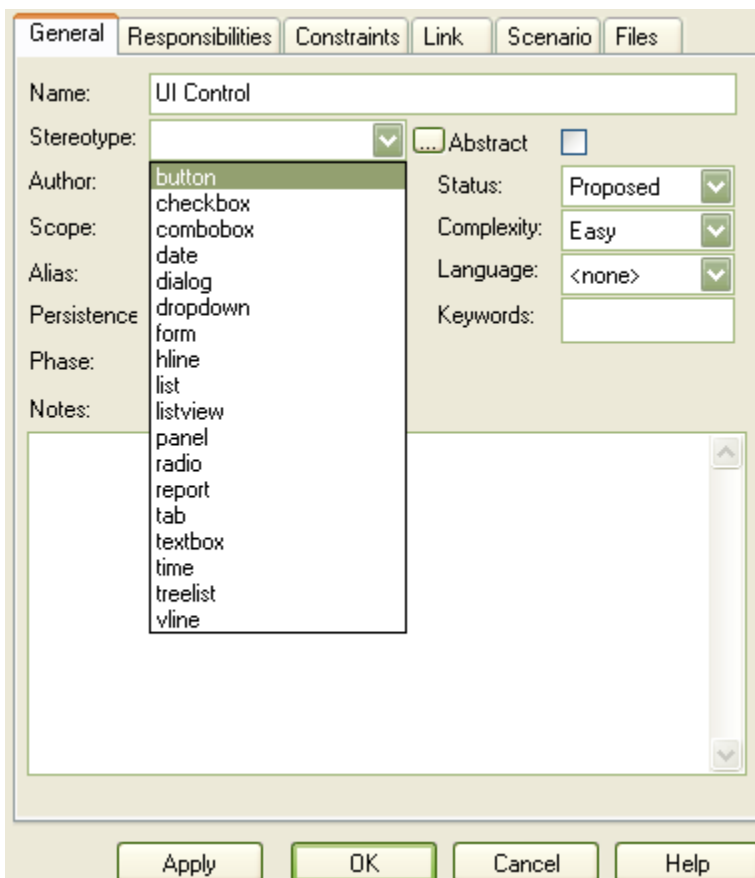
A **Screen** element represents a graphical user interface. You will be able to place GUI elements onto the screen element.

A **GUI Elements** are placed onto the screen element to build up a graphical user interface diagram. There are different stereotypes that represent different elements such as buttons and combo box's etc.

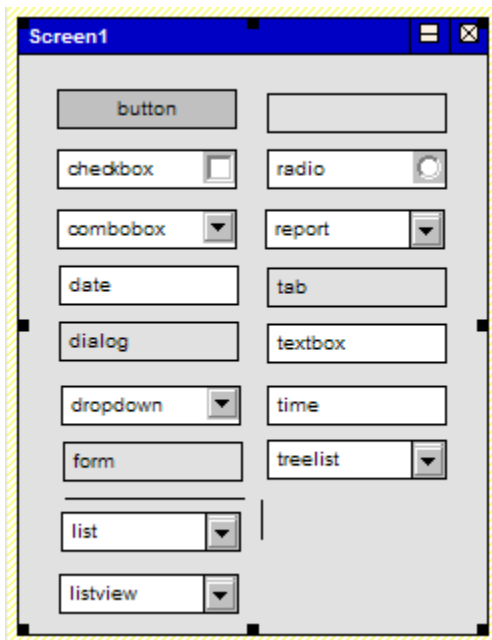
An **Object** is an instance of a class. (see [Object](#)).

GUI Element Stereotype available

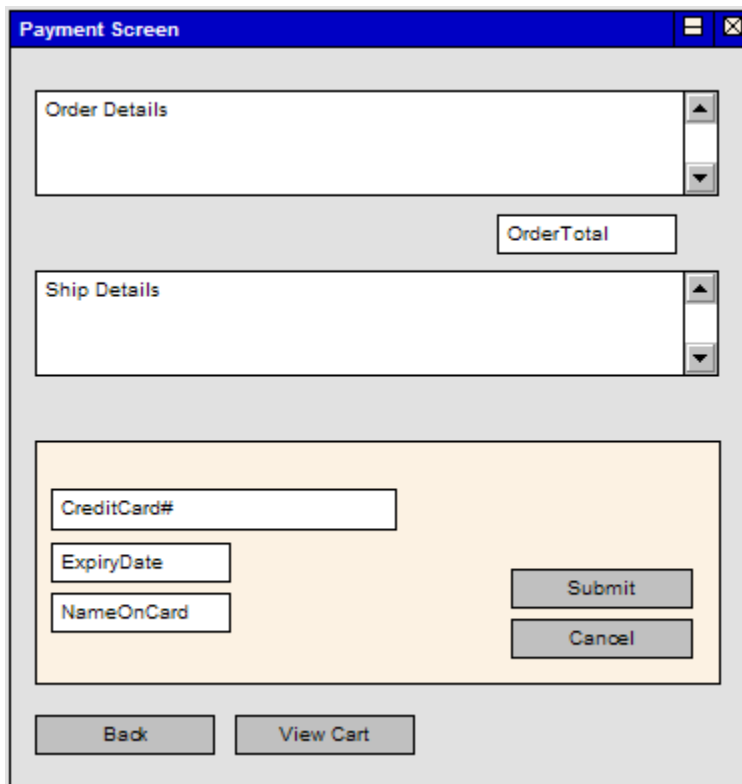
The following GUI elements are available as stereotypes.



The various elements as they appear on a screen element.

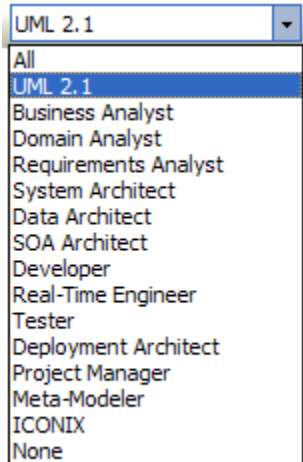


Example GUI Diagram



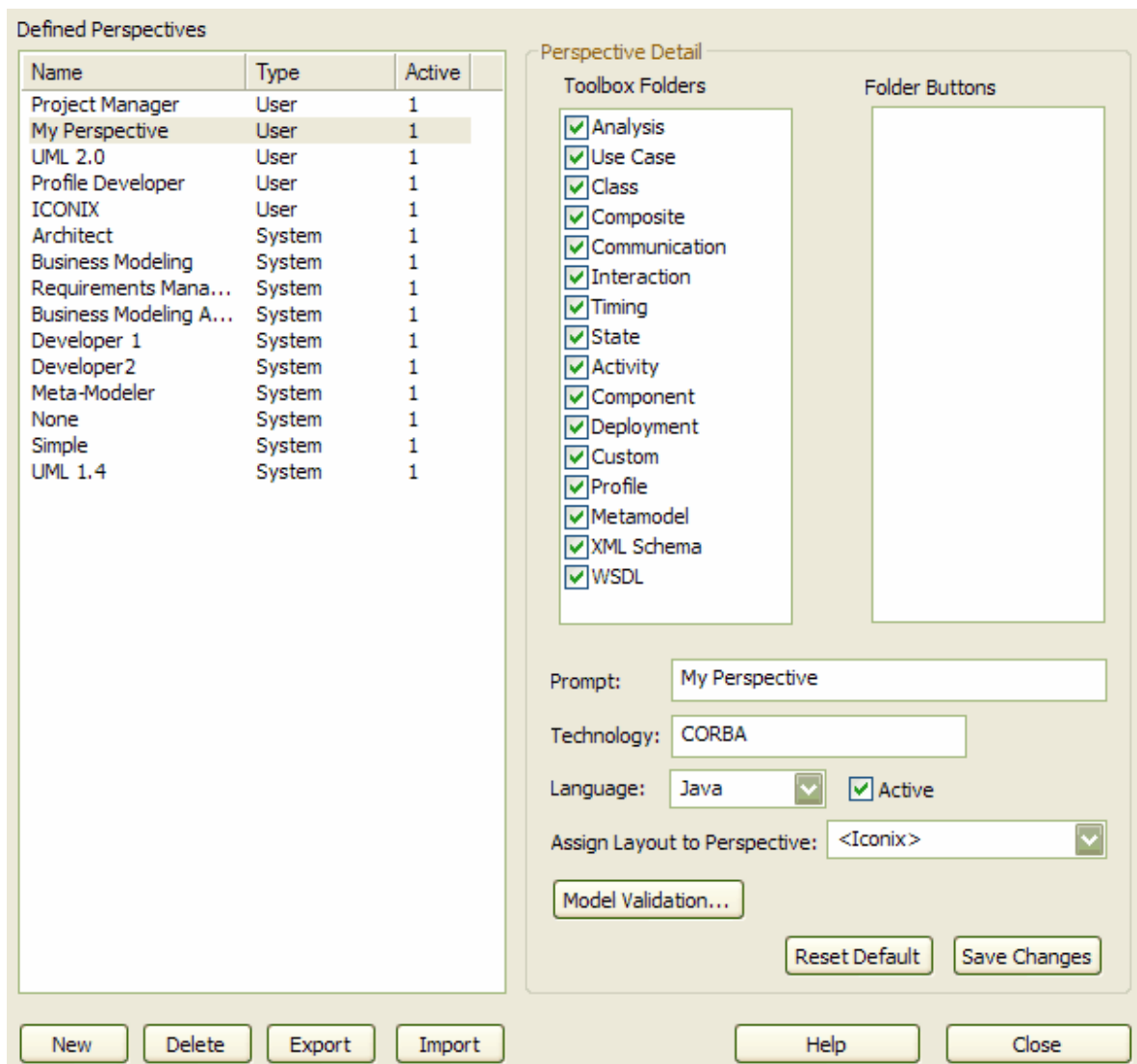
4.12.21 Perspectives

The Workspace perspective tool allows the users of EA to alter the contents of the UML Toolbox to suit a particular modeling role. For example selecting the Code Engineering Basic option will cause the UML toolbox to only include the basic tools used in code engineering and removes the tools which do not apply to code engineering. The Workspace Perspective is selected from the [UML Elements Toolbar](#) or from the dropdown list at the top of the [UML ToolBox](#).



4.12.21.1 Configure Perspectives

Workspace perspectives may be configured to enable users of Enterprise Architect to customize the UML toolbox. The default perspectives may be modified to provide a customized modeling environment. New perspectives may be created as needed. In order to configure *Perspectives*, you need to access the *Configure Perspective* dialog from the main menu **View | Configure Perspectives...**. Alternatively right click on any blank area of the [UML Toolbox](#) and select **Configure Perspectives**.



You may edit any perspective by simply selecting it from the *Defined Perspectives* list and checking off box's within the *Toolbox Folders* window. By selecting a toolbox folder in that window, you will see all the buttons associated with that folder in the *Folder Buttons* window. By checking off box's within the *Folder Buttons* window, you can customize what buttons will appear for that toolbox folder. Don't forget to press **Save Changes** after making any change.

You may wish to create a new perspective, to do this, simply press the **New** button and enter a name for the perspective and press **OK**. Your perspective will now appear in the *Defined Perspectives* list and you may now edit it.

| Control | Description |
|---------|---|
| New | Create New Defined Perspectives |
| Delete | Delete the selected Perspective. |
| Export | Export the Perspective to a .XML file for use with other workstations. |
| Import | Import a previously created Perspective for use with the current workstation. |

| | |
|------------------|---|
| Toolbox Folders | Shows a list of the categories of tools available in the UML Toolbox. |
| Folder Buttons | Shows the selected UML toolbox items that will be included with selected category of the UML toolbox. |
| Prompt | Note for describing the function of the Perspective |
| Language | Select the default code language from the list of available code languages. |
| Technology | Select a Technology from the resource view. |
| Active | The Active check box is used to activate the perspective in the user interface, this setting is enabled by default, use this setting to hide/unhide the perspective from the user interface. |
| Model Validation | Assigns a set of Model Validation Rules from the list of available rules on the Model Validation Configuration Dialog. For more information regarding Model Validation , please see the Model Validation topic. |
| Reset Default | Resets the perspective to its default settings. |
| Save Changes | Save the changes made to the perspectives. |
| Help | Opens the help menu associated with the configuration of perspectives. |
| Close | Close the dialog |

4.12.21.2 Import Export Perspectives

You have the ability to save and load an Individual perspective as an XML file. Simply select the perspective from the *Defined Perspectives* list and press the **Export** button in order to save the selected perspective, press the **Import** button in order to load a perspective.

Note: If you make any changes to a perspective, you must save those changes first before exporting otherwise those changes won't be saved.

Note: Only individual perspective are saved, not the entire *Defined Perspectives* list.

4.13 Package Tasks

A *Package* is a container of model elements, and is displayed in the Project Browser using the 'folder icon' familiar to Windows users. This section explores the tasks you can perform with packages, including:

- [Open a package](#)
- [Add a package](#)
- [Rename a package](#)
- [Drag a package onto a diagram](#)
- [Show or hide a package](#)
- [Delete a package](#)

See Also

- [The Project View Browser](#)

4.13.1 Open a Package

Open a Package

A *Package* is a container of model elements, and is displayed in the Project Browser using the 'folder' icon familiar to Windows users.

To open a package:

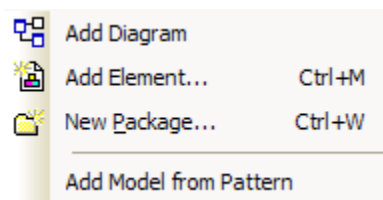
- Double clicking on a *package* will open it and display any contents in the [Project Browser](#) (or tree).
- Alternatively, you can click once on the + and - symbols next to the folder icon to open or close the package respectively.

Tip: Package contents are arranged alphabetically and elements may be dragged from one folder to another using the mouse.

4.13.2 Adding Packages

To add a new package

1. Select the package or view in the [Project Browser](#) under which you wish to add a new package.
2. Right click the mouse on the folder icon within the Project Browser to open the context menu.
3. Select **Add | Add Package...**



4. In the dialog box, fill in the name of the new package and press **OK**.

The new package will be inserted into the tree at the current location.

Note: A package may also be added using the UML Toolbox - and pasting a new package element into a diagram. In that case the package is created under the diagram's owning package, and will be created with a default diagram of the same type as that the Package is created in.

Tip: Note that in a multi-user environment, other users will not see the change until they reload their project.

4.13.3 Renaming Packages

To rename a package

1. Select the package you wish to rename in the [Project Browser](#).
2. Right click to open the context menu.
3. Select the **Package Properties** option.
4. Enter the new name in the **Name** field.
5. Press **OK**.

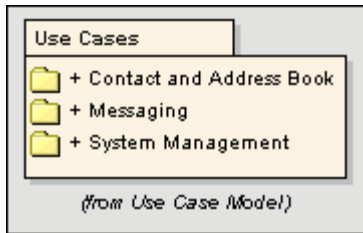
Alternatively, highlight the package you wish to rename, and press **F2**

Note: In a multi-user environment, other users will not see the change until they reload their project.

4.13.4 Drag a Package onto a Diagram

You can drag a package element from the Project Browser onto the current diagram. This will display the package and any contents within. This is a useful feature to help organise the display and documentation of models.

The illustration below displays how a package will be displayed in a diagram - note the Actor and child package icons.



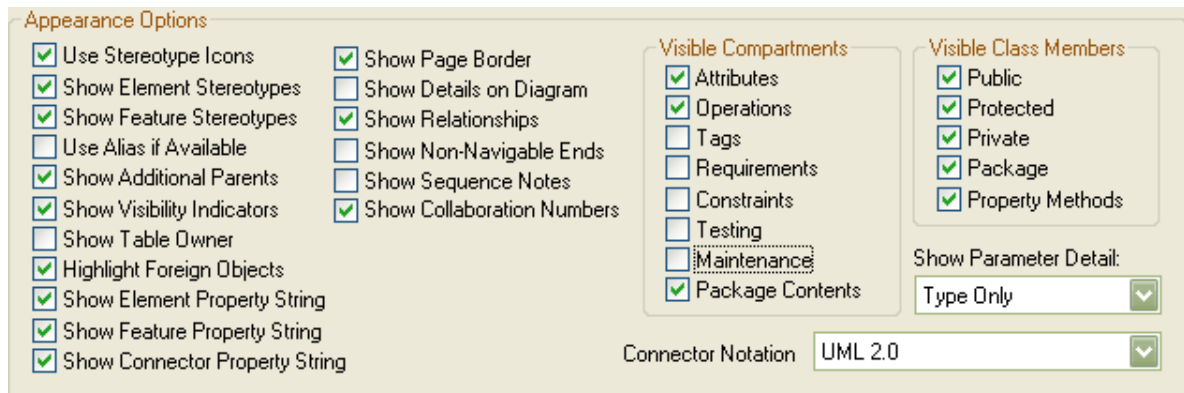
See Also

- [Show or Hide Package Contents](#)

4.13.5 Show or Hide Package Contents

You can show or hide the contents of packages in a diagram. To do so, follow these steps:

1. Load a diagram.
2. Double click in the background area to open the diagram properties window.
3. Check or clear the *Show Package Contents* box as required.
4. Press *OK*.



4.13.6 Delete a Package

You can delete a model element by highlighting it in the [Project Browser](#), then right clicking to open the context menu.

Select the *Delete* option. You will be asked to confirm your request – press *OK* to proceed.

Warning: *Deleting a package deletes all contents of the package as well, including sub-packages and elements. Make very sure that you really want to do this before proceeding.*

Note: *In a multi-user environment, other users will not see the change until they reload their project.*

4.14 Diagram Tasks

This section explores the tasks you can perform with diagrams, including:

- [Select a diagram](#)
- [Add a diagram](#)
- [Zoom a diagram view](#)
- [View last and next diagram](#)
- [Set diagram page size](#)
- [Scale image to page size](#)
- [Lock diagram](#)
- [Show or hide attributes and operations](#)
- [Layout a Diagram](#)
- [Set appearance options](#)
- [Undo last action](#)

4.14.1 Add a Diagram

To add a new diagram, follow these steps:

1. In the *Project Browser* window, select the appropriate package or element under which you wish to place the diagram.
2. Right click to open the context menu and select *Add | Add Diagram*. Alternatively, select *Add Diagram* from the *Project* menu
3. Enter the name of the new diagram in the dialog provided and select the type of diagram you require.
4. Press *OK* to create your new diagram.

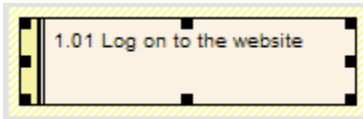
Note: *The type of diagram type determines the default toolbar associated with the diagram and whether it can be set as a child of another element in the Project Browser (eg. a sequence diagram under a use case)*

See Also

- [UML Diagrams](#)

4.14.2 Highlight Context Element

You can show a hatched border around the selected element by checking the *Always Highlight Context Element* checkbox on the *Tools | Options | Diagram | Behavior* page. If you have this option checked, when you select an element it will appear like so:

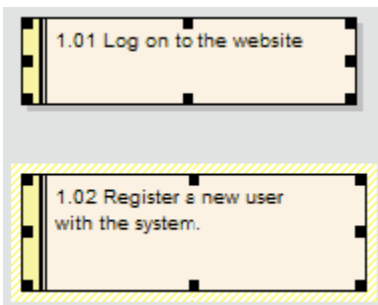


If you do not have this option checked, the selected element will not have a hatched border around it.

Multiple Selections

Whether you have the *Always Highlight Context Element* option selected or not, when you select multiple elements, one of the elements you select will have a hatched border. If you align the elements, this element will be the one used to align the other elements against.

For example, if the elements in the diagram below were to be aligned, the top element would align to the bottom element (the element showing a hatched border).



Changing the Element to Align Against

To change which element has a hatched border in a selected group - thus the element that will be aligned against - simply click on the element you want the other elements to align against.

4.14.3 Layout a Diagram

A facility is provided to layout diagrams automatically. This will attempt to create a reasonable tree based structure from the diagram elements and relationships in a diagram. Owing to the complexity of many diagrams, the results may need some manual 'tweaking'.

Layout a Diagram

To layout a diagram, follow the steps below:

1. Select a diagram.
2. From the *Diagram* menu, select *Layout Diagram* -OR- use the *Auto Layout* button on the diagram toolbar .

Note: *Dynamic and analysis diagrams are NOT suited to this form of layout - please ensure that the diagram type you are laying out will benefit from the action first.*

Accessing the Diagram Layout Options Dialog

For a fine degree of control of the elements in your diagram, you can use the *Diagram Layout Options* dialog. Generally the default layout parameters provide adequate layouts for a wide range of diagrams, but there are times when more specific settings are required. To access the *Diagram Layout Options* dialog, follow the steps below:

1. Open the diagram's *Properties* dialog by double clicking on the background of the diagram.

2. Press *Set Layout Style* to open the *Diagram Layout Options* dialog.
3. When you have made any required changes, press *OK* to save.

The screenshot shows the 'Diagram Layout Options' dialog box with the following settings:

- Cycle Remove Options:** Greedy (unselected), Depth First Search (selected).
- Crossing Reduction Options:** Iterations: 4, Aggressive (unchecked).
- Layering Options:** Longest Path Sink (unselected), Longest Path Source (unselected), Optimal Link Length (selected).
- Layout Options:** Spacing: Layer Spacing: 20, Column Spacing: 20.
- Initialize Options:** Naive (unselected), Depth First Search Outward (selected), Depth First Search Inward (unselected).
- Direction:** Up (selected), Left (unselected), Down (unselected), Right (unselected).
- Set as Project Default:** (unchecked).

You can alter any of these settings on the *Diagram Layout Options* dialog to refine your layout:

Cycle Remove Options

These settings determine the type of cycle removal to be used during layout.

Greedy - Uses the Greedy Cycle Removal algorithm.

Depth First Search - Uses the Depth First Search Cycle Removal algorithm.

Layering Options

These settings determine the type of layering to be used during layout.

Longest Path Sink - Uses the Longest Path Sink Layering algorithm.

Longest Path Source - Uses the Longest Path Source Layering algorithm.

Optimal Link Length - Uses the Optimal Link Length Layering algorithm.

Initialize Options

These settings determine the type of initialization of indices and columns to be used during layout.

Naive - Uses the Naive Initialize Indices algorithm.

Depth First Search Outward - Uses the Depth First Out Initialize Indices algorithm.

Depth First Search Inward - Uses the Depth First In Initialize Indices algorithm.

Crossing Reduction Options

Iterations - Enter the number of iterations to be used during cycle removal.

Aggressive - This option allows you to specify whether or not to use an aggressive (read time-consuming) crossing reduction step. If it is checked, the aggressive crossing reduction will be used. If it is unchecked, the aggressive crossing reduction will not be used.

Layout Options

Layer Spacing - Enter the default number of logical units between layers.

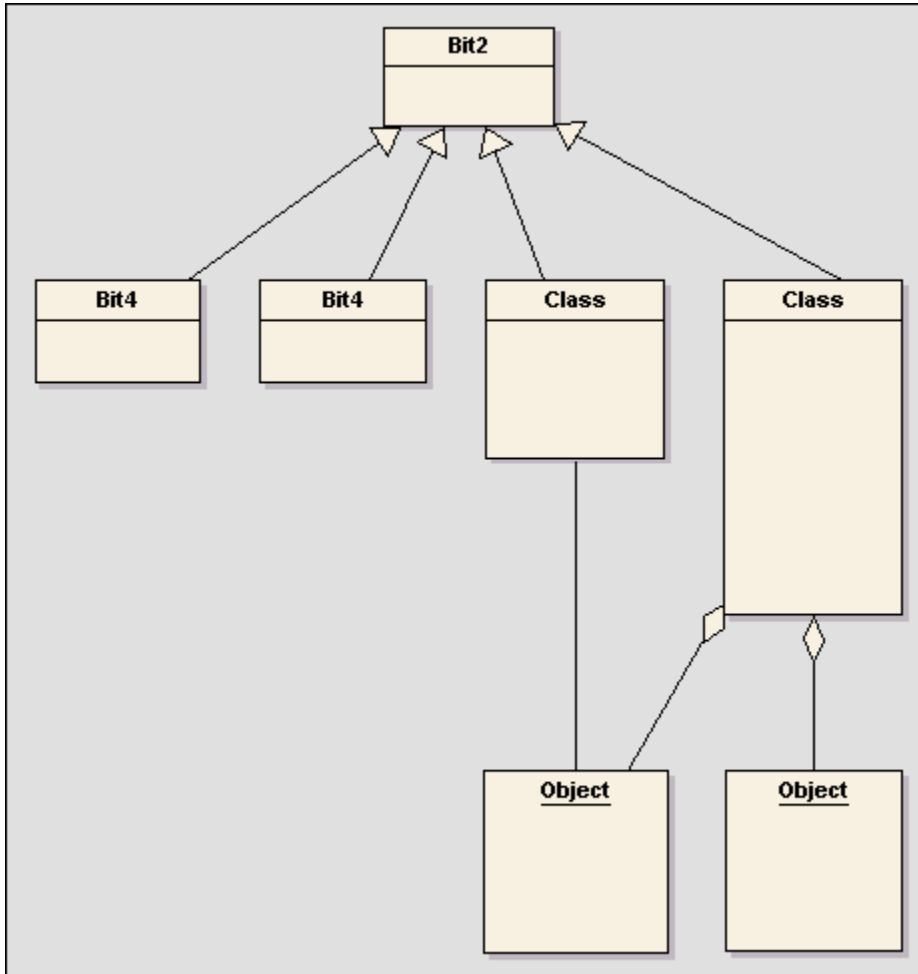
Column Spacing - Enter the default number of logical units between columns.

Direction - Set the direction that directed links should point - Up, Down, Right, Left.

Set as Project Default Checkbox

Check this to apply the Diagram Layout settings to all diagrams in the project. If you then check this box and press OK for a different diagram, the new settings will override the settings saved earlier.

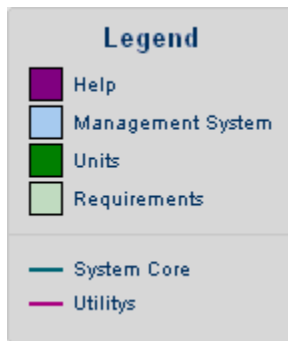
An example of an automatically laid out diagram:



4.14.4 Legends

Adding a Legend

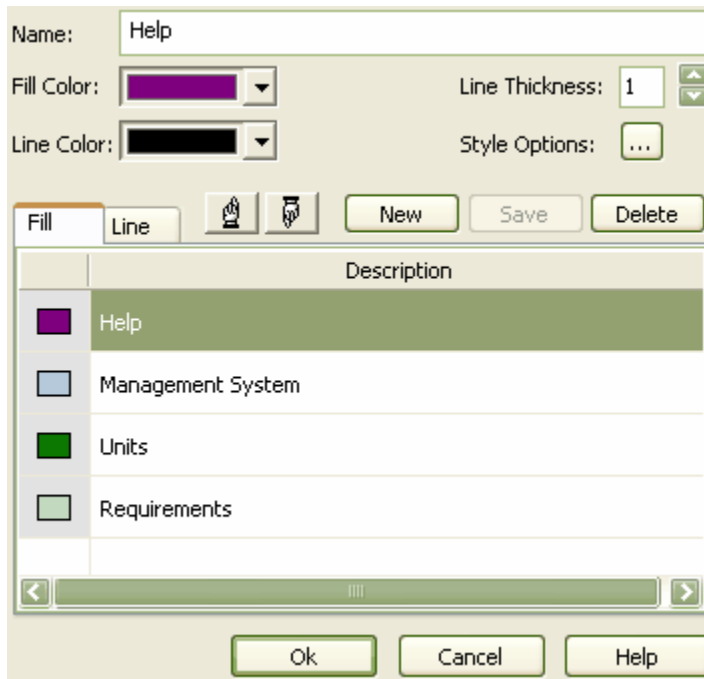
- *Right click* on a diagram to open the context menu, then select *Create Element or Connector | Common Elements | Diagram Legend*
- By clicking *New Diagram Legend* on the *UML Elements toolbar*.



Editing a Legend

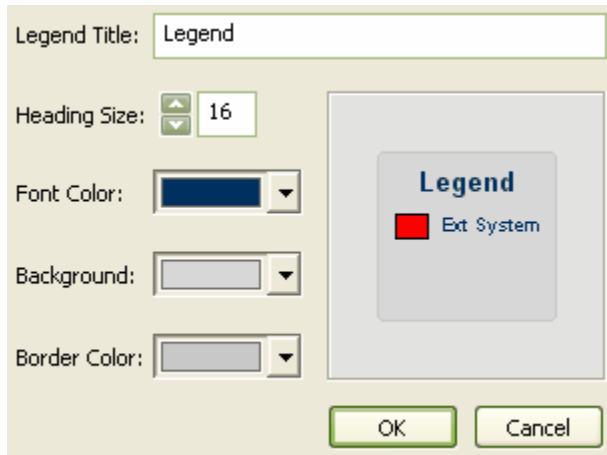
- *Double click* the Legend.
- *Right Click* on the Legend and select *Properties*.

The properties window allows you to add, delete, modify or position your legends. There are two types of legends provided, Fill and Line types. You can use the legend element to assist in distinguishing different elements, connectors, systems, etc. on a diagram.



Style Options

By clicking the Style Options button, you are able to modify a legends title, title size, background font and border color. If you choose default for the colors, the legend will automatically choose its color based on the diagram background color.

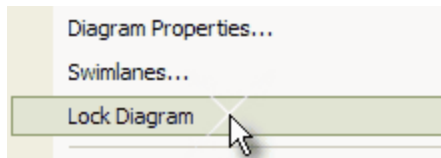


4.14.5 Lock Diagram

You can lock a diagram against inadvertent changes - moving or sizing elements, for example.

To lock a diagram, follow these steps:

1. Open the diagram you wish to lock.
2. Right click on the background to open the diagram context menu.
3. Click the *Lock Diagram* option to prevent further changes.



4. Press *OK*.

4.14.6 Pan a Diagram

Panning the Diagram View can be achieved by the following methods:

- Using the Directional, Page Up/Page Down and Home/End Keys when the Diagram view is selected
- Using the Scrollbars
- Using the Middle Mouse Button to Pan
- Using the [Pan & Zoom](#) Window

4.14.7 Scale Image to Page Size

You may scale a diagram image to fit the currently selected page size, scale the diagram image to fit the current page size or define a custom scaling layout.

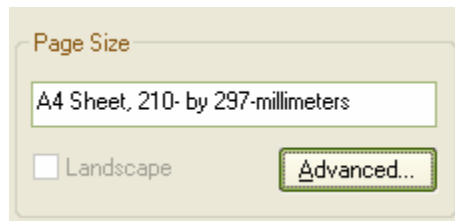
The default diagram image setting is to fit the size of the currently selected printer paper. The image will not be scaled up to fill the page, but will be scaled down if it exceeds the current page boundary. Also, the image will retain its current dimensions - that is it will be scaled down equally in the X and Y dimensions.

Scale image option

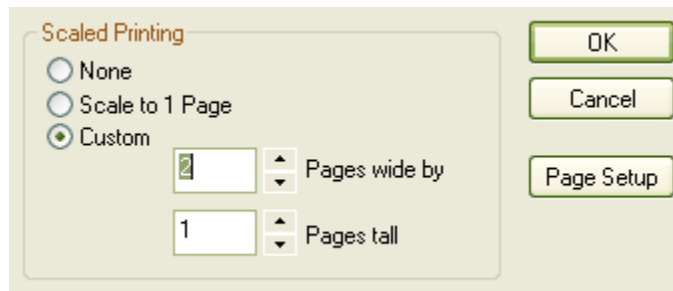
To access the image scaling options, follow these steps:

1. Select the diagram to scale.

2. Double click on the diagram background to get the properties dialog, or select the *properties* option from the context menu.



3. Under the *Page Size* Heading press the *Advanced* button.

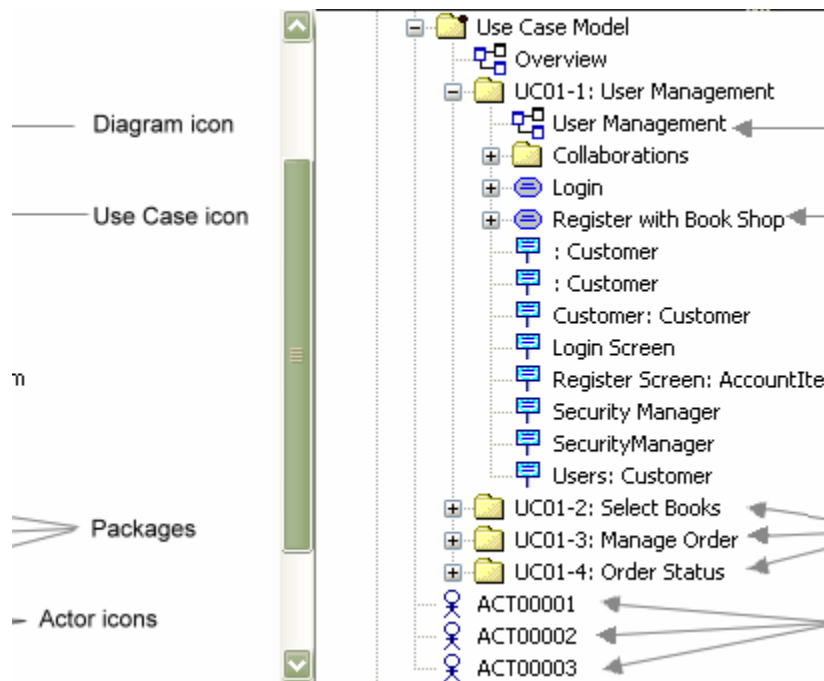


4. From the Print Advanced Dialog the following options are available:
 - *None*, the none option prints on as many pages as the diagram image covers.
 - *Scale to 1 page*, this option scales the diagram image to fit on the currently selected page.
 - *Custom*, the custom setting allows the user to specify the width and height of the diagram images across a specified amount of pages
5. *Page Setup* allows the user to select the Page size and alignment.

Note: Before printing, make sure you have selected the required Page Layout from the Page Setup button.

4.14.8 Select a Diagram

Diagrams are used to display model elements (elements, use cases, components, etc.).



To open a diagram that already exists, follow these steps:

1. Search the *Project Browser* until you locate the diagram you require. You may need to explore several levels before you find your diagram.
2. Double click on the *diagram icon*.
3. Enterprise Architect will open the diagram and display any model elements in the diagram view.

4.14.9 Set Appearance Options

You can customize the appearance of a diagram in a number of ways. The following options can be set on the diagram dialog:

Appearance Options

| | | | |
|--|--|--|--|
| <input checked="" type="checkbox"/> Use Stereotype Icons | <input checked="" type="checkbox"/> Show Page Border | Visible Compartments | Visible Class Members |
| <input checked="" type="checkbox"/> Show Element Stereotypes | <input type="checkbox"/> Show Details on Diagram | | |
| <input checked="" type="checkbox"/> Show Feature Stereotypes | <input checked="" type="checkbox"/> Show Relationships | <input type="checkbox"/> Operations | <input checked="" type="checkbox"/> Protected |
| <input type="checkbox"/> Use Alias if Available | <input type="checkbox"/> Show Non-Navigable Ends | <input type="checkbox"/> Tags | <input type="checkbox"/> Private |
| <input checked="" type="checkbox"/> Show Additional Parents | <input type="checkbox"/> Show Sequence Notes | <input type="checkbox"/> Requirements | <input checked="" type="checkbox"/> Package |
| <input checked="" type="checkbox"/> Show Visibility Indicators | <input checked="" type="checkbox"/> Show Collaboration Numbers | <input type="checkbox"/> Constraints | <input checked="" type="checkbox"/> Property Methods |
| <input type="checkbox"/> Show Table Owner | | <input type="checkbox"/> Testing | |
| <input checked="" type="checkbox"/> Highlight Foreign Objects | | <input type="checkbox"/> Maintenance | |
| <input checked="" type="checkbox"/> Show Element Property String | | <input checked="" type="checkbox"/> Package Contents | |
| <input checked="" type="checkbox"/> Show Feature Property String | | | Show Parameter Detail: Type Only |
| <input checked="" type="checkbox"/> Show Connector Property String | | | |

Connector Notation: UML 2.0

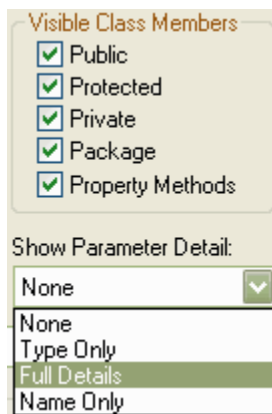
The specific options available in the Appearance Options are:

| Option | Description |
|--------|-------------|
|--------|-------------|

| | |
|--------------------------------|---|
| Use Stereotype Icons | Toggle the display of stereotype icons in the current diagram. This only applies to stereotypes with icons internal to EA, i.e. Analysis stereotypes and Business Modeling stereotypes . |
| Show Element Stereotypes | Show the stereotypes on all elements |
| Show Feature Stereotypes | Show the stereotypes on all features |
| Use Alias if Available | Use element alias as name if the alias is specified. |
| Show Additional Parents | Show the name of all parents not in the current diagram for all classes and interfaces. |
| Show Visibility Indicators | Hide the visibility indicators on the diagram. |
| Show Table Owner | Displays the Table Owner, more information relating to this Topic may be found in the Set Table Owner topic. |
| Highlight Foreign Objects | Highlight objects from other packages. |
| Show Element Property String | Show the advanced property string for all elements. eg. {leaf} |
| Show Feature Property String | Show the advanced property string for all element features. eg. {readOnly} |
| Show Connector Property String | Show the advanced property string for all connectors. eg. {ordered} |
| Show Page Border | Show a page border to align elements with |
| Show Details on Diagram | Show some diagram details in a note on diagram. |
| Show Relationships | Show relationships in the current diagram. |
| Show Non-Navigable Ends | If an association end is not navigable then a cross will be presented at the association connector. |
| Show Sequence Notes | Show the Sequence Notes on the current diagram. |
| Show Collaboration Numbers | Show numbering in collaboration diagrams. |
| Visible Compartments | Allows the following compartments to be shown or hidden for any element using rectangle notation. <ul style="list-style-type: none"> • Attributes (See the Show or Hide Attributes and Operations topic.) • Operations (See the Show or Hide Attributes and Operations topic.) • Tags • Requirements • Constraints • Testing (See the Show Testing Scripts topic.) • Maintenance (See the Show Maintenance Scripts topic.) |
| Package Contents | Show icons for the contents of packages displayed in the current diagram. |
| Connector Notation | Three options exist for Connector Notation: <ul style="list-style-type: none"> • UML 2.0 - uses the standard UML 2.0 notation for connectors. • Information Engineering - uses the Information Engineering (IE) connection style, for more information please see the http://www.agiledata.org/essays/dataModeling101 page. • IDEFX1 - uses the Integrated DEFinition Methods IDEFX1 connection style, for more information please see the http://www.idef.com/IDEF1X.html page. |

4.14.9.1 Visible Class Members

The visible Class Members section of the diagram properties dialog Appearance Options allows you to hide class members by their scope and methods that specify properties. Use the checkboxes to select the visibility of class members.



Show Parameter Detail

The Show Parameter Detail drop down allows the user to control the display of method parameters with the following options:

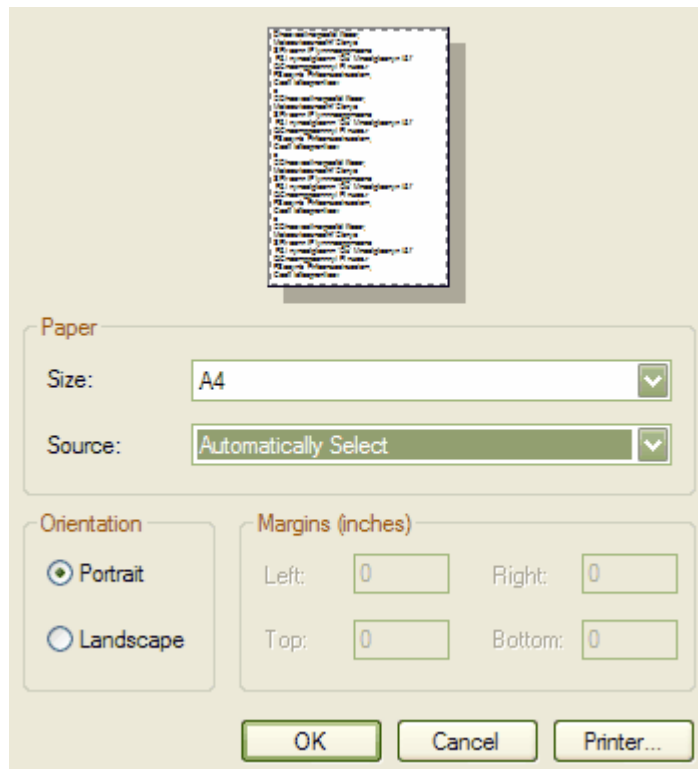
| | |
|--------------|---|
| None | No details shown |
| Type Only | Shows only the type of parameter. |
| Full Details | Shows all of the details for parameters |
| Name Only | Shows the name of the parameter only. |

4.14.10 Set Diagram Page Size

You can change the size of the diagram area (or scrollable/printable area) using the diagram properties window.

To set the page size, follow these steps:

1. Load a diagram.
2. Double click on the background to open the properties dialog.
3. In the Page Size section press the *Advanced* button and from the *Print Advanced* dialog, press the *Page Setup* button.
4. Select an appropriate page size from the drop-down list and then choose the appropriate Orientation of the page that you intend to print.
5. Press *OK*.



The scrollable area for your diagram will be expanded or reduced accordingly. Note that when you print or print preview, the output is cropped to the bounding rectangle for the diagram.

Setting the Default Paper Size for New Diagrams

You can set the default paper size for new diagrams on the View/Options/Diagram options dialog. Once set there, all new diagrams will have that as the default size.

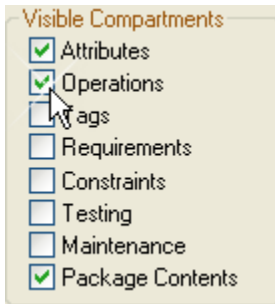
See [Diagram Settings](#).

4.14.11 Show or Hide Attributes and Operations

You can show or hide attributes and operations within classes on a diagram.

To show or hide attributes and operations, follow these steps:

1. Open a diagram.
2. Double click on the diagram background to open the properties dialog.
3. In the *Visible Compartments* section, check the *Attributes* and/or *Operations* as appropriate.



4. Press the **OK** button to confirm the selections.

4.14.12 Undo Last Action

When editing diagrams, Enterprise Architect supports multiple undo levels for moving, re-sizing and deletion of elements, and deletion of connectors.

There are three ways to undo the last action:

- Press **Ctrl-Z** OR
- Use the **Edit | Undo** menu option OR
- Use the **Undo** toolbar button (below)



Warning: Currently the following cannot be undone:

- Element additions cannot be undone.
- Connector moves cannot be undone.

4.14.13 View Last and Next Diagram

Enterprise Architect retains a list of recently visited diagrams. This is useful for stepping backwards and forwards through the list of recently accessed diagrams.



To view the previous or next diagram use the **Forward** or **Next** buttons on the diagram toolbar.

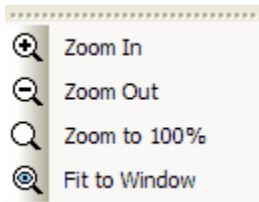
There is also a **Home** button which may be used to go to the [default project diagram](#) (if one has been specified)

4.14.14 Zoom a Diagram View


You can zoom into and out from a diagram view using the zoom buttons on the diagram toolbar, or by using the **Diagram | Zoom** submenu.



Change the zoom level by 10% by pressing either the **Zoom In (+)** or **Zoom Out (-)** buttons. The same is achieved by selecting **Zoom In** or **Zoom Out** from the **Diagram | Zoom** submenu.



There are three ways to return the diagram to 100%:

- Click the  button
- Select **Zoom to 100%** from the *Diagram | Zoom* submenu
- **Ctrl-Middle-Click** with the mouse

Tips & Tricks: The main window can be zoomed in and out dynamically by holding the CTRL key and using the mouse wheel.

Note: At high levels of zoom, element features will cease to display. This is because of the difficulty the Windows font mapper has in choosing a font for extreme conditions, and the result can look odd.

4.15 Element Tasks

This section explores the tasks you can perform with elements, including:

- [Add an element](#)
- [Connect elements](#)
- [The Quick Linker](#)
- [Move elements and packages](#)
- [Auto element naming](#)
- [Search a project](#)
- [Default element template](#)

4.15.1 Add an Element

Elements within a model are typically arranged on diagrams to visually communicate the relationships between a given set of elements. Enterprise Architect provides simple mechanisms for creating elements in the model, using diagrams or the *Project Browser*.

Creating Elements on a diagram

The fastest and simplest ways to create elements directly on a diagram is via the *Quick Linker* and the *UML toolbox*. The following sections describe these and other approaches for creating elements on a diagram:

- [Creating Elements in place using the Quick Linker](#)
- [Creating Elements using the UML Toolbox](#)
- [Creating Elements using the Diagram Context Menu](#)
- [Creating a group of Elements using UML Patterns](#)
- [Creating Domain Specific Elements from UML Profiles](#)

Adding Elements directly to a package

Sometimes it is useful to add elements to a package, without a diagrammatic representation. This can be accomplished via the *Project Browser* and is explained in the following section:

- [Adding Elements directly to a package](#)

4.15.2 Connecting Elements

Creating Connectors on a diagram

The fastest and simplest ways to create connectors are via the *Quick Linker* and the *UML toolbox*. The following sections describe these and other approaches for creating connectors on a diagram:

- [Creating Connectors in place using the Quick Linker](#)
- [Creating Connectors using the UML Toolbox](#)
- [Creating a group of Elements using UML Patterns](#)
- [Creating Domain Specific Connectors from UML Profiles](#)

Note: You may reposition a connectors by selecting and dragging the links as required.

Tip: To repeat the last connector you used, press the **F3** key .

Tip: To reattach one connector end to a different element, use Shift+Left-Click on a connector end, then select the desired element.

Creating Connector without a diagram

Sometimes it is useful to create relationships between elements, without a diagrammatic representation. This can be accomplished via the *Project Browser* and the *Relationship Matrix*. These approaches are explained in the following topics:

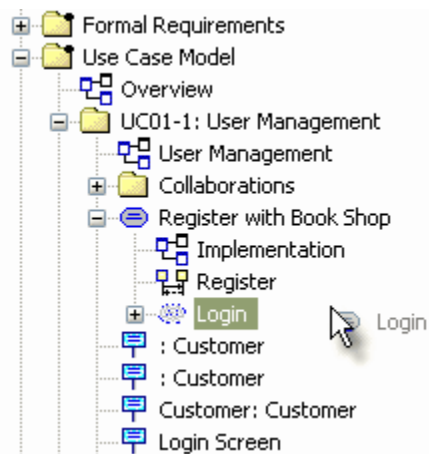
- [Adding Connectors with the Project Browser](#)
- [Adding Connectors with the Relationship Matrix](#)

4.15.3 Moving Elements and Packages

Elements and packages can be moved from one package to another by dragging and dropping the source element to the target destination in the Project Browser. Note that if you move a package, ALL the child packages and their contents will be moved to the new location also.

To move a model element between packages, follow these steps:

1. Left click the model element in the Project Browser.
2. Holding down the mouse button, drag the element to the target package.
3. Ensure you have positioned the mouse cursor over a package icon.



4. Release the mouse button. The element will automatically be shifted.

Warning: In a multi-user scenario, if one person moves or updates the Project Browser structure, other users will need to reload their project to see the latest changes in the Project Browser. Although this is true of any addition or modification to the tree, it is most important when big changes are made, such as dragging a package to a different location.

4.15.4 Auto Counters

The *Auto Element Naming* dialog allows you to configure automatic naming for any element type. Each element can have separately configured automatic names and aliases.

To set up auto naming, follow these steps:

1. Select *Settings | Auto Counters* from the main menu. This will open the *Auto Element Naming* window.

2. Select the element *Type* (eg. Activity) from the drop down menu.
3. In the Name group:
 - Set the (optional) Prefix the new name will have.
 - Set the Counter value - use a many 0's as required to pad the name.
 - Set the (optional) Suffix the new name will have.

- Tick the Active check box to turn auto naming on for this element type.
- 4. In the Alias group:
 - Set the (optional) Prefix the new alias will have.
 - Set the Counter value - use a many 0's as required to pad the alias.
 - Set the (optional) Suffix the new name will have.
 - Tick the Active check box to turn auto naming on for this element type.
- 5. Press **Save**.

New elements of this type will now have an automatically generated name and/or alias with an incrementing counter value.

Note: If Alias is active the auto naming will apply, however to view the Alias in a diagram requires that the option Use Alias if Available is select in Diagram Properties.

See Also

- [Diagram Properties](#)

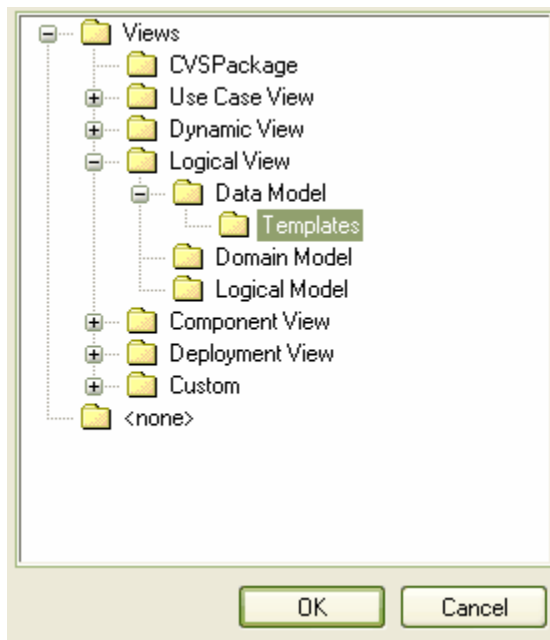
4.15.5 Templates Package

For control over the appearance of elements, the default element template can be set. This functionality could be used, for example, to denote different stages of a project. A template with different color fill could be created for each stage, for example, so that elements added in each stage are instantly identifiable as belonging to that stage. Element templates do not support the definition of stereotypes within the template. This means that the when a stereotype is selected the stereotype settings will cause the template to revert to the default setting.

To configure the default element template, follow these steps:

1. First create a new package - this could be named "Templates" for example.
2. Within the Templates package create new diagrams - one for each type of diagram you wish to template. Name them so that they are easy to recognise eg. ClassTemplate for the template for class diagrams.
3. Add new elements to the template diagrams from the UML Toolbox and configure the size, appearance, notes, version, etc.
4. To set the templates as the default element templates, select **Settings | Template Package** from the main menu.

The **Browse Project** window is displayed.



5. Navigate to the *Templates* package. Select it and press *OK* to set it as the default element template.

Now each new element you add to your project will be created with the settings in the template diagrams. When you create elements, EA will check the templates directory first and if a template is found, it will copy the settings from there.

Note: If you decide you do not wish to use the default element template, set the default element template to *<none>* in the *Browse Project* window.

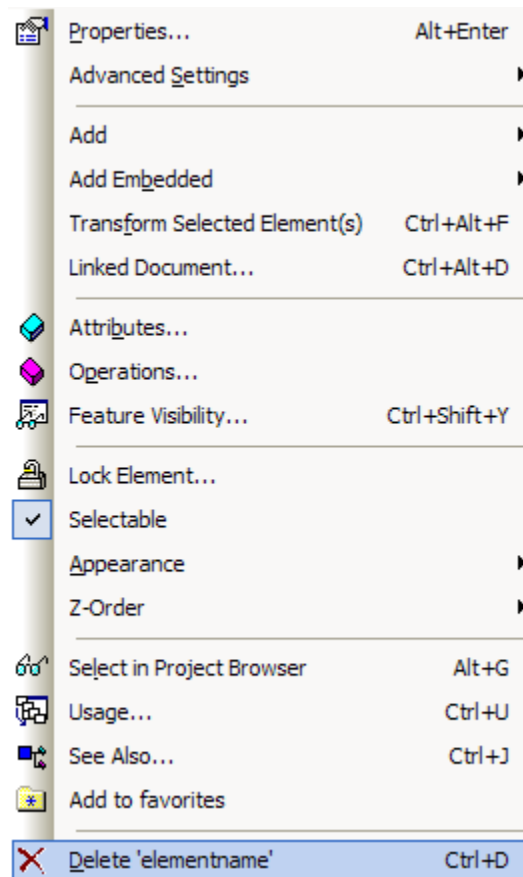
4.15.6 Deleting an Element

Elements in Enterprise Architect exist in the Model and are displayed in the Project Browser and diagrams, and are deleted separately.

Deleting an Element from a Diagram

There are several methods for deleting an element from a diagram.

1. Select the Element and press the Delete key
2. Select the Element and use the keyboard shortcut - Control-D
3. Right Click the element on the *diagram* and select *Delete 'elementname'*



Note: Control-Delete (Ctrl-Del) will delete an element from the entire model. See below.

Deleting an Element from a Model

These are **permanent** deletions of an element from a model. It will be removed from **every** diagram in the project. To delete an element from an entire model, you can:

1. Select the element and use the keyboard shortcut Control-Delete (Ctrl-Del)
2. Right click the element in the *Project Browser* and select *Delete 'elementname'*

4.16 Element Inplace Editing Options

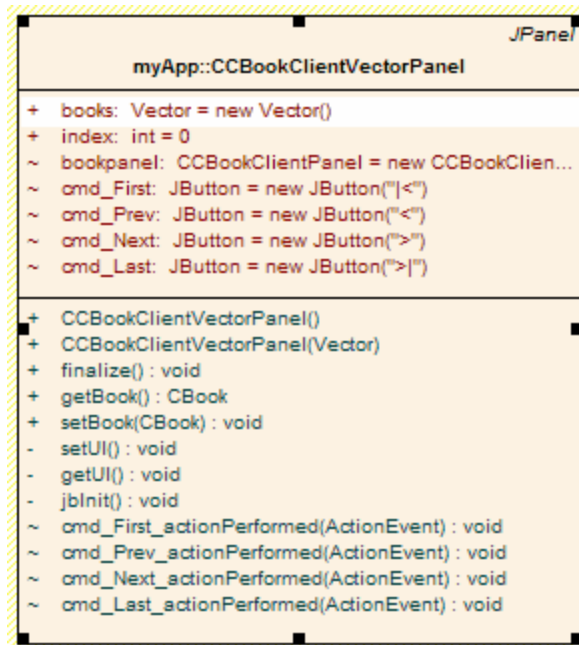
This section explores the tasks that can be performed using hotkeys and in place editing of elements. The tasks include:

- [Edit Name](#)
- [Edit Stereotype](#)
- [Edit Scope](#)
- [Edit Attribute keyword](#)
- [Edit Operation Parameter keyword](#)
- [Insert Operation Parameter](#)
- [Edit Parameter kind](#)
- [View Properties](#)
- [Insert New after Selected](#)
- [Add Maintenance Item](#)
- [Add Test Item](#)
- [Delete Selected from Model](#)

4.16.1 Inplace Element Item Tasks

To use the options that are available through the inline editing menu use the following steps:

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. The items in the element can now be directly edited and manipulated, this may be achieved by either using the appropriate hotkey or by using right clicking on the highlighted item and choosing a task from the Element Items menu.

The following commands are available:

| Menu Option | Hotkey | Description |
|---------------------------|------------------------------|--|
| Edit Selected | <i>F2</i> | Attach a note or attach a constraint to the element. |
| View Properties | <i>Enter</i> | Opens up the dialog window containing details of the element. |
| Insert New After Selected | <i>Ctrl + Shift + Insert</i> | Inserts a new item to the element. |
| Add Attribute | <i>Ctrl + Shift + F9</i> | Adds an attribute to the element. |
| Add Operation | <i>Ctrl + Shift + F10</i> | Adds an operation to the element. |
| Add Other | <i>Ctrl + F11</i> | Allows the user to insert a feature on the specific element item, such as Maintenance features and Testing features. |

| | | |
|----------------------------|---------------------------------|---|
| Delete Selected from Model | <i>Ctrl + Shift + Delete</i> | Deletes the selected item from the model. |
| | <i>Ctrl + Shift + Arrow key</i> | Navigate Diagram Selection, this option allows users to navigate the diagram between elements without needing to use the mouse. |
| | <i>Shift + Enter</i> | Toggle element highlight option on and off |

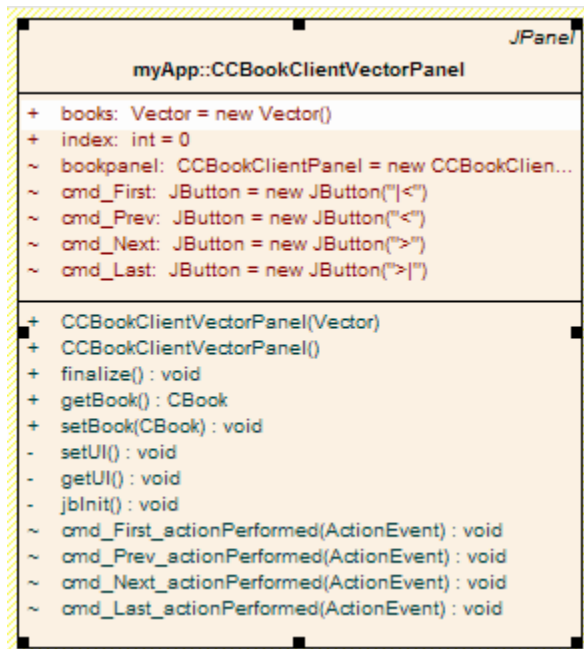
Other options that are available while in editing elements mode in a diagram (when an attribute or operation is highlighted):

| Hotkey | Description |
|---------------------|--|
| <i>Enter</i> | Accept current changes |
| <i>Ctrl + Enter</i> | Accept current changes and open a new slot to add a new item |
| <i>Esc</i> | Abort edit, without save |
| <i>Shift + F10</i> | Context menu for in-place editing |
| <i>Ctrl + Space</i> | Invoke Classifier Dialog |

4.16.2 Edit Element Name

By using the inline editing feature it is possible to change the name of an operation or attribute directly from the diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).

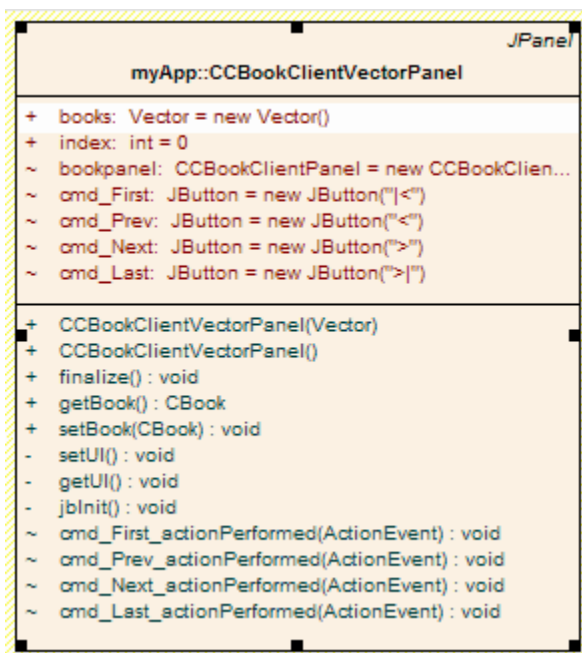


3. Right click the mouse and select *Edit Selected* from the menu or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Delete or type over the previous name to change the name of the attribute or operation. Then press the *Enter* key to accept the change or press the *Esc* key to cancel the change.

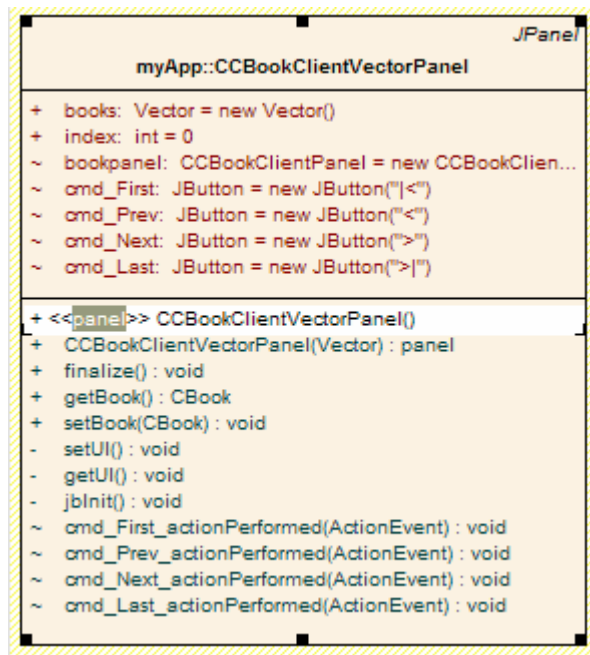
4.16.3 Edit Attribute or Operation Stereotype

By using the inline editing feature it is possible to change the Stereotype of an operation or attribute directly from the diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Move the cursor to the position before the name or within the existing attribute or operation Stereotype (denoted by << stereotype name >>).

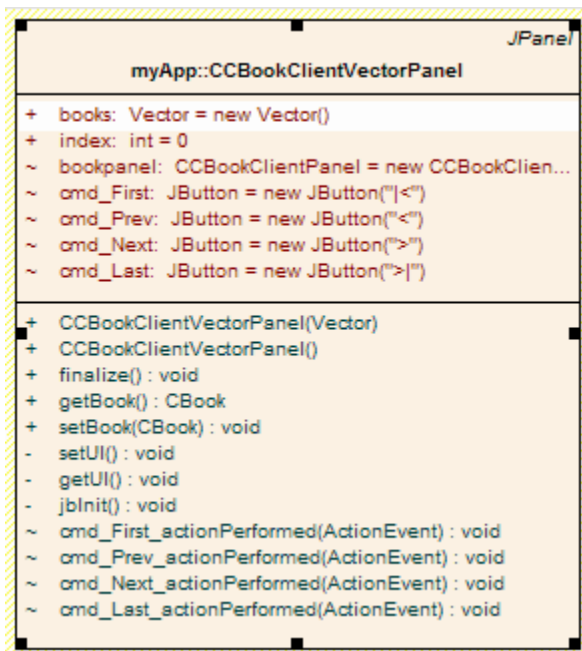


6. Delete or type over the previous Stereotype to change the Stereotype name of the attribute or operation. Then press the *Enter* key to accept the change or press the *Esc* key to cancel the change. Multiple stereotypes can be assigned by including a comma separated list inside the stereotype markers.

4.16.4 Edit Attribute and Operation Scope

The inline editing feature allows the user to rapidly change the scope of an attribute or operation directly from the diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).

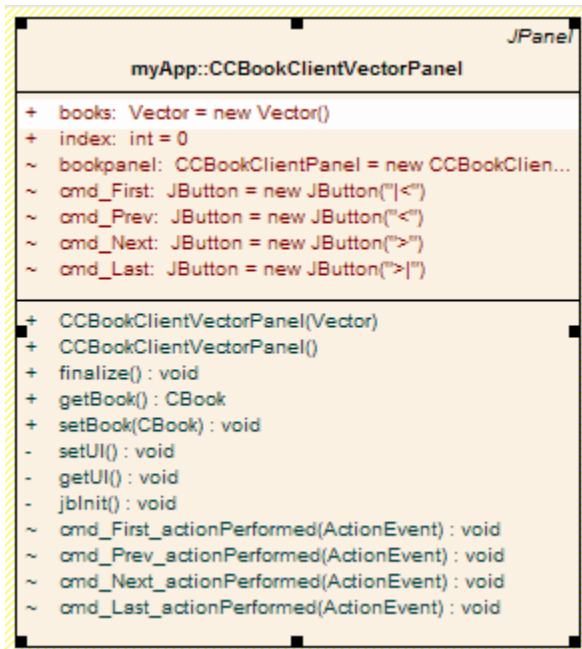


3. Right click the mouse and select **Edit Selected** or use the **F2** hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Move the cursor to the attribute or scope of the item and delete the previous entry then reassign the entry by typing in the following symbols:
 - + indicates that the scope is Public.
 - - indicates that the scope is Private.
 - ~ indicate that the scope is Package.
 - # indicates that the scope is Protected.
6. Then press the **Enter** key to accept the change or press the **Esc** key to cancel the change. Once this procedure is completed the diagram will be updated to reflect the changes.

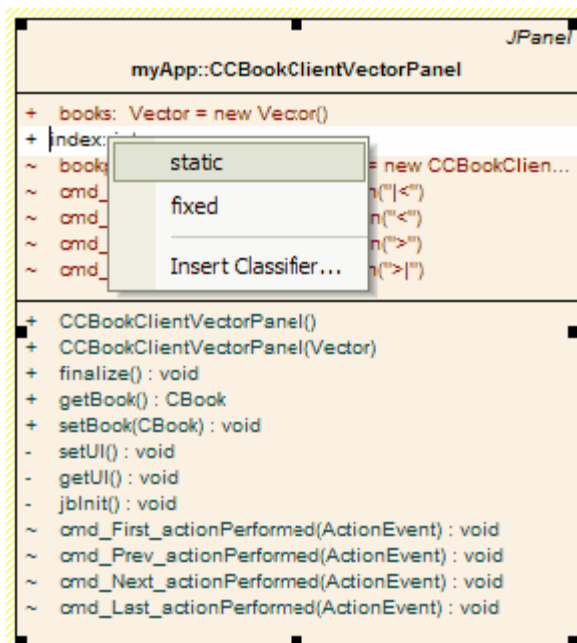
4.16.5 Edit Attribute Keyword

Features such as element keywords and classifiers may be directly added to an element by using the Element Keywords and Classifiers menu. This allows the user rapidly assign details on a per element item basis directly from a diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



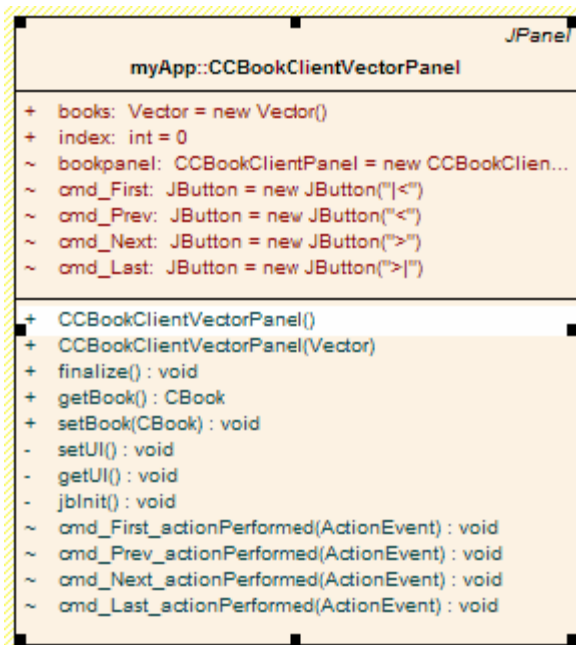
3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. If the item selected is an attribute it is possible to change the attribute classifier to static or fixed by moving the cursor to the beginning of the attribute name then right clicking the highlighted attribute and selecting the static or fixed options as needed the diagram will then update to reflect the changes.



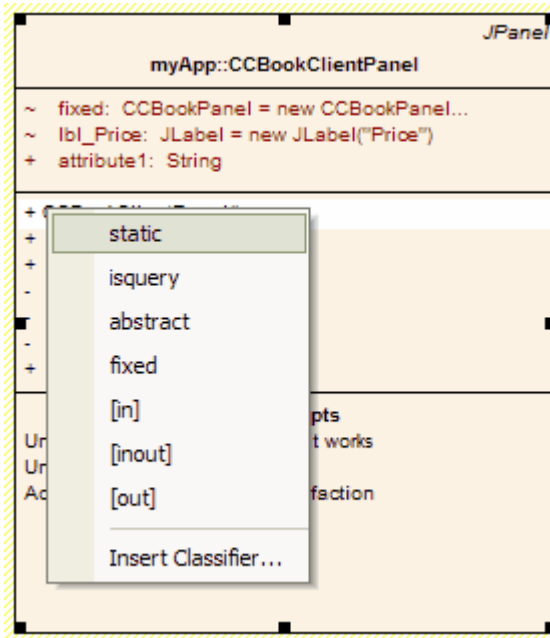
4.16.6 Edit Operation Parameter Keyword

Operation classifiers may be directly edited element by using the functions of the in-place editing menu. This allows the user rapidly assign parameter keywords. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the operation that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



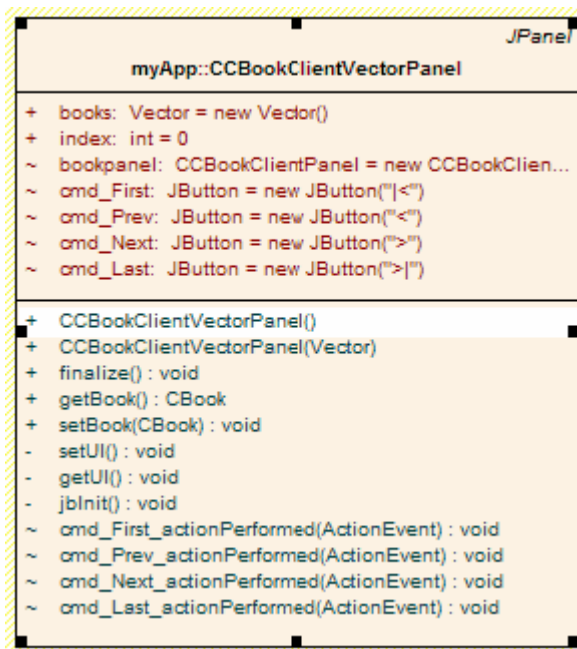
3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the operation to be edited directly from the diagram.
4. This will highlight the name of the operation.
5. If the item selected is an operation it is possible to change the operation classifier to static, isquery, abstract or fixed by moving the cursor to the beginning of the operation name then right clicking the highlighted attribute and selecting the appropriate options as needed the diagram will then update to reflect the changes.



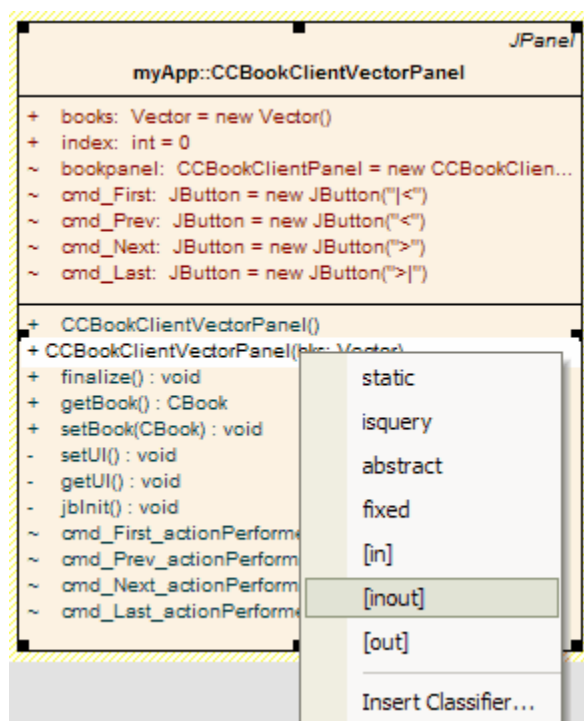
4.16.7 Edit Parameter Kind

The operation parameter kinds such as [in], [inout] and [out] directly edited an element by using the Element Keywords and Classifiers menu. This allows the user rapidly assign the parameter directly from a diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



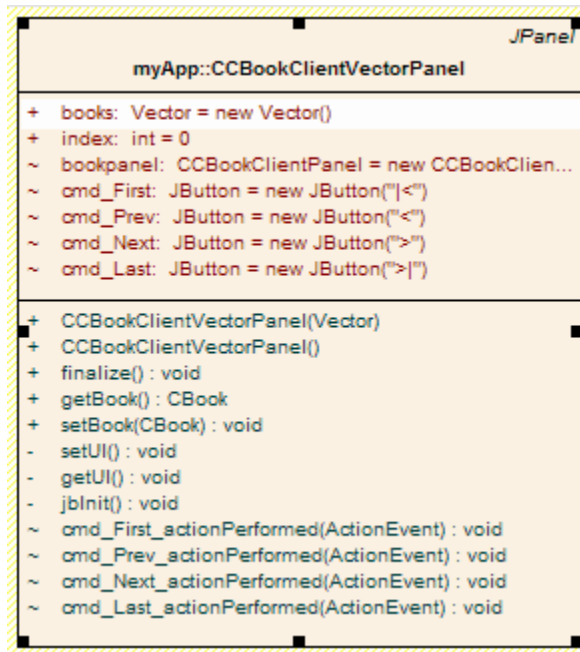
3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Operations also allow the user to set the parameter kind value to [in], [inout] and [out] by setting the cursor to the beginning of the parameter and then right clicking the highlighted operation and selecting the appropriate parameter kind value.



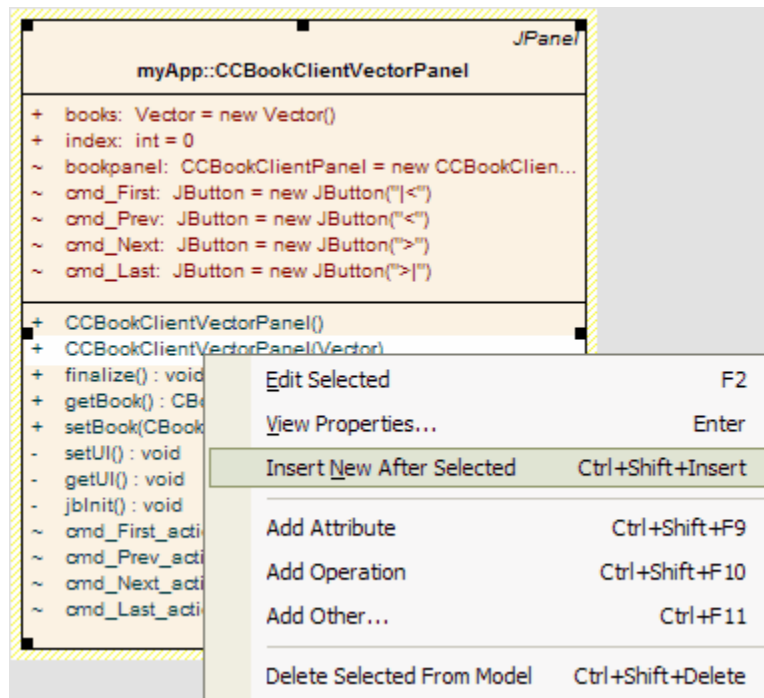
4.16.8 Insert new Attribute or Operation

Attributes and operations may be added to an element through the inline editing options by using a series of hotkey commands. To add attributes and operations to a class diagram element use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. Highlight the element that is to have either an attribute or operation added, then press the **Ctrl + Shift + Insert** hotkey combination or right click on the selected element item and choose the **Insert New After Selected** menu option.

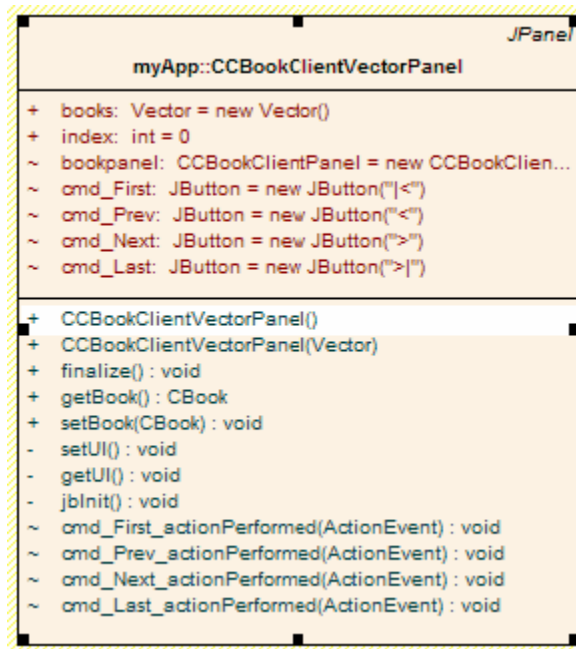


4. Type in the relevant information relating to the attribute or operation then press the **Enter** key to accept the change or press the **Esc** key to cancel the change. Once this procedure is completed the diagram will be updated to reflect the changes.

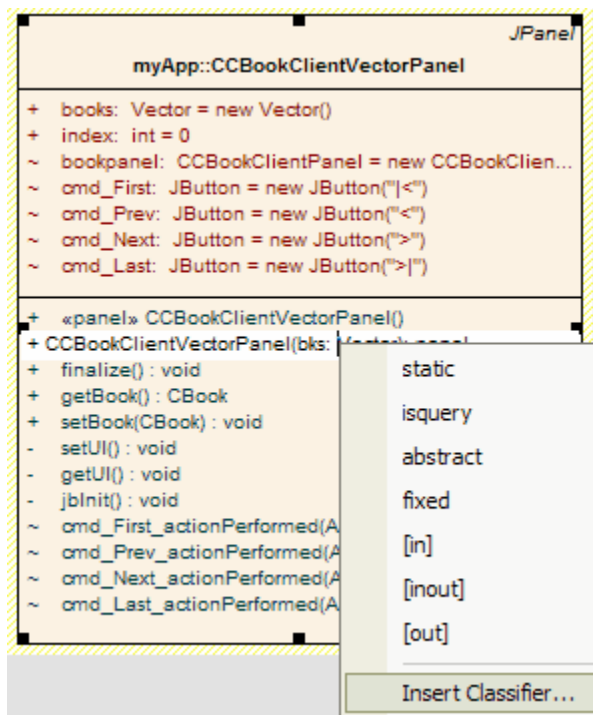
4.16.9 Insert Operation Parameter

Operations parameters may be added to an operation through the inline editing options by using a series of hotkey commands or menu shortcuts. To add parameters to operations in a class diagram element use the following instructions.

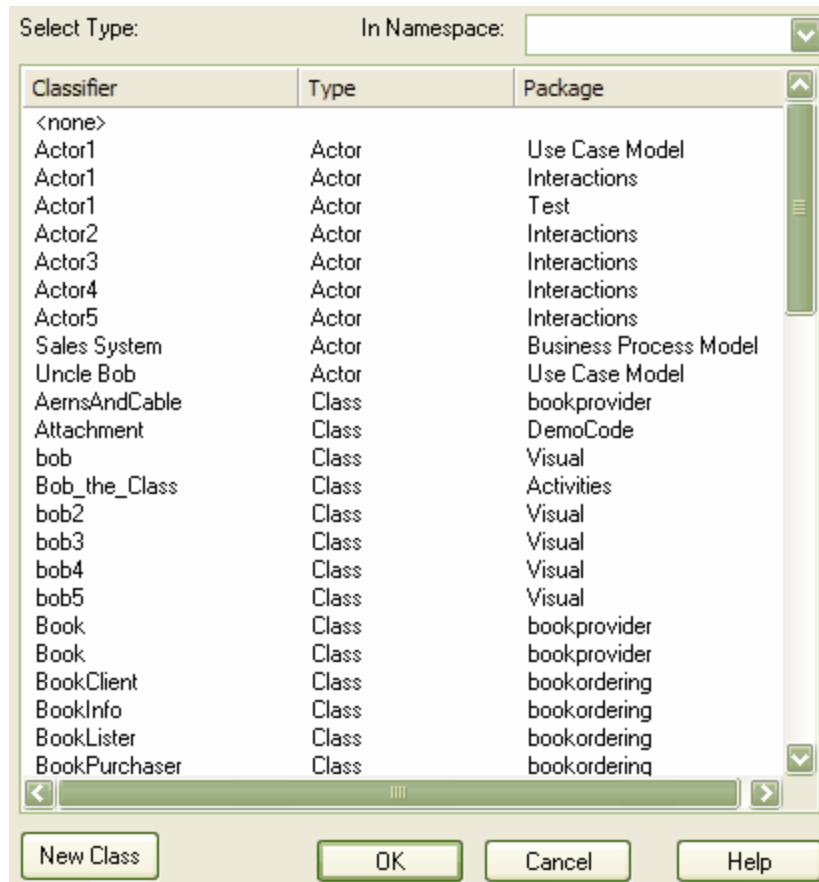
1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. Highlight the operation that is to have the parameter added or changed, then press **F2** or right click the selected item and select the **Edit Selected** menu item.
4. Move the cursor to the inside of the parameter brackets the choose reference to the parameter (i.e. bks: for a vector containing books) then either type the name of the parameter or place the cursor after the reference then right click the mouse to bring up the inline editing options menu and select the **Insert Classifier..** option.



- The Set Element Classifier dialog will then appear allowing the selection of an appropriate parameter from the list of available items. Select the appropriate one then press the **OK** button to confirm your choice.

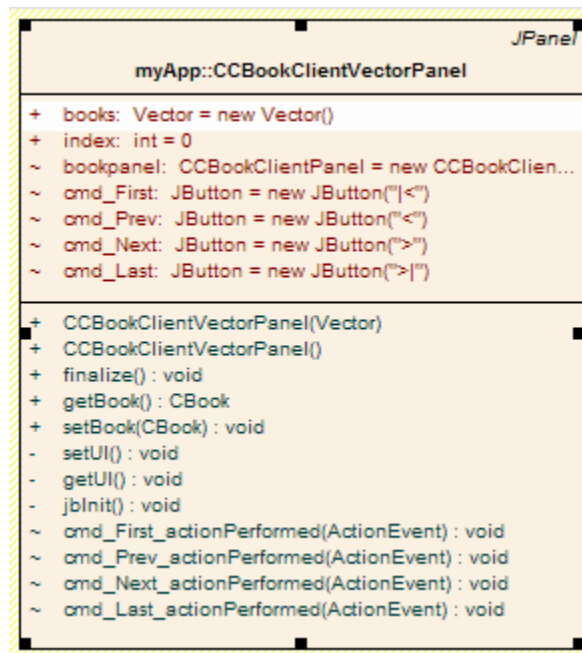


- Press the **Enter** key to accept the change or press the **Esc** key to cancel the change. Once this procedure is completed the diagram will be updated to reflect the changes.

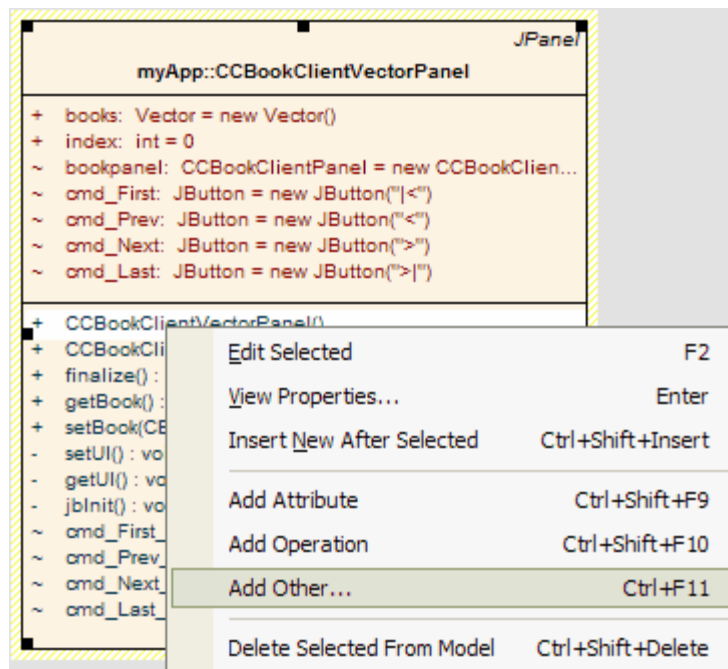
4.16.10 Insert Maintenance Feature

Maintenance items such as Defects, Changes, Issues and Tasks can be directly added to an element by using the Element Items menu. This allows the user rapidly assign maintenance details to an element item directly from a diagram. To use this feature use the following instructions.

- Open the diagram containing the element.
- Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



- The items in the element can have a feature associated with it by using the **Ctrl + F11** hotkey combination or by right clicking the highlighted item and selecting the **Add Other** option.



- This will open up the Insert Feature dialog which allows the user to associate testing or maintenance features to the element item by selecting the appropriate radio button option and then pressing the **OK** button

Select feature to insert

Maintenance

Defect
 Change
 Issue
 Task

Testing

Unit
 Integration
 System
 Acceptance
 Scenario

OK
Cancel

- This will open the appropriate Maintenance Dialog for the selected element, fill in the details and then save the details by pressing the **Save** button.
- When all of the maintenance item/s for this maintenance type for the element have been completed press the **OK** button.

Details

Name:

Date: Version:

Reported by: Priority:

Description:

History

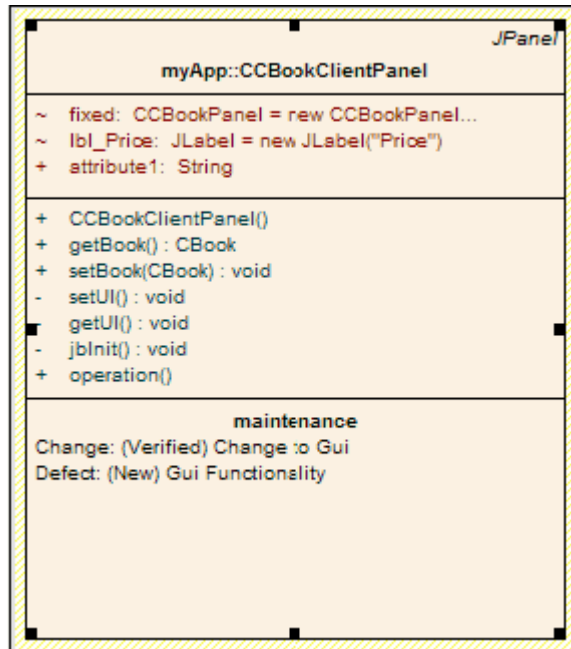
Status: Resolved by:

Date Resolved:

History:

- The maintenance details will now be added to the element. If the diagram properties appearance options

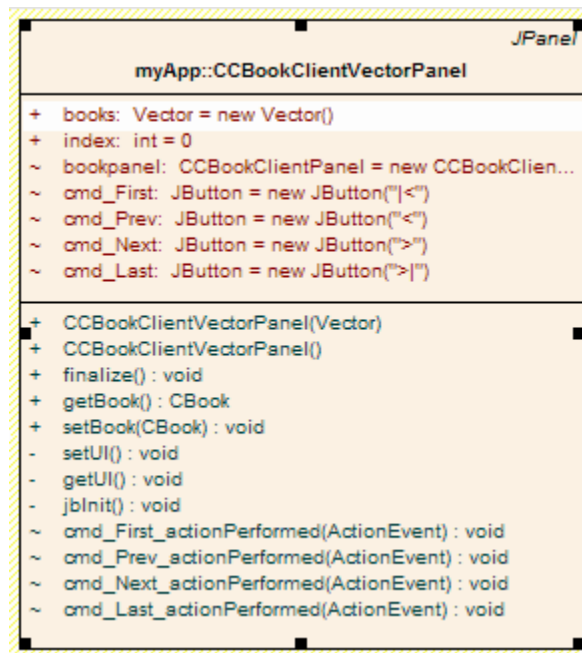
have had the *Show Maintenance* option checked the maintenance item/s will be visible in the diagram element, as shown in the example below. For more information relating to diagram appearance options is found in the [Show Maintenance Scripts in Compartments](#) section.



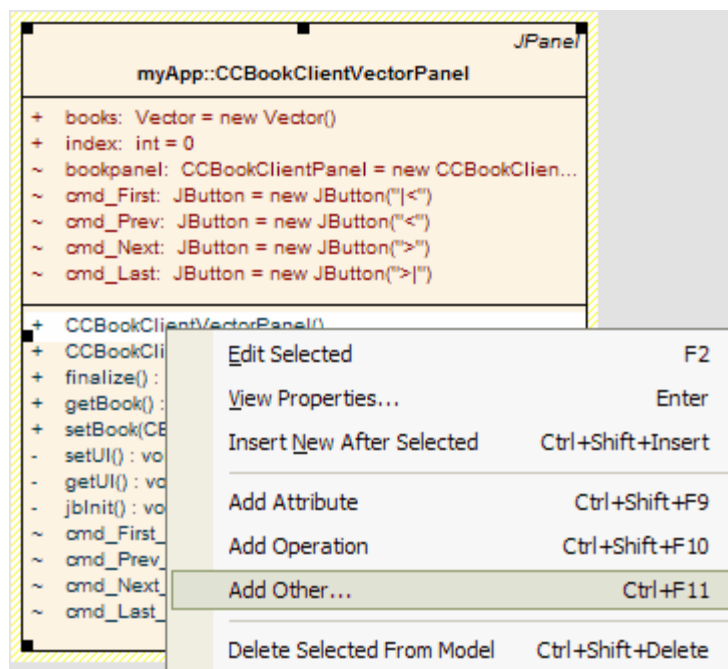
4.16.11 Insert Testing Feature

Testing features such as Unit, Integration, System, Acceptance and Scenario tests can be directly added to an element by using the Element Items menu. This allows the user rapidly assign tests to an element item directly from a diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



- The items in the element can have a feature associated with it by using the **Ctrl + F11** hotkey combination or by right clicking the highlighted item and selecting the **Add Other** option.



- This will open up the Insert Feature dialog which allows the user to associate testing or maintenance features to the element item by selecting the appropriate radio button option and then pressing the **OK** button

Select feature to insert

Maintenance

Defect

Change

Issue

Task

Testing

Unit

Integration

System

Acceptance

Scenario

OK

Cancel

5. This will open the appropriate Test Dialog for the selected element, fill in the details and then save the details by pressing the **Save** button.
6. when all of the test items for this type of test for the element have been completed press the **OK** button.

Test Details and Execution Status

Test: Type:

Description:

Input:

Acceptance Criteria:

Execution

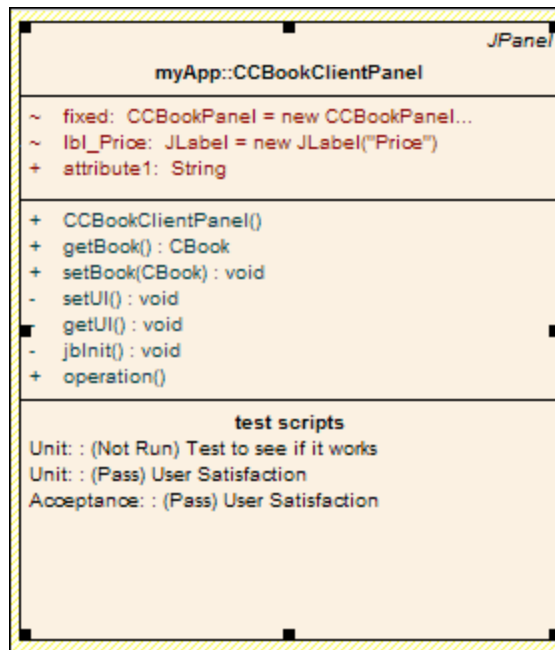
Status: Last Run Date:

Run By: Checked By:

Results:

7. The test details will now be added to the element. If the diagram properties appearance options have had the **Show Testing** option checked the test item/s will be visible in the diagram element, as shown in the

example below. For more information relating to diagram appearance options is found in the [Show Test Scripts in Compartments](#) section.



4.17 Defaults and User Settings

You can configure various settings using the [Local Options dialog](#) (open by selecting *Options* from the *Tools* menu). In addition, there are several options to change the [overall look and feel of EA](#) in the *View | Visual Style* submenu. Those settings and options are explored in the following section.

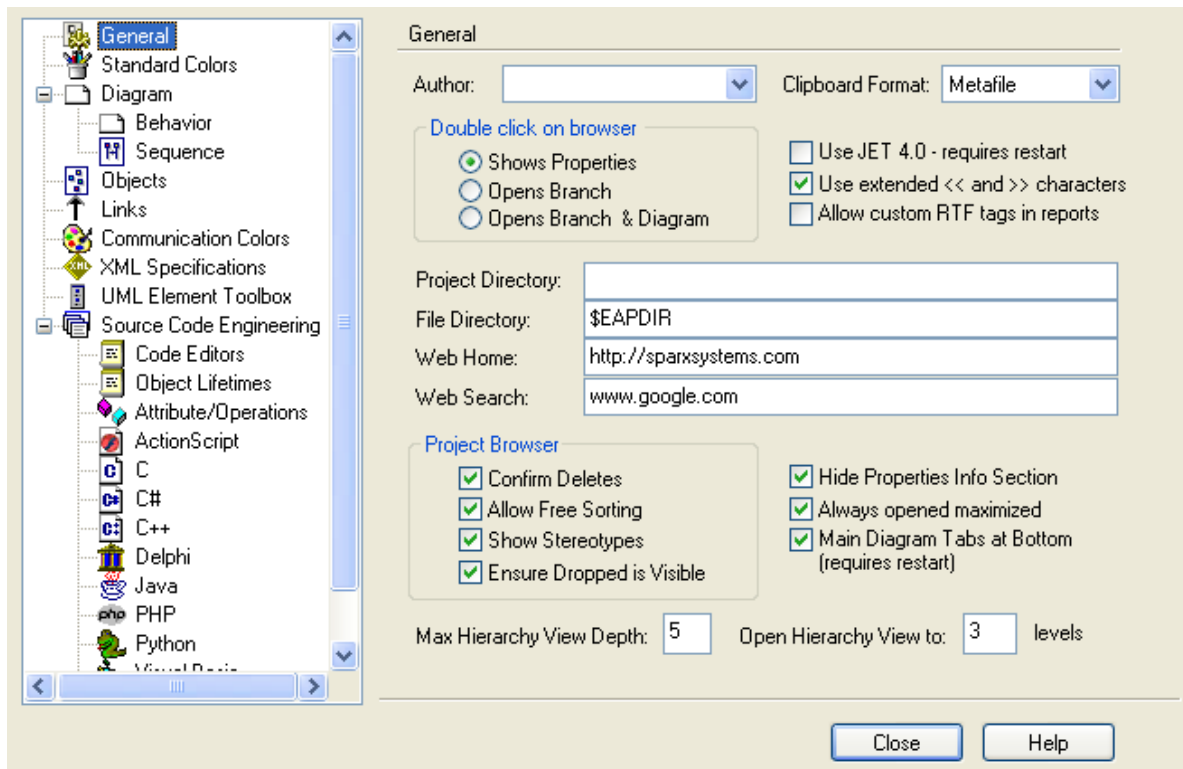
See Also:

- Configure Local Options
- Custom Layouts
- Visual Styles

4.17.1 Configure Local Options

There are a large number of options to customize how EA displays and works with models and model elements. This section describes those settings that are local to a particular user and machine.

Open the *Options* dialog by selecting *Tools | Options* from the main menu.

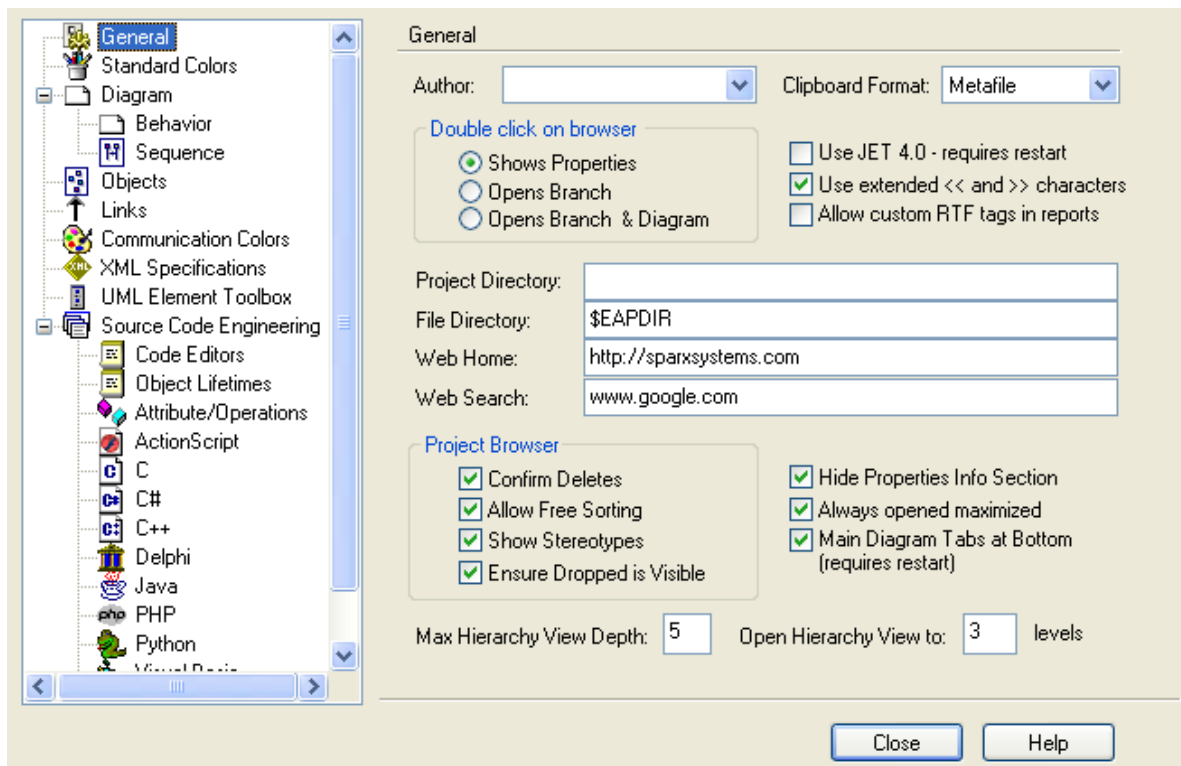


Most of these settings are stored in the user's registry so they are set for that user only. For a networked workplace, registry settings can be copied down to any network workstation a user logs in to. Otherwise, the settings are valid for the current machine only.

Note: Additional defaults and settings are discussed under the various code generation and import/export topics in this help file.

4.17.1.1 General

The *General* section of the *Options* dialog is shown below:



General Settings:

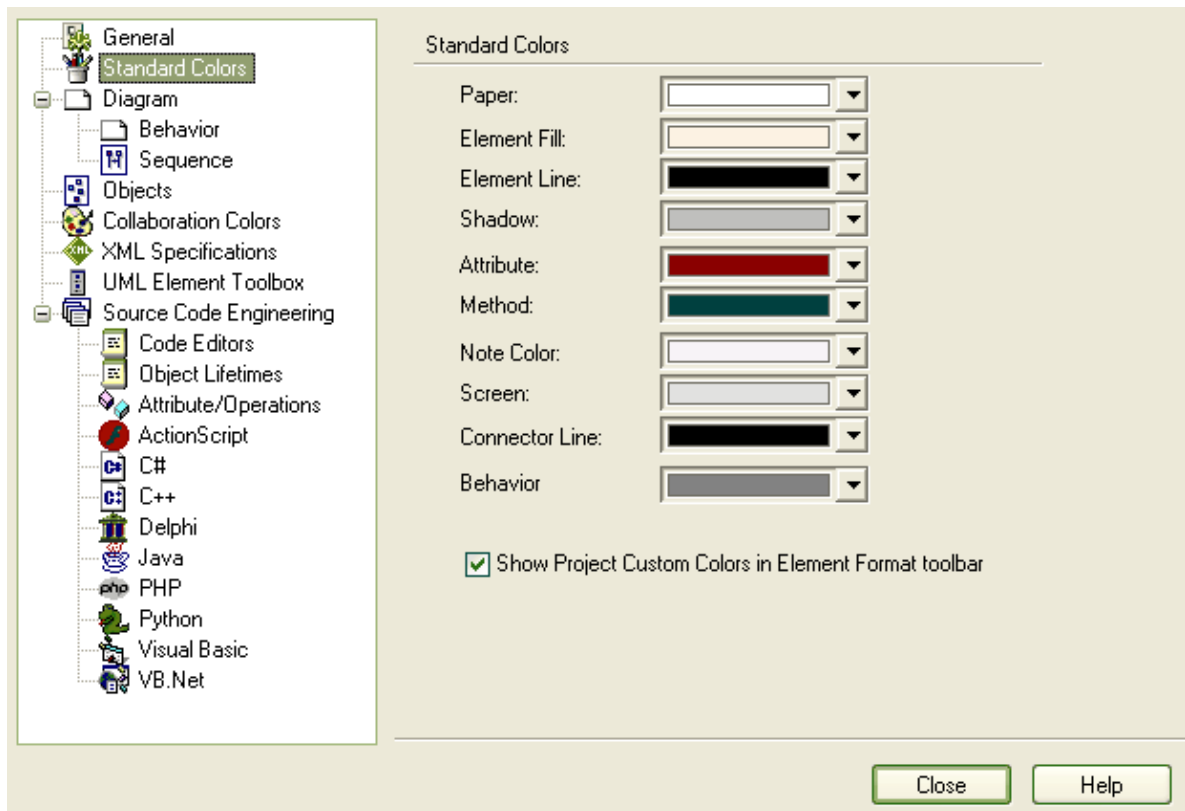
- **Author** - This will be used as the default author when new elements are created and modifications made.
- **Clipboard Format** - Graphic format in which to save image to clipboard - Metafile has the best detail.
- **Double Click on Browser** - Configure [Project Browser behavior](#).
- **Use JET 4.0 - requires restart** - Use JET 4.0 as database engine, this ensures compatibility with .EAP files compatible with versions of MS Access later than Access 97.
- **Use extended << and >> characters** - For some double byte character sets. It is best to check this box.
- **Allow custom RTF tags in reports** - Allow the use of custom RTF tags in reports.
- **Project Directory** - Default location of EA projects.
- **File Directory** - Default location for files.
- **Web Home** - Specifies the default home page to open when pressing the Home button in the internal web browser
- **Web Search** - Specifies the default web page to open when pressing the *open Web Search* button is pressed in the [Instant Help Window](#).
- **Confirm Deletes** (formerly "Confirm when objects are deleted from browser") - Clear to bypass confirmation dialog - only for experienced users!
- **Allow Free Sorting** - Allows "free sorting" of elements in the Project Browser regardless of type.
- **Show Stereotypes** - Shows element and feature stereotypes in the Project Browser.
- **Ensure Dropped is Visible** - If this option is checked, when moving an element in the Project Browser to another folder, the destination folder will open when you drop the element. If this option is unchecked, the destination folder will stay closed when you drop the element.
- **Hide Properties Info Section** - When checked, hides the properties information status bar on the Properties tab.
- **Always open maximized** - When checked, EA will always start up in a maximized window.
- **Main Diagram Tabs at Bottom (requires restart)** - When checked (by default) the diagram tabs will

appear at the bottom of the mainview. Clearing this option will result in the tabs being located at the top of the mainview.

- **Max Hierarchy View Depth** - Sets the maximum number of levels that the hierarchy will open to on the [Hierarchy tab](#).
- **Open Hierarchy View to** - Sets the initial number of levels that the hierarchy will open to on the [Hierarchy tab](#).

4.17.1.2 Standard Colors

The **Standard Colors** section of the **Options** dialog is shown below:



Standard Colors Settings:

- **Paper** - Paper color in diagrams.
- **Element Fill** - Fill color of elements.
- **Element Line** - Line color of elements.
- **Shadow** - Shadow color.
- **Attribute** - Attribute color.
- **Method** - Method color.
- **Note Color** - Note background color.
- **Screen** - Screen (element) color.
- **Connector Line** - Connector line color.
- **Behavior** - Color for behaviors in Activity diagrams.
- **Show Project Custom Colors in Element Format toolbar** - This checkbox is used allow the showing of project custom colors for more information relating to setting and getting the custom colors see the [Get and Set Project Custom Colors](#) topic.

4.17.1.3 Diagram

The *Diagram Settings* section of the *Options* dialog shown below allows you to configure overall options for new diagrams and general diagram behavior.

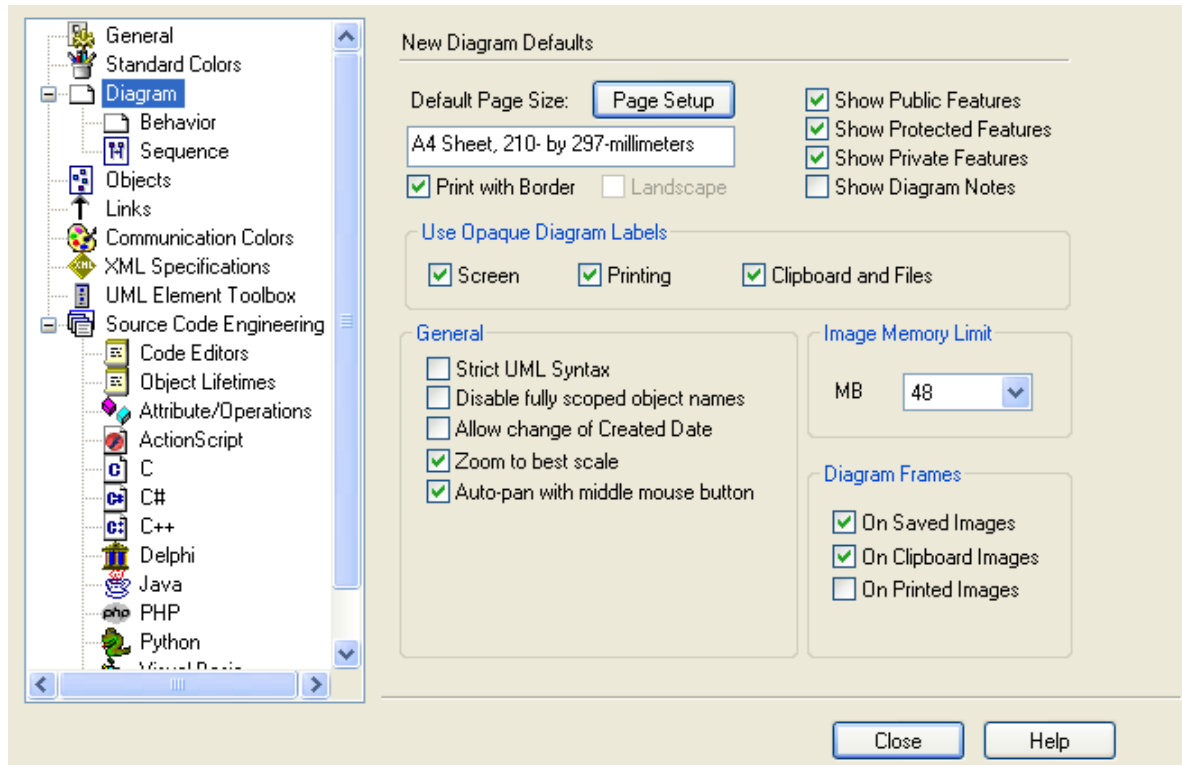


Diagram Settings:

- **Default Page Size** - Default page size for new diagrams. Change this using the *Page Setup* dialog
- **Print with Border** - Select to have pages print with a border.
- **Landscape** - If checked, pages print in landscape orientation. This checkbox is controlled from the *Page Setup* dialog.
- **Show Public/Protected/Private Features** - Set the default visibility of class features.
- **Show Diagram Notes** - Set default visibility of details on new diagrams - details include diagram name, package, version and author.
- **Use Opaque Diagram Labels** - Screen and Printing are best, Clipboard and images may not be desirable.
- **Strict UML Syntax** - Enforces when adding new connectors etc.
- **Disable fully scoped object names** - Use this when an element is in a diagram but not in its home package. A scoped name may look like "MyClasses::foo" , the '::' character indicating the class is within another namespace.
- **Allow change of Created Date** - Select to allow the creation date on the diagram properties dialog to be altered.
- **Zoom to best scale** - If this box is checked, the diagrams will fit neatly to screen.
- **Auto-pan with middle mouse button** - Turns on auto-panning using the middle mouse button. With this option off, a different type of panning is used with the middle mouse button.
- **Image Memory Limit** - Used when generating images for RTF or HTML and when saving images to file. It is important when you have very large diagrams, as it affects the point at which EA will start to scale down the image - a low memory setting means it will scale the image sooner.

4.17.1.3.1 Behavior

The *Diagram Behavior* section of the *Options* dialog is shown below:

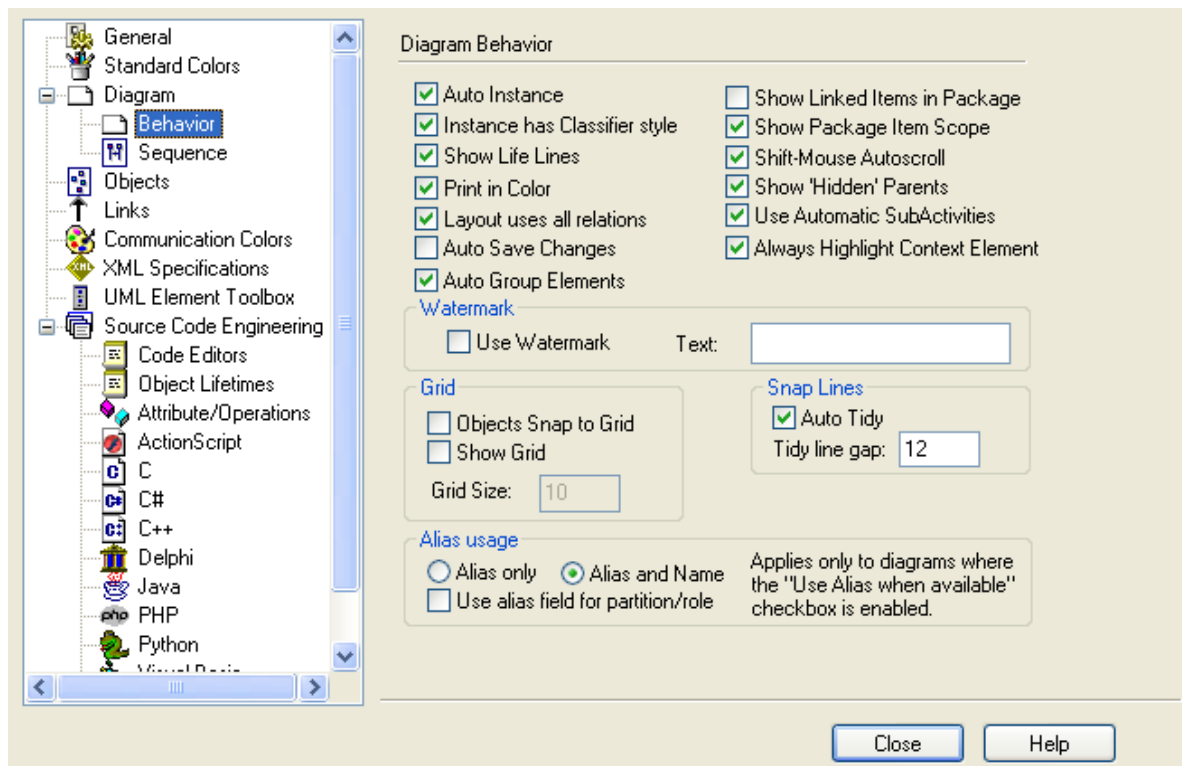


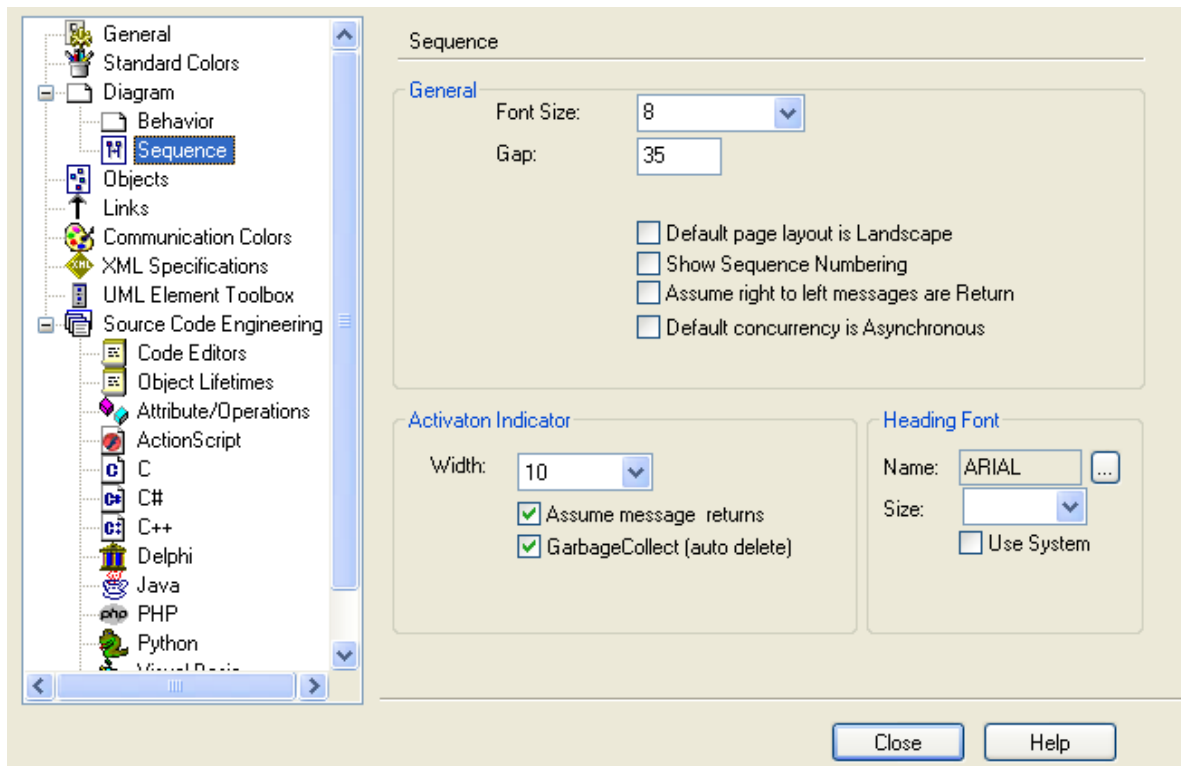
Diagram Behavior Settings:

- **Auto Instance** - When Auto Instanting is turned on, some element types - such as Class and Component - will create object instances with the dragged object as the classifier.
- **Instance has Classifier style** - When an instance is created it will have the classifier style of the element it was instantiated from.
- **Show Life Lines** - Toggle whether to show life lines for sequence elements in NON-SEQUENCE diagrams.
- **Print in Color** - Elect whether you want your diagrams printed in color or black and white.
- **Layout uses all relations** - If unchecked, only generalizations and associations will be shown.
- **Autosave changes** - If checked, EA will automatically save your changes as you work, without prompting to do so.
- **Auto Group Elements** - If checked, elements will automatically be grouped so the items can be moved using the "container" object.
- **Show Linked Items in Package** - If checked, packages will display any linked items.
- **Show Package Item Scope** - If checked, the + and - representing the scope of the items will be display.
- **Shift-Mouse Autoscroll** - If checked, when you hold down shift you can use the mouse to autoscroll around diagrams.
- **Show 'Hidden' Parents** - If checked, any parents of elements in the diagram which aren't part of the diagram will be displayed.
- **Use Automatic Subactivities** - If checked, Structured Activity diagrams dragged from the tree will generate a new Structured Activity that is linked to this diagram.
- **Always Highlight Context Element** - Toggle whether to show a hatch border around the selected element.
- **Use Watermark** - You can use a watermark in any diagrams you print.
- **Text** - Enter the watermark text if a watermark is to be used.
- **Objects Snap to Grid** - If checked, all elements will snap to the grid lines.
- **Show Grid** - If checked, the grid will be displayed.
- **Grid Size** - If **Objects Snap to Grid** is checked, you can set the grid size here.

- **Tidy line gap** - Set the amount EA will allow you to move a line away from horizontal and vertical when you are [tidying lines](#) for custom connectors. (See *Auto Tidy* below).
- **Auto tidy** - Automatically [tidy line angles](#) for custom connectors. This 'nudges' the custom line into horizontal and vertical increments.
- **Alias only** - When Alias only is selected the elements with alias's will be rendered with only the Alias
- **Alias and Name** - When Alias and name is selected both the element name and the Alias will be rendered in the format *(Alias) Name*.
- **Use alias field for partition/role** - When checked, replaces the Alias property of Instances with a Role property.

4.17.1.3.2 Sequence

The *Sequence* section of the *Options* dialog shown below allows you to configure various font settings and focus of the control indicator.



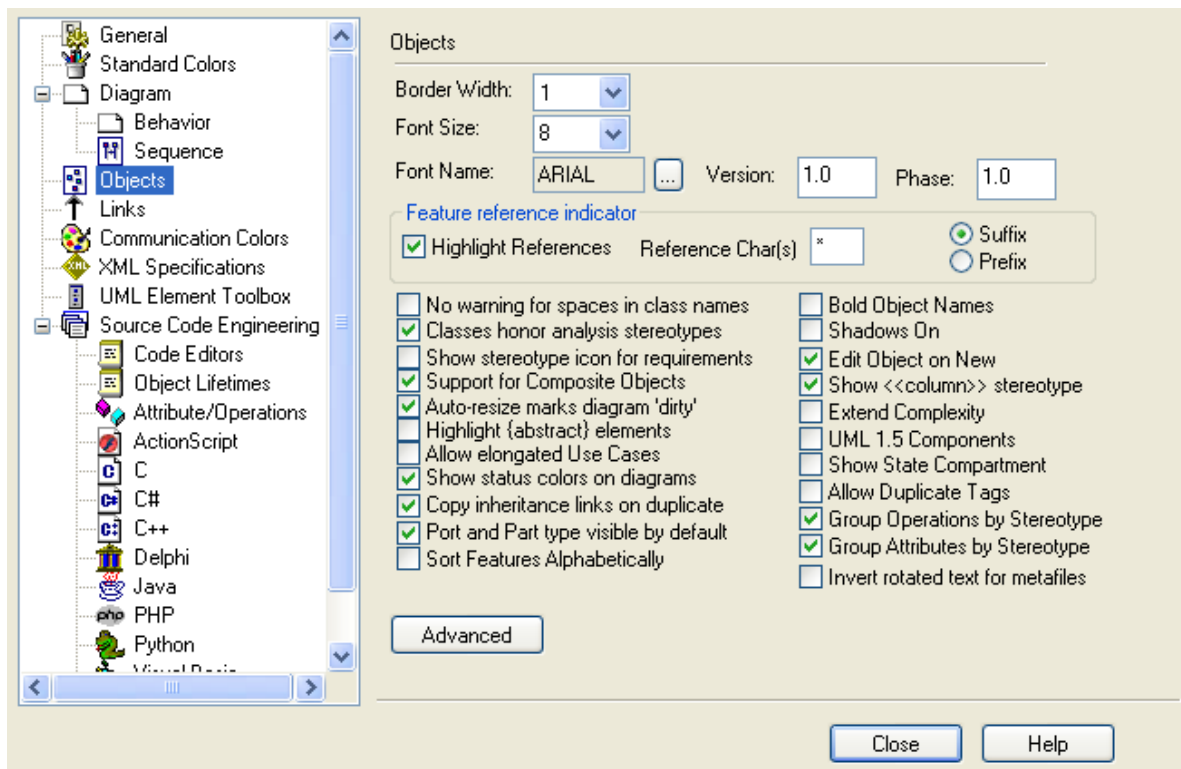
Sequence Settings:

- **Font Size** - For messages in sequence diagrams.
- **Gap** - Set the vertical gap between sequence messages (may be overridden manually by dragging a message up or down).
- **Default page layout is Landscape** - Set the default orientation of Sequence diagrams to landscape.
- **Show Sequence Numbering** - Select whether to show sequence numbers on sequence messages
- **Assume right to left messages are Return** - Select whether to automatically generate return messages.
- **Default concurrency is Asynchronous** - Select whether the default concurrency for [Sequence Messages](#) is Synchronous or Asynchronous
- **Width** - The width of the 'focus of control' rectangle (thick part of lifeline).
- **Assume return** - Assume implicit returns when none are explicitly drawn (recommended).

- **Garbage Collect** - Automatically truncate lifelines for created elements after the last message (ie. assume garbage collect rather than explicit delete).
- **Heading Font Name/Size/Use System** - Set the heading font name and size (above diagram in caption bar). Particularly useful for non-English character sets.

4.17.1.4 Objects

The **Objects** section of the **Options** dialog shown below lets you configure how elements look and respond in diagrams, as well as some connector defaults.



Objects Settings:

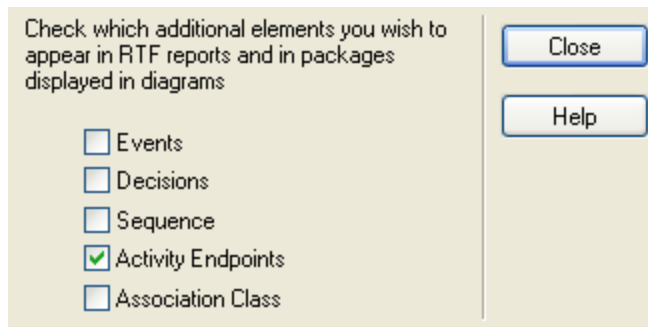
- **Border Width** - Default border width (in pixels).
- **Font Size** - Default font size on diagrams.
- **Font Name** - Font name and style to use for elements.
- **Version** - Default version for new elements.
- **Phase** - Default phase for new elements.
- **Highlight References** - Highlight parameters in operations that are passed by Reference rather than value.
- **Reference Char(s)** - Specify a character to use for the above.
- **Prefix/Suffix** - Indicate whether to use Reference Char(s) as a prefix (before) or a suffix (after).
- **No warning for spaces in class names** - If unchecked, a warning message will appear if an element name has embedded spaces.
- **Classes honor analysis stereotypes** - If checked, classes will be shown as their stereotype - eg. if a class is stereotyped as a boundary, it will appear as a boundary rather than a class.
- **Show stereotype icon for requirements** - Show/hide a small icon in Requirement, Change and Issue elements

- **Support for Composite Objects** - Support embedded or composite elements through automatic aggregation.
- **Auto-resize marks diagram 'dirty'** - Make auto-re sizing of elements (eg. classes) mark the current diagram as dirty.
- **Highlight {abstract} elements** - Highlight abstract elements with a suitable tag "{abstract}" in the top right of the class.
- **Allow elongated Use Cases** - If checked, use cases or use case extension points with long names can be stretched to a disproportionate width to allow space for the name. If unchecked, use case re sizing is proportional.
- **Show status colors on diagrams** - If checked this option enables color coding for requirements.
- **Copy inheritance links on duplicate** - Checked by default, this option duplicates inheritance and realization links when an edit/copy is performed (**Ctrl + Shift + V**).
- **Port and Part type visible by default** - This allows for the ability to show Port and Part types by default.
- **Sort Features Alphabetically** - Sorts element features alphabetically. Features include Attributes, Operations, Tags, Constraints, Test Cases, etc.
- **Bold Object Names** - Elements in diagrams will be shown bold face.
- **Shadows On** - Toggle element shadows.
- **Edit Object on New** - Automatically show the element properties dialog when a new element added.
- **Show <<column>> stereotype** - Hide or show the <<column>> stereotype used when data modeling.
- **Extend Complexity** - If checked, five levels of complexity will be available in the Complexity option on the Properties tab. Otherwise only three levels are available.
- **UML 1.5 Components** - Use UML 1.5 components (Enterprise Architect version 4 and over supports UML 2.0).
- **Show State Compartment** - This shows or hides the visibility of the State Compartment divider under the state name.
- **Allow Duplicate Tags** - This allows for tags to be duplicated.
- **Group Operations by Stereotype** - Groups an elements operations by their stereotype on the diagram
- **Group Attributes by Stereotype** - Groups an elements attributes by their stereotype on the diagram
- **Inverted rotated text for metafiles** - use when external metafile readers are having issues.
- **[Advanced](#) button** - Use this to set visibility of certain elements in reports and in diagram packages.

4.17.1.4.1 Element Visibility

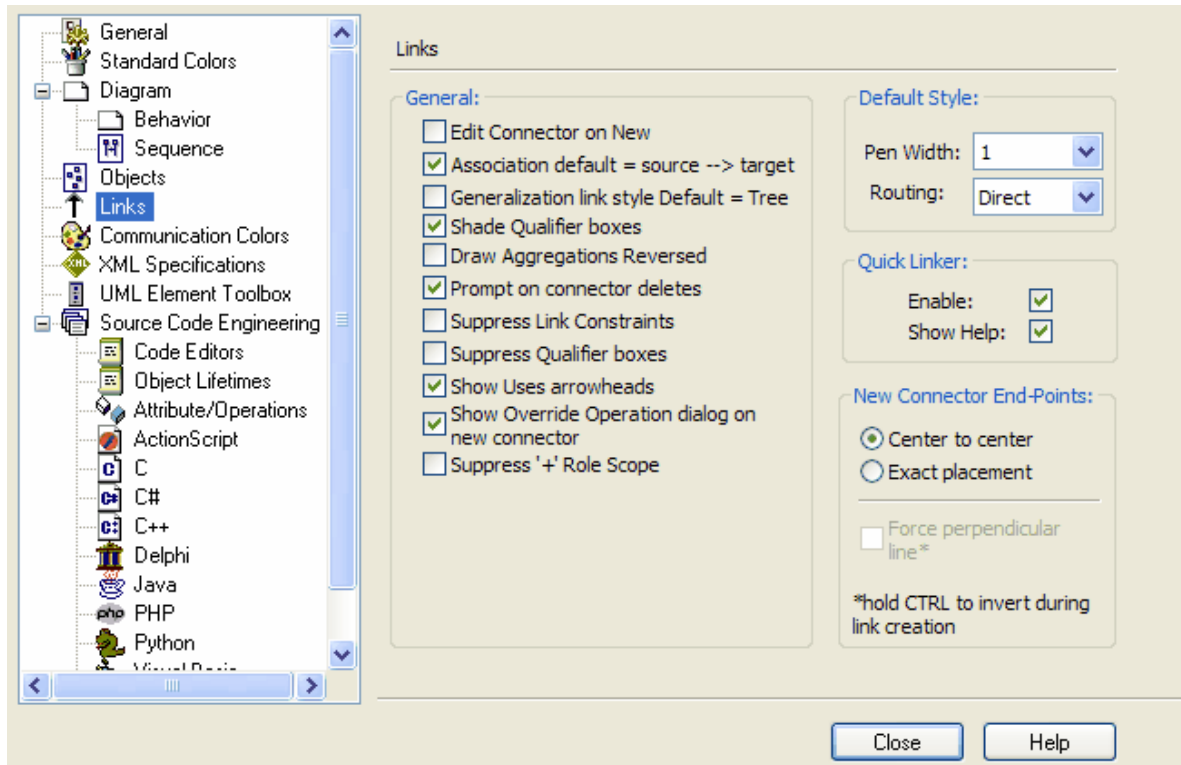
Some elements do not appear in packages and in RTF output by default. Select **Advanced** from the **Objects** section of the **Local Options** dialog to specify which elements should be visible.

See the section on [customizing element visibility](#) for more details.



4.17.1.5 Links

The *Links* section of the *Options* dialog as below provides options for the creation, behavior, and notation for links.



General

- **Edit Connector on New** - Automatically show the connector properties dialog when a new connector is added.
- **Association default = source -> target** - Set the direction of new associations to Source->Target (ie. with an arrow head at target).
- **Generalization link style Default = Tree** - If checked, generalizations will be shown as tree style hierarchies.
- **Shade Qualifier boxes** - Qualifier boxes are lightly shaded if this option is checked.
- **Draw Aggregations Reversed** - By default, aggregate and composite connectors are drawn in EA from Source to Target. In some modeling tools they are drawn in the opposite direction. When this option is set, EA will mimic those other tools. (nb All tools will have the parent as the target and the child as the source of the connector, that is a requirement of UML; only the direction of dragging the mouse to draw the connector is changed).
- **Prompt on connector deletes** -If this option is on the user will be prompted before deleting connectors.
- **Suppress Link Constraints** - If this option is on, links constraints will not appear in the diagram.
- **Suppress Qualifier boxes** - If this option is on, qualifiers will not appear in a box.
- **Show Uses arrowheads** - Show an arrowhead on Actor->Use Case associations.
- **Show Override Operation dialog on new connector** - Use this when adding generalizations and realizations between classes and interfaces. When ticked, the dialog will show automatically if the target element has overridable features.
- **Suppress '+' Role Scope** - ensure that the role and scope are not displayed on the diagram.

Default Style

- **Pen Width** - Default connector width.
- **Routing** - Default connector link style for new connectors.

Quick Linker

- **Enable** - Enable the Quick Linker
- **Show Help** - Adds a "help" menu item to the tail of the Quick Linker menu.

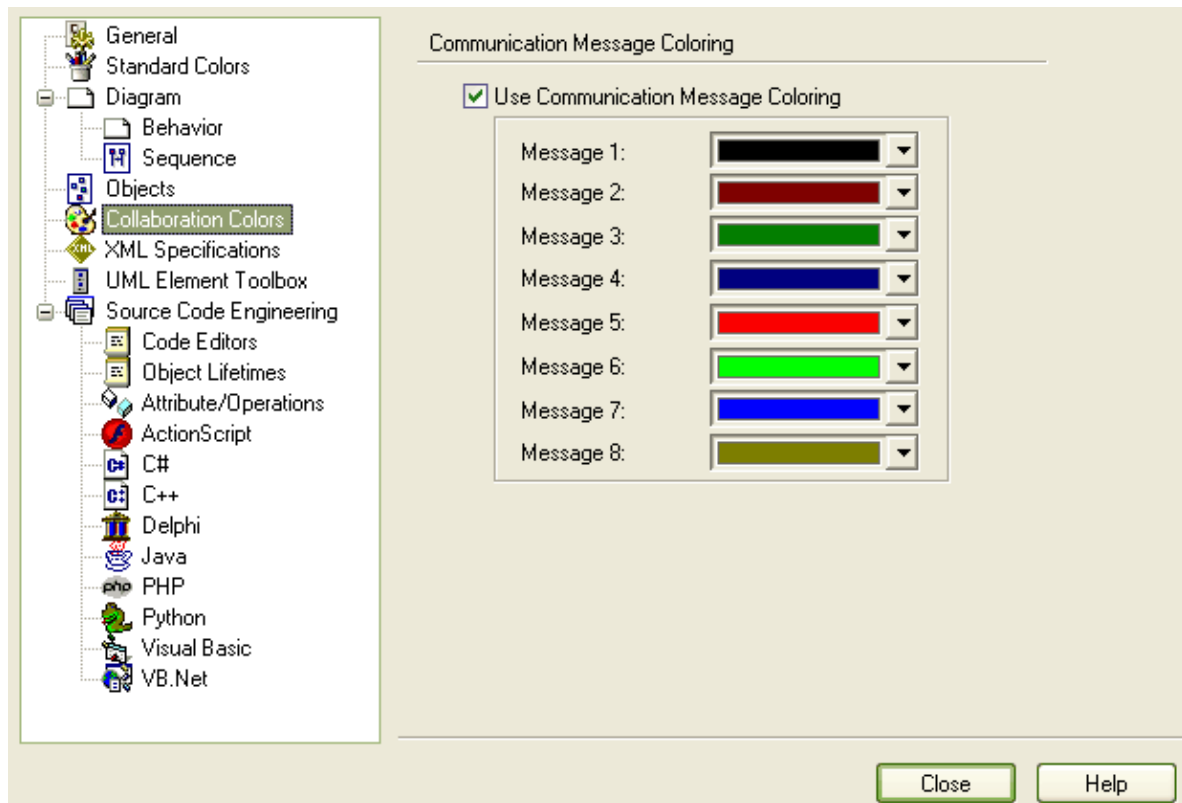
New Connector End-Points

These options affect where the dashed guide line for new connectors is positioned.

- **Center to center**
- **Exact placement**
- **Force perpendicular line**

4.17.1.6 Communication Colors

The *Communication Colors* settings in the *Options* dialog allow you to configure the colors used in collaboration diagrams. When you enable this option, collaboration messages will appear in different colors depending on the sequence group they belong to on a diagram - eg. 1.n are black, 2.n are red, 3.n are green etc.

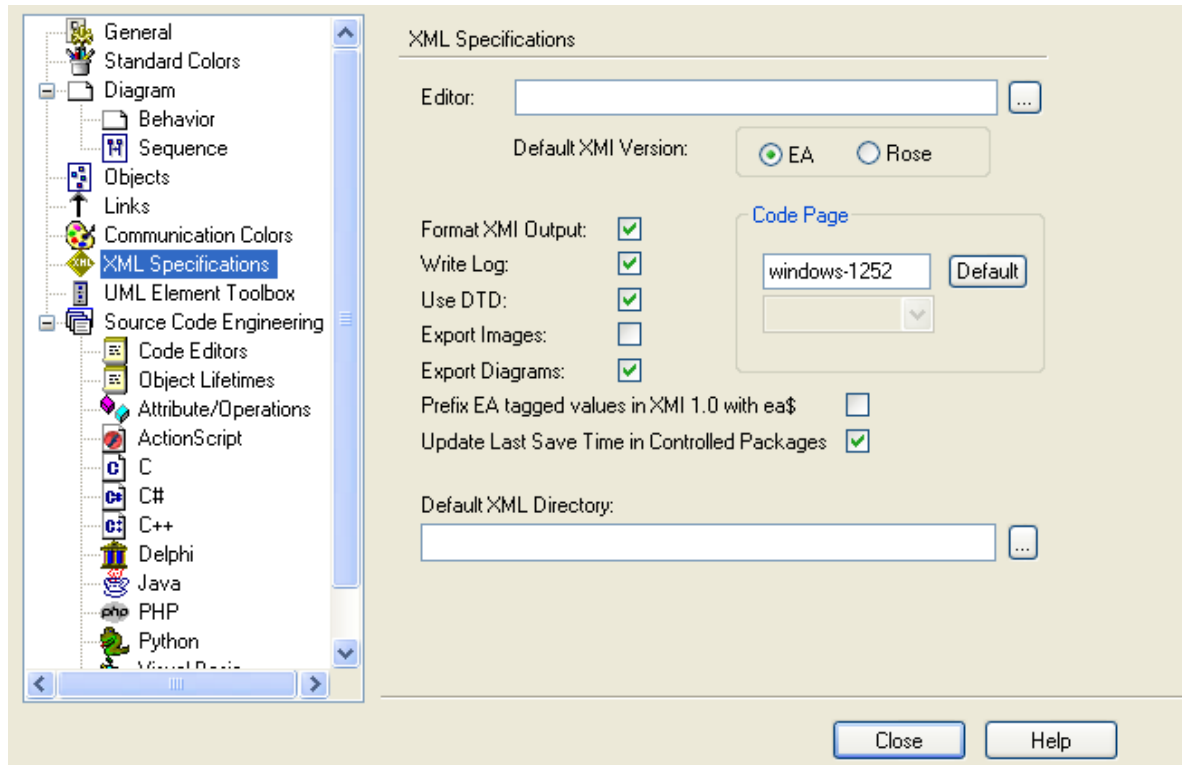


Check the *Use Communication Message Coloring* to turn on colored messages.

Set the color sequence to what you require - the pattern will repeat after 8 sequence groups.

4.17.1.7 XML Specifications

The *XML Specifications* section of the *Options* dialog is shown below. This allows you to configure various settings for working with XML.

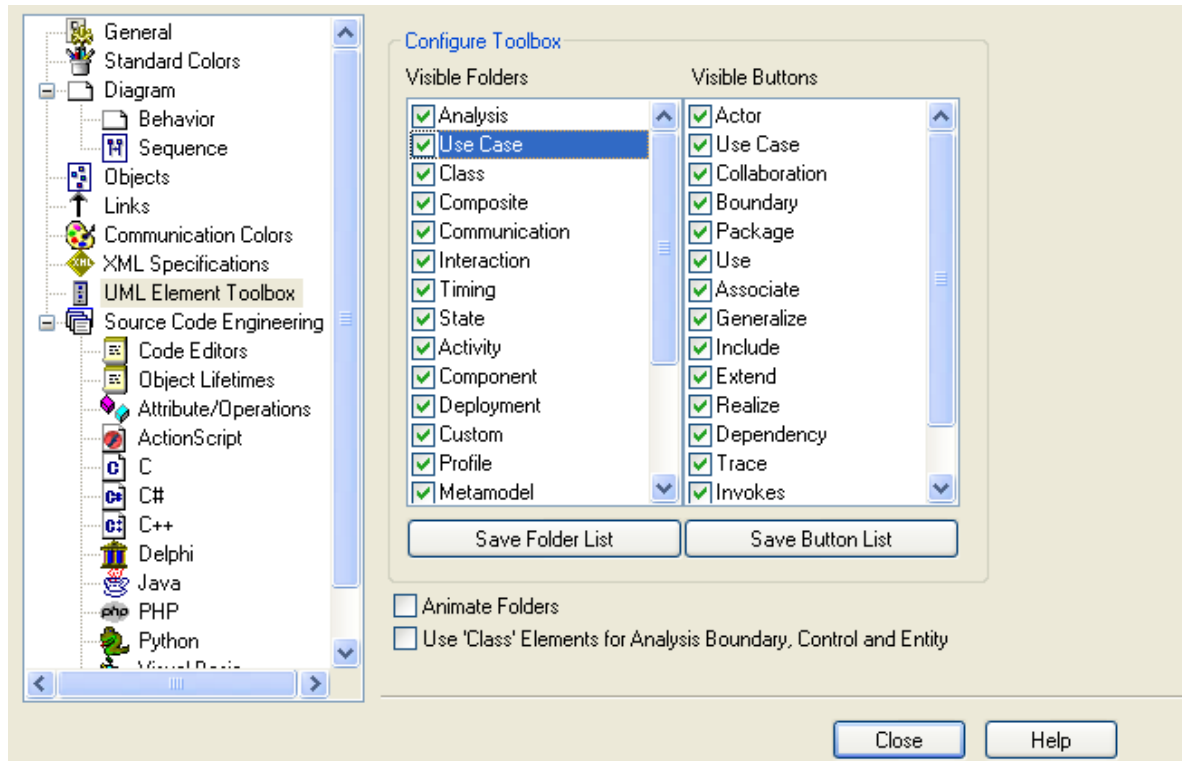


XML Specifications Settings:

- **Editor** - Set the default editor for XML documents you open within EA.
- **Default XML Version** - XML version to use - EA or Rose.
- **Format XML Output** - This option allows the user to determine whether formatting will be applied to your XML output.
- **Write Log** - Set whether to write to a log file when you import or export XML.
- **Use DTD** - Set whether to use DTD.
- **Export Images** - Set whether to export images when you export XML.
- **Export Diagrams** - Set whether to export diagrams when you export XML.
- **Prefix EA tagged values in XML 1.0 with ea\$** - Set whether to prefix any EA tagged values within any XML 1.0 you create with ea\$.
- **Update Last Save Time in Controlled Packages** - Set whether to update the time you last saved in controlled packages.
- **Code Page** - Sets the Code Page to use, setting a NULL encoding string will result in the encoding tag being entirely omitted from the XML output.
- **Default XML Directory** - Default XML directory to use when importing and exporting XML.

4.17.1.8 UML Element Toolbox

The *UML Element Toolbox* contains the model elements you will use in diagrams. You can configure the appearance and behavior of the toolbar.



Outlook Toolbar Settings:

- *Animate Folders* - When checked, the toolbar will be animated whenever a new category is selected.
- *Use 'Class' Elements for Analysis Boundary, Control and Entity* - Ensures that a class element is used for analysis boundaries, control and entities.

Assigning Visible Folders:

To assign the visibility of folders use the following procedure:

1. Highlight the folder of interest, and uncheck the check box to hide all of the entire button group under this category.
2. Press the *Save Folder List* button
3. Restart Application to have the new settings take effect.

Assigning Visible Buttons:

To assign the visibility of folders use the following procedure:

1. Highlight the folder of interest, and in the Visible buttons section uncheck the check boxes that apply to the buttons that you wish to hide from use.
2. Press the *Save Button List* button.

3. Restart the application to have the new settings take effect.

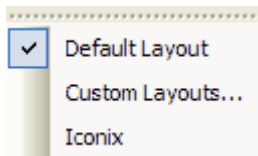
4.17.1.9 Source Code Engineering

Information about the Code Generation settings can be found in the [Code Engineering Settings](#) section.

4.17.2 Custom Layouts

EA supports some customization of the default desktop layout and toolbox folder sequence. This is useful for resetting your current layout to the standard layout, and for using custom layouts for working with specific processes.

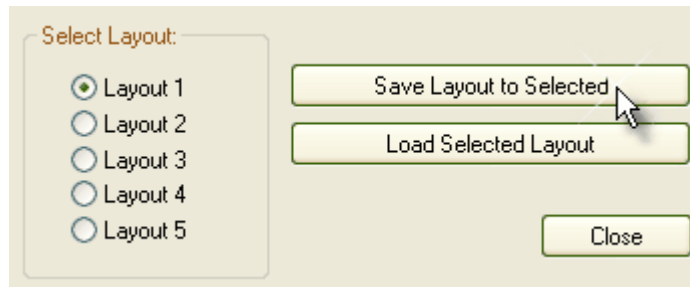
This feature is accessible from the *View | Visual Layouts* submenu.



When you select an option EA will realign toolbars and docked windows to the defaults for that option.

Setting User Layouts

If you have the layout of the EA windows and tools just as you want them, you can preserve your layout by saving it as a custom user layout. Go to *View | Visual Layouts | User Layouts* - this will open the *Custom Layouts* dialog as shown below:



You can save up to 5 custom layouts. To save the current layout to Layout 1 ensure the *Layout 1* option is selected, then press *Save Layout to Selected*.

If you want to make changes to an existing layout, you can update the currently saved layout by opening the *Custom Layouts* window, selecting the radio button for the layout you want to update, then pressing *Save Layout to Selected*.

Load a saved layout by going to the *Custom Layouts* window, selecting the layout you wish to load, and pressing *Load Selected Layout*.

Warning About Custom Layouts and Keyboard Shortcuts

If you have set keyboard shortcuts, these will not be overridden if you switch to the Default Layout or the Iconix Layout option.

However, if you have set keyboard shortcuts and you switch to a User Layout, your keyboard shortcuts will

be overridden, unless you have saved them as part of the User Layout you have switched to.

For more information about setting keyboard shortcuts, see the [Customize Keyboard](#) topic.

4.17.3 Visual Styles

You can configure the overall look and feel of EA to suit your working environment. Options range from a classic windows application to an enhanced XP look.

To change the appearance of EA

1. Open the **View** | **Visual** Style submenu
2. Select the style you want from the list:

| | |
|----------------|---|
| XP | Similar to applications like MS Office and MS Visual Studio |
| 2003 | colorful modern style |
| 2005 | Rounded modern style. |
| Classic | Classic windows application look. |

You can also [enable or disable menu shadows](#) and animation of [auto-hidden](#) windows.

Note: The 2005 style uses the [navigation compass](#) method for docking windows.

4.18 Using Add-In's

Add-In's enhance the existing functionality of Enterprise Architect through a variety of mechanisms such as [UML Profiles](#) and the [Automation Interface](#). Once an add-in is [registered](#), it can be managed using the [Add-In Manager](#).

See Also:

- [Register Add-In's](#)
- [The Add-In Manager](#)

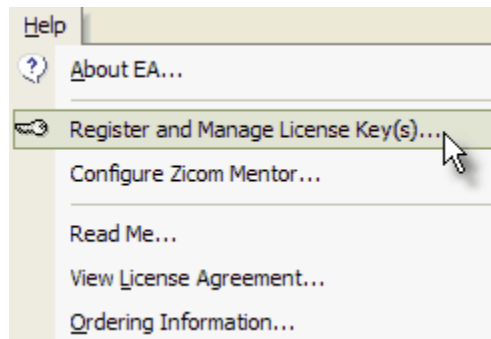
4.18.1 Register Add-In

You can register Add-ins for EA by using the following instructions:

Note: Zicom Mentor requires a different method of registration. To register your version of Zicom Mentor go to the [Registering Zicom Mentor](#) page.

Registering an Add-in for EA

1. Purchase one or more licenses of the Add-in from your Add-in provider.
2. Once you have paid for a licensed version of the Add-in, you will receive (via email or other suitable means) a licence key for the product.
3. Save the license key and the latest full version of the Add-in.
4. Run the Add-in's setup program to install the Add-in.
3. From the **Help** menu, select the **Register and Manage Licence Key(s)** option.



4. This will bring up the Licence Management Dialog, press the *Add Key button*.
5. The user will be presented with the *Enter Registration Key* dialog (enter the key that you received with the Add-in - **including the { and } characters**).

 A screenshot of the 'Enter Registration Key' dialog box. It has a light green background. At the top, there are two text input fields: 'Name: John Redfern' and 'Company: Sparx Systems'. Below these is a text area containing the registration key: '{ABCDE123-0389-4d1f-AA60-EE&6D-TFK1-8wJ1-IMNI-JACK-1C}'. At the bottom, there are three buttons: 'Register', 'Cancel', and 'Help'.

6. When the Add-in has been added successfully, close down EA and restart to apply the new changes.

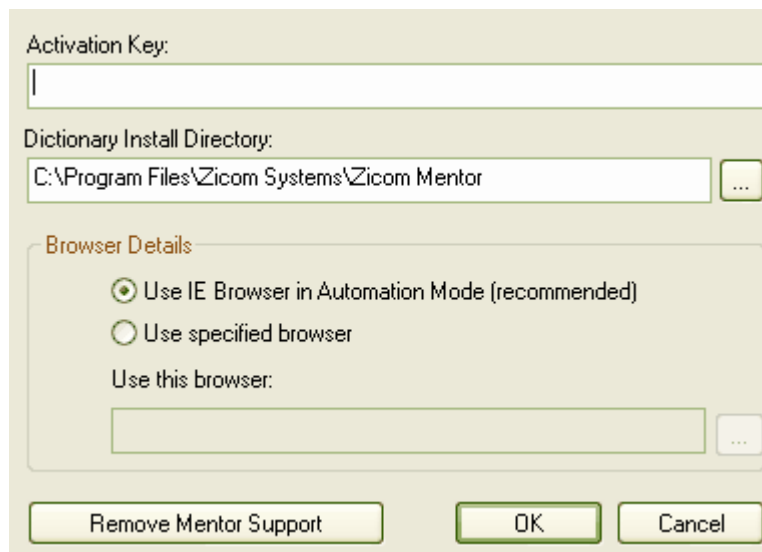
4.18.1.1 Register Zicom Mentor

[Zicom Mentor](#) is a detailed Visual Dictionary for UML by Zicom Systems. It is HTML based and integrates into Enterprise Architect, such that you can get context sensitive help on any UML Element directly from within EA. To use UML Mentor you will need to first purchase a license from Sparx Systems, then install UML Mentor and register the product in EA.

Integrating UML Mentor with EA

Integration can only be carried out if both Enterprise Architect and the Zicom Mentor Visual UML Dictionary have been installed.

1. Start Enterprise Architect.
2. From the *Help* menu, under select the *Configure Zicom Mentor* option.
3. Enter your Zicom Mentor serial number (use "evaluation" for the trial version).
4. Enter (or browse [...]) to the location where you installed Zicom Mentor.
5. Enter your Browser location.
6. Click *OK*.



Accessing Zicom Mentor from Enterprise Architect

The Visual UML Dictionary can be accessed directly in EA from the diagram context menu (right-click menu) or selected Project Browser item context menus.

1. Right click on a UML element or diagram.
2. From the context menu, select the *Mentor* option.

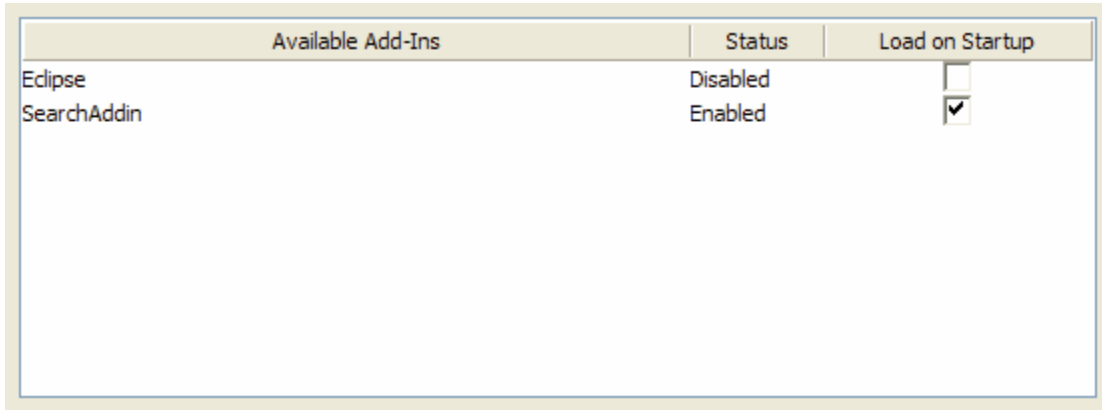
Removing Zicom Mentor Integration in Enterprise Architect

Zicom Mentor integration can be removed from Enterprise Architect at any time. Removal of the integration does not uninstall Zicom Mentor.

1. Start Enterprise Architect.
2. From the *Help* menu, select the *Configure Zicom Mentor* option.
3. Click *Remove Mentor Support*.

4.18.2 The Add-In Manager

The Add-In manager can be used to view and disable Add-In's.



Note: EA must be restarted for changes to take effect.

4.19 Keyboard Shortcuts

The table below lists the default keyboard shortcuts functions within EA:

| Function | Shortcut | Category |
|---|--------------------|----------|
| Create a new Enterprise Architect project | Ctrl + N | File |
| Open an Enterprise Architect project | Ctrl+ O | File |
| Print the active diagram | Ctrl + P | File |
| Reload the current project | Ctrl + Shift + F11 | File |
| Add a single element to the clipboard list | Ctrl + Space | Edit |
| Paste element as link | Shift + Insert | Edit |
| Paste element as new | Ctrl + Shift + V | Edit |
| Bookmark current element with red marker | Shift + Space | Edit |
| Delete selected element(s) | Ctrl + D | Edit |
| Search for elements in the project | Ctrl + F | Edit |
| Paste element(s) from the clipboard as a link | Shift + Insert | Edit |
| Paste element as new element in model | Ctrl + Shift + V | Edit |
| Autohide the current window | Ctrl + Shift + F4 | View |
| Close the current window | Ctrl + F4 | View |
| View Hierarchy window | Ctrl + Shift + 4 | View |
| View Maintenance window | Alt + 4 | View |
| View Notes window | Ctrl + Shift + 1 | View |
| View Project Browser | Alt + 0 | View |
| View Properties window | Alt + 1 | View |
| View Relationships window | Ctrl + Shift + 2 | View |
| View Resource Browser | Alt + 6 | View |
| View Requirements and Constraints window | Ctrl + Shift + 3 | View |
| View Scenarios window | Ctrl + Shift + 5 | View |

| | | |
|---|------------------|---------|
| View Source Code window | Alt + 7 | View |
| View System window | Alt + 2 | View |
| View Tagged Values window | Ctrl + Shift + 6 | View |
| View Testing window | Alt + 3 | View |
| Display the UML toolbox | Alt + 5 | View |
| View Project Management Properties window | Ctrl + Shift + 7 | View |
| View Output window | Ctrl + Shift + 8 | View |
| View Instant Help window | Ctrl + Shift + 9 | View |
| Generate selected classes | Shift + F11 | Project |

| Function | Shortcut | Category |
|---|-----------------------|----------|
| Import package from XMI | Ctrl + Alt + I | Project |
| Import package from directory | Ctrl + Shift + U | Project |
| Manage locks applied by current user | Ctrl + Shift + L | Project |
| Configure package control | Ctrl + Alt + P | Project |
| Create documentation | F8 | Project |
| Synchronize current element | Ctrl + R or F7 | Project |
| Synchronize package contents | Ctrl + Alt + M | Project |
| Generate HTML Report | Shift + F8 | Project |
| Add new package to project | Ctrl + W | Project |
| Transform Current Package | Ctrl + Shift + H | Project |
| Generate Diagrams-only Report | Ctrl + Shift + F8 | Project |
| Generate a single class | Ctrl + G or F11 | Project |
| Generate C++ source code | Ctrl + Alt + K | Project |
| Transform Selected Elements | Ctrl + Alt + F | Project |
| Export Package to XMI | Ctrl + Alt + E | Project |
| Import and Export to CSV files | Ctrl + Alt + C | Project |
| Manage Package Baselines | Ctrl + Alt + B | Project |
| Diagram properties | F5 | Diagram |
| Repeat last connector | F3 | Diagram |
| Save | Ctrl + S | Diagram |
| Save image to clipboard | Ctrl + B | Diagram |
| Save image to file | Ctrl + T | Diagram |
| Set visible relations | Ctrl + Shift + I | Diagram |
| Repeat last element | Shift + F3 | Diagram |
| Insert a new diagram | Ctrl + Y | Diagram |
| Set focus to current view | Ctrl + Shift + O | Diagram |
| Space elements evenly horizontally | Alt + - | Element |
| Add attribute | Ctrl + Shift + F9 | Element |
| Add operation | Ctrl + Shift + F10 | Element |
| Add other | Ctrl + F11 | Element |
| Edit element attributes | F9 | Element |
| Auto size selected elements | Alt + Z | Element |
| Align bottom edges of selected elements | Ctrl + Alt + Down | Element |
| Configure element appearance | Ctrl + Shift + E | Element |
| Delete selected feature from model | Ctrl + Shift + Delete | Element |
| Edit selected | F2 | Element |

| | | |
|---|-----------------------|---------|
| Align selected elements on left boundaries | Ctrl + Alt + Left | Element |
| Align selected elements on right boundaries | Ctrl + Alt + Right | Element |
| Space selected elements evenly | Alt + = | Element |
| Manage embedded elements | Ctrl + Shift + B | Element |
| Insert new feature after current selection | Ctrl + Shift + Insert | Element |
| Locate in browser | Alt + G | Element |
| New element | Ctrl + M | Element |
| View source code in default editor | Ctrl + E or F12 | Element |

| Function | Shortcut | Category |
|---|-------------------|---------------|
| Operation | F10 | Element |
| Override inherited features | Ctrl + Shift + O | Element |
| Configure element properties | Alt + Enter | Element |
| Project resourcing, metrics and risk | Ctrl + Shift + K | Element |
| Select alternate image | Ctrl + Shift + W | Element |
| Specify feature visibility | Ctrl + Shift + Y | Element |
| Set element parent or implement interface(s) | Ctrl + I | Element |
| Set references to other elements and diagrams | Ctrl + J | Element |
| View element usage | Ctrl + U | Element |
| Add tagged value | Ctrl + Shift + T | Element |
| Align top edges of selected elements | Ctrl + Alt + Up | Element |
| View properties dialog | Enter | Element |
| Check data integrity | Shift + F9 | Tools |
| Configure system options | Ctrl + F9 | Tools |
| Spell check current package | Ctrl + Shift + F7 | Tools |
| Spell check model | Ctrl + F7 | Tools |
| Edit code generation templates | Ctrl + Shift + P | Configuration |
| Edit transformation templates | Ctrl + Alt + H | Configuration |

Part

5

5 The UML Language

Enterprise Architect's modeling platform is based on the Unified Modeling Language (UML). This language is designed to be flexible, extendable and comprehensive, but generic enough to serve as a foundation for all system modeling needs. With its specification, there is a wide range of elements characterized by the diagrams they serve, and the attributes they provide. All can be further specified by using stereotypes, tags, and profiles. EA also provides additional custom diagrams and elements, to address further modeling interests. This section is intended to provide an introduction to EA's diagrams, elements, connectors and its modeling process, along with illustrating its alignment, when applicable, to the Unified Modeling Language.

This section is divided into three parts:

- [UML Diagrams](#)
- [UML Elements](#)
 - [Behavioral Diagram Elements](#)
 - [Structural Diagram Elements](#)
- [UML Connections](#)

5.1 What is UML

The Unified Modeling Language

Enterprise Architect's modeling platform is based on the Unified Modeling Language (UML), a standard that defines rules and notations for specifying business and software systems. The notation supplies a rich set of graphic elements for modeling object oriented systems, and the rules state how those elements may be connected and used. UML is not a tool for creating software systems. Instead, it is a visual language for communicating, modeling, specifying and defining systems.

This language is designed to be flexible, extendable and comprehensive, yet generic enough to serve as a foundation for all system modeling needs. With its specification, there is a wide range of elements characterized by the kinds of diagrams they serve, and the attributes they provide. All can be further specified by using stereotypes, tags, and profiles.

Wide range of Application

Although initially conceived as a language for software development, UML may be used to model a wide range or real world domains. For example, UML can be used to model many real world Processes (in business, science, industry, education and elsewhere), Organizational Hierarchies, Deployment maps and much more.

EA also provides additional custom diagrams and elements, to address further modeling interests. This section is intended to provide an introduction to EA's diagrams, elements, connectors and its modeling process, along with illustrating its alignment, when applicable, to the Unified Modeling Language.

Extending UML for new Domains

Using UML Profiles, UML Patterns, Grammars, Data types, Constraints and other extensions, UML and EA may be tailored to address a particular modeling Domain not explicitly covered in the original UML specification. EA makes extending the UML simple and straightforward, and best of all, the extension mechanism is still part of the UML Specification.

Find out more

UML is a standard specified by the Object Management Group. Further information, including the full UML 2.0 documentation can be found on their website at <http://www.omg.org>

See Also

- [UML Diagrams](#)
- [UML Elements](#)

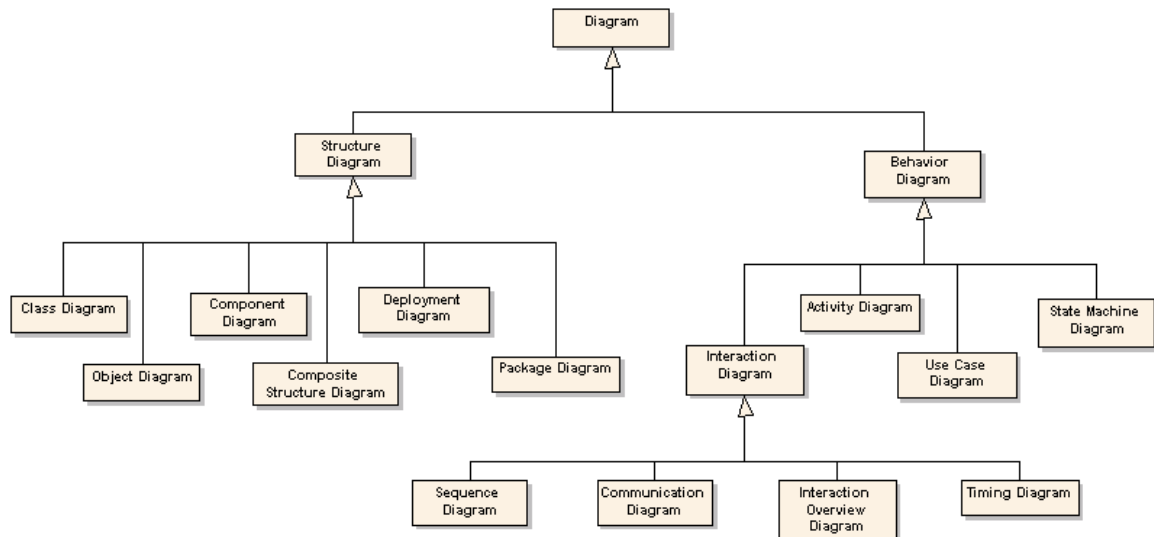
- [UML Connections](#)
- [Modeling with UML](#)

5.2 UML Diagrams

Types of Diagram

The figure below shows the taxonomy of the 13 UML diagrams, as defined by the Object Development Group's UML 2.0 specification. As can be seen, there are two major groupings of diagrams: [Structural diagrams](#) which show a static view of the model; and [Behavioral diagrams](#) which show a dynamic view of the model. For more insight into building these diagrams, click on the links:

- **Structural Diagrams**
 - [Class Diagrams](#)
 - [Object Diagrams](#)
 - [Component Diagrams](#)
 - [Composite Structure Diagrams](#)
 - [Deployment Diagrams](#)
 - [Package Diagrams](#)
- **Behavioral Diagrams**
 - [Interaction Diagrams](#)
 - [Sequence Diagrams](#)
 - [Communication Diagrams](#)
 - [Interaction Overview Diagrams](#)
 - [Timing Diagrams](#)
 - [Activity Diagrams](#)
 - [Use Case Diagrams](#)
 - [State Machine Diagrams](#)



Refer to figure 464 (*UML 2.0 Superstructure*, p. 590).

Working with Diagrams

Diagrams are developed in the main workspace in which you create and connect model elements. They are

created by right-clicking a package and selecting *New Diagram*, or loaded by double-clicking their diagram icon in the Project Browser. For full details on how to work with diagrams, see [Common Diagram Tasks](#).

Extended Diagrams

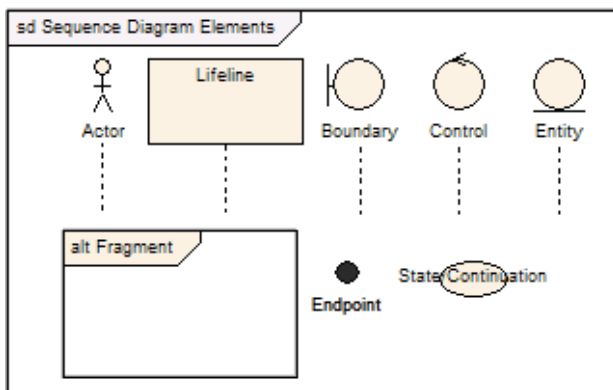
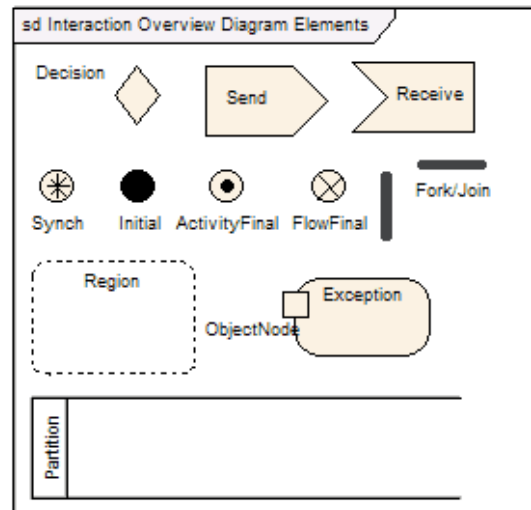
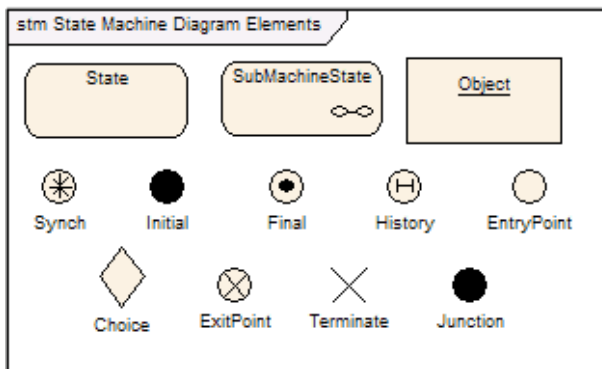
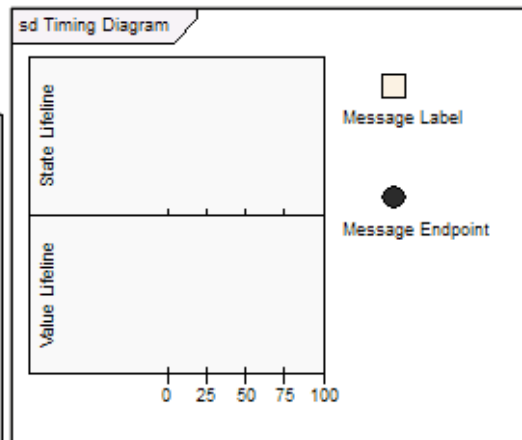
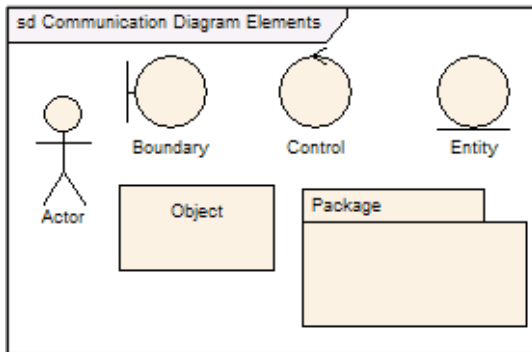
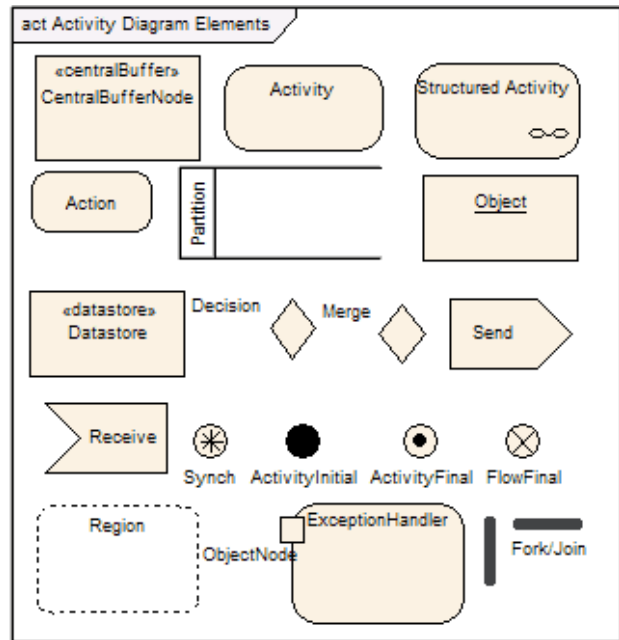
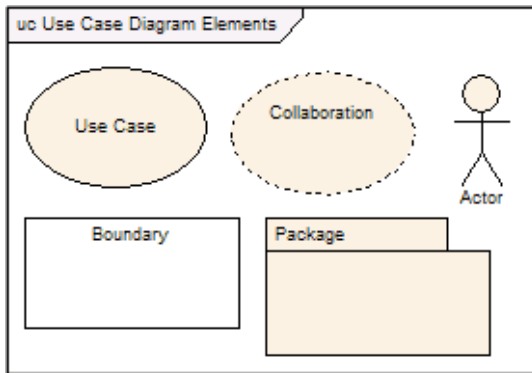
In addition to the 13 UML diagrams, EA provides the following Extended diagrams:

- [Analysis](#) diagrams
- [Custom](#) diagrams
- [Requirements diagrams](#)
- [Maintenance diagrams](#)
- [User Interface diagrams](#)
- [Database](#) diagrams
- [Robustness](#) diagrams

5.2.1 Behavioral Diagrams

Behavioral diagrams depict the behavioral features of a system or business process. Behavioral diagrams include:

- [Activity diagrams](#)
- [Use Case diagrams](#)
- [State Machine diagrams](#)
- [Timing diagrams](#)
- [Sequence diagrams](#)
- [Communication diagrams](#)
- [Interaction Overview diagrams](#)



5.2.1.1 Activity Diagram

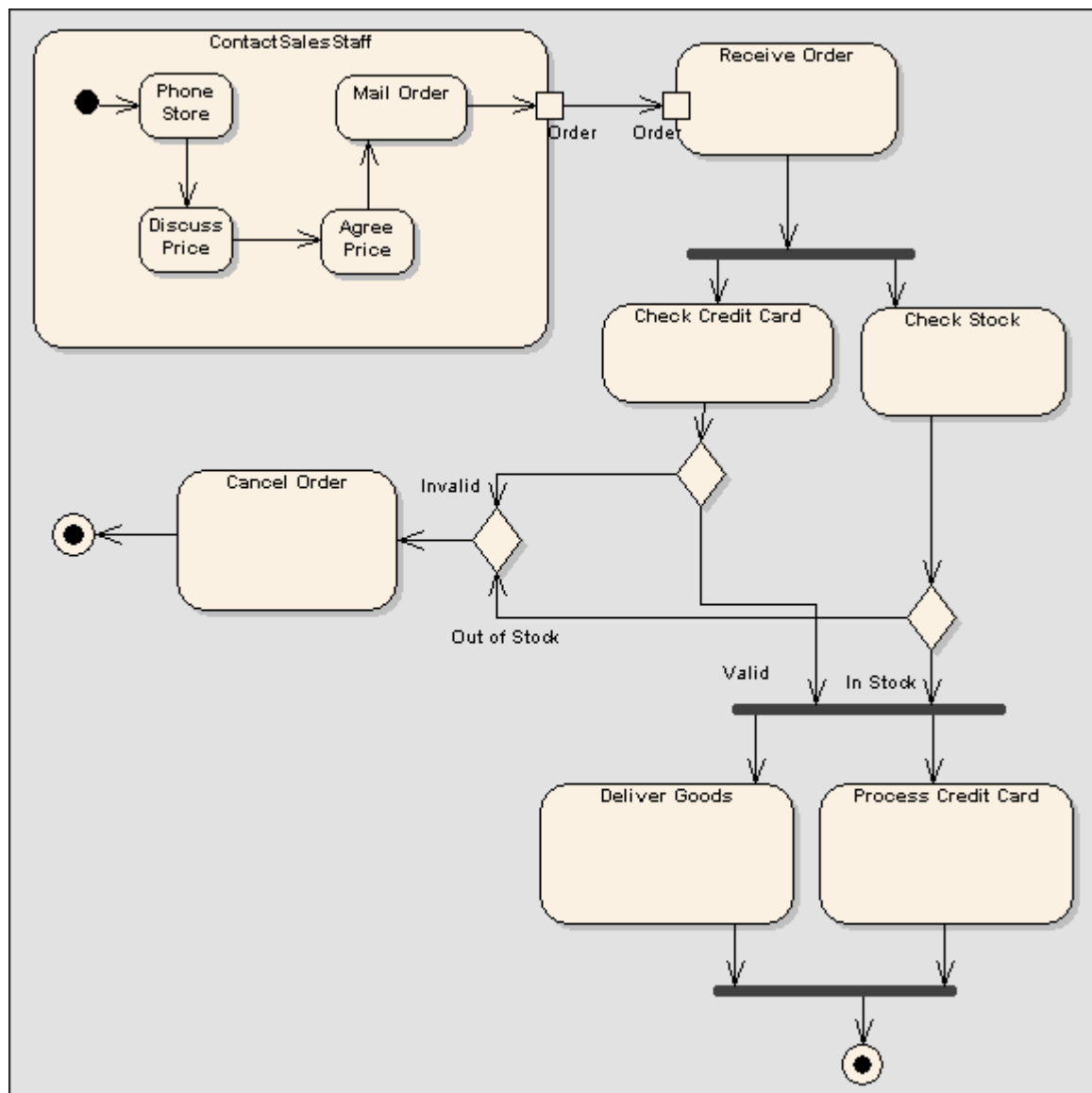
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

Activity diagrams are used to model the behaviors of a system, and the way in which these behaviors are related in an overall flow of the system. The logical paths a process follows, based on various conditions, concurrent processing, data access, interruptions and other logical path distinctions, are all used to construct a process, system or procedure.

Note: Analysis diagrams (Simplified Activity), containing the elements most useful for business process modeling, can be created using the [New Diagram](#) dialog.






















Example Diagram

The diagram below illustrates some of the features of Activity diagrams, including activities, actions, start nodes, end nodes and decision points.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Activity Diagram Elements | Activity Diagram Connectors |
|---|--|
|  Activity |  Fork/Join |
|  Structured Activity |  Fork/Join |
|  Action |  Control Flow |
|  Partition |  Object Flow |
|  Object |  Dependency |
|  Datastore |  Interrupt Flow |
|  Decision | |
|  Send | |
|  Receive | |
|  Synch | |
|  Initial | |
|  Final | |
|  Flow Final | |
|  Region | |
|  Exception | |

Related Topics

- [Analysis Diagram](#)
- [Time Event](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 284) states:

"An activity specifies the coordination of executions of subordinate behaviors, using a control and data flow model. The subordinate behaviors coordinated by these models may be initiated because other behaviors in the model finish executing, because objects and data become available, or because events occur external to the flow. The flow of execution is modeled as activity nodes connected by activity edges. A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents. Activity nodes also include flow-of-control constructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to operations that are directly invoked.

"Activities may describe procedural computation. In this context, they are the methods corresponding to operations on classes. Activities may be applied to organizational modeling for business process engineering and workflow. In this context, events often originate from inside the system, such as the finishing of a task, but also from outside the system, such as a customer call. Activities can also be used for information system modeling to specify system level processes.

"Activities may contain actions of various kinds:

- occurrences of primitive functions, such as arithmetic functions.
- invocations of behavior, such as activities.
- communication actions, such as sending of signals.
- manipulations of objects, such as reading or writing attributes or associations.

"Actions have no further decomposition in the activity containing them. However, the execution of a single action may induce the execution of many other actions. For example, a call action invokes an operation which is implemented by an activity containing actions that execute before the call action completes."

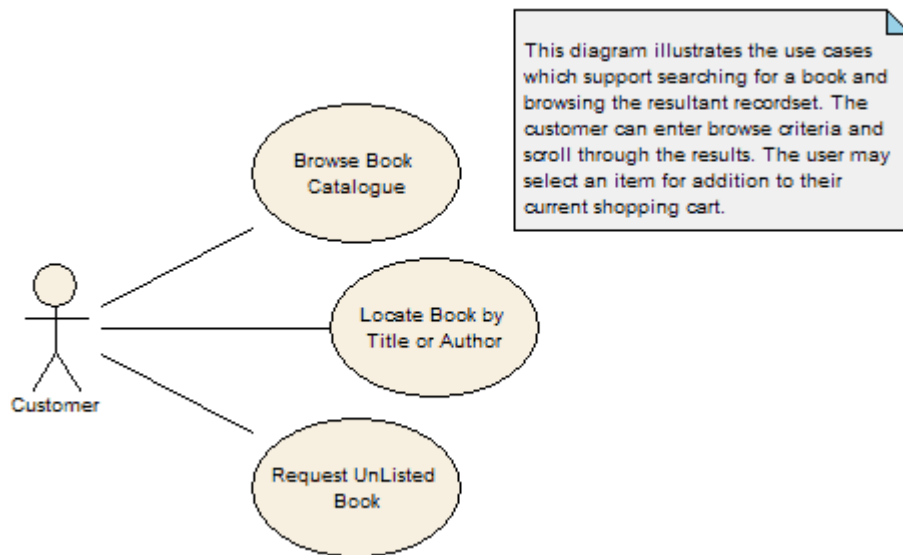
5.2.1.2 Use Case Diagram

[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

A Use Case diagram captures use cases and actor interactions. It describes the functional requirements of the system, the manner that outside things (actors) interact at the system boundary and the response of the system.









Example Diagram



The diagram below illustrates some features of Use Case diagrams:



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Use Case Diagram Elements | Use Case Diagram Connectors |
|---|--|
|  Actor |  Use |
|  Use Case |  Associate |
|  Collaboration |  Generalize |
|  Boundary |  Include |

| | |
|---|--|
|  Package |  Extend |
| |  Realize |
| |  Dependency |
| |  Trace |

Related Topics

- [Use Case Extension Points](#)
- [Using Rectangle Notation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

"A diagram that shows the relationships among actors and the subject (system), and use cases."

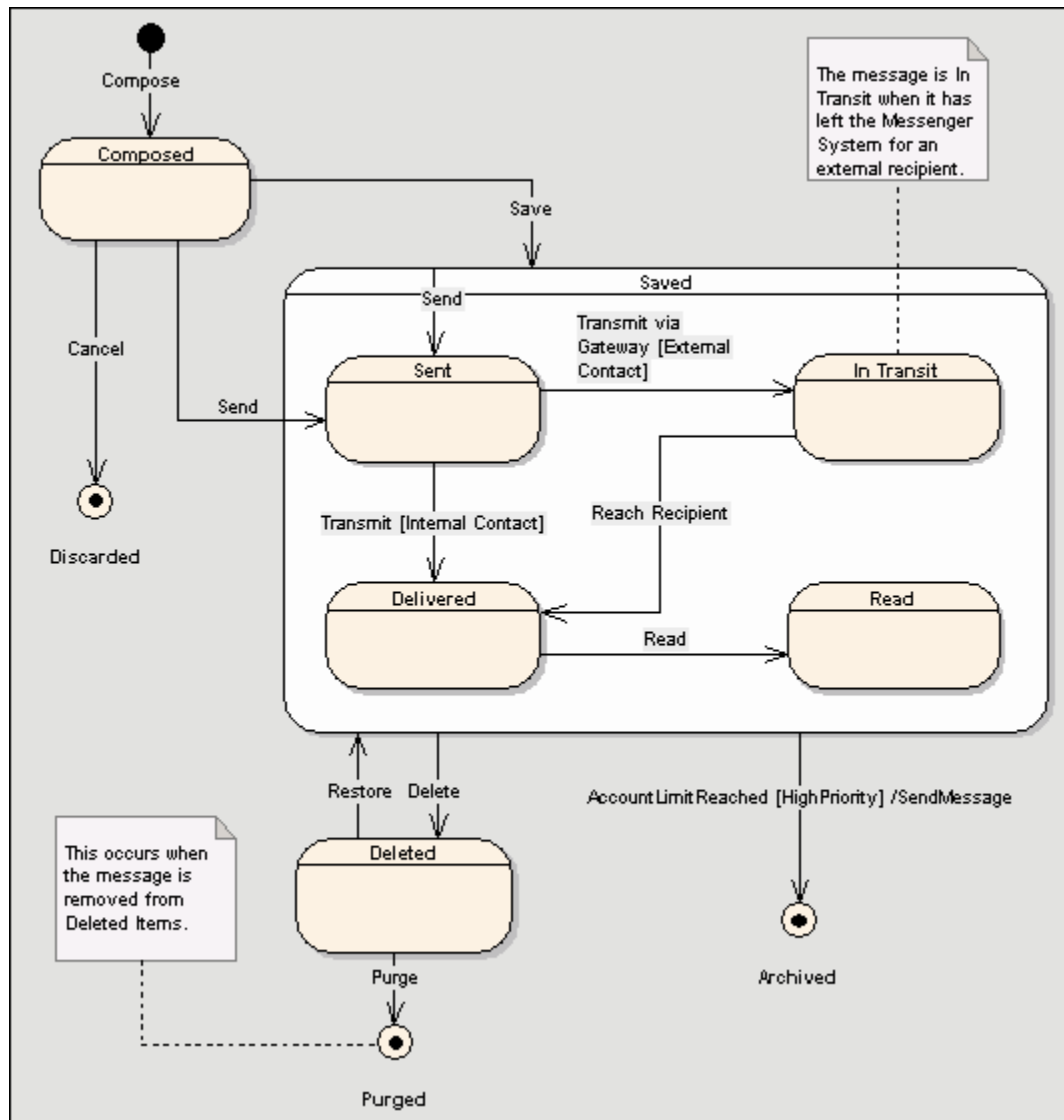
5.2.1.3 State Machine Diagram (formerly State Diagram)

[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

A *State Machine diagram* illustrates how an element, often a class, can move between states classifying its behavior, according to transition triggers, constraining guards and other aspects of State Machine diagrams that depict and explain movement and behavior.

Example Diagram






The diagram below illustrates some features of State Machine diagrams. The *Saved* state is a composite state, and enclosed states are sub-states. Initial and final pseudo-states indicate the entry to and exit from the state machine.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| State Machine Diagram Elements | State Machine Diagram Connectors |
|--------------------------------|----------------------------------|
| State | Fork/Join |
| SubMachineState | Fork/Join |
| Initial | Transition |
| Final | Object Flow |
| History | |
| Synch | |

| | | |
|--|--|--|
|  Object | | |
|  Decision | | |
|  Junction | | |
|  Entry | | |
|  Exit | | |

Related Topics

- [Regions](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 490) states:

"A state machine owns one or more regions, which in turn own vertices and transitions.

"The behaviored classifier context owning a state machine defines which signal and call triggers are defined for the state machine, and which attributes and operations are available in activities of the state machine. Signal triggers and call triggers for the state machine are defined according to the receptions and operations of this classifier. As a kind of behavior, a state machine may have an associated behavioral feature (specification) and be the method of this behavioral feature. In this case the state machine specifies the behavior of this behavioral feature. The parameters of the state machine in this case match the parameters of the behavioral feature and provide the means for accessing (within the state machine) the behavioral feature parameters. A state machine without a context classifier may use triggers that are independent of receptions or operations of a classifier, i.e. either just signal triggers or call triggers based upon operation template parameters of the (parameterized) statemachine."

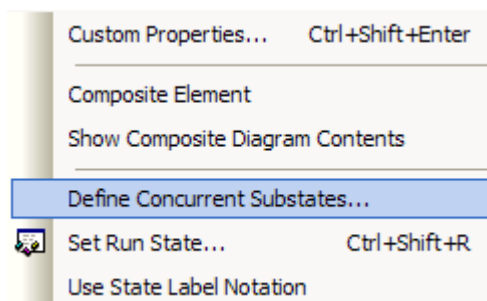
5.2.1.3.1 Regions

[Related Topics](#) | [OMG UML Specification](#)

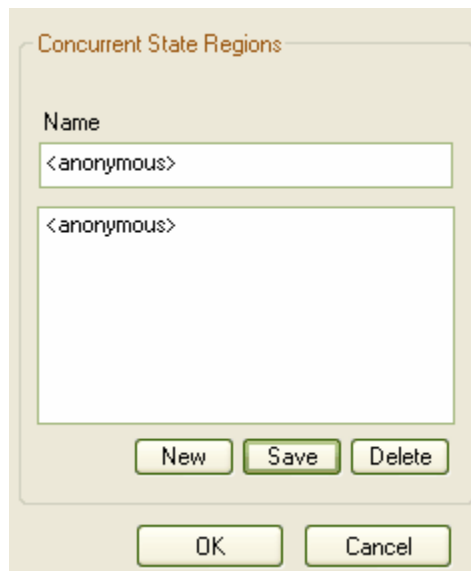
Regions can be created in composite states or state machines. Regions indicate concurrency, such that a single state is active in each region. Multiple transitions can occur from a single event dispatch, so long as similarly triggered transitions are divided by regions.

To create a region in a composite state:

1. **Right-click** on a state, and select **Advanced Settings** which will open the child menu:



3. Select the **Define Concurrent Substates** command. This will open the **State Regions** dialog (below). Construct the regions of a state, which can be named or anonymous. Press **OK**.



Related Topics

- [Composite State](#)
- [State Machine Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 476) states:

"A region is an orthogonal part of either a composite state or a state machine. It contains states and transitions."

5.2.1.3.2 Pseudo-States

Pseudo-states are a UML 2 abstraction for various types of transient vertices used in State Machine diagrams. Pseudo-states are used to express complex transition paths. For more details on the kinds of pseudo-states, consult the list below.

Common Usage

- [Initial](#)
- [Entry Point](#)
- [Exit Point](#)
- [Choice](#)
- [Junction](#)
- [History](#)
- [Terminate](#)
- [Final](#)
- [Fork](#)
- [Join](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 469) states:

"A pseudostate is an abstraction that encompasses different types of transient vertices in the state"

machine graph... Pseudostates are typically used to connect multiple transitions into more complex state transitions paths. For example, by combining a transition entering a fork pseudostate with a set of transitions exiting the fork pseudostate, we get a compound transition that leads to a set of orthogonal target states."

5.2.1.4 Interaction Diagrams

An interaction is a generalization for a type of interaction diagram. Interaction diagrams can be any of the following:

- [Timing Diagrams](#)
- [Sequence Diagrams](#)
- [Interaction Overview Diagrams](#)
- [Communication Diagrams](#)

5.2.1.4.1 Timing Diagram

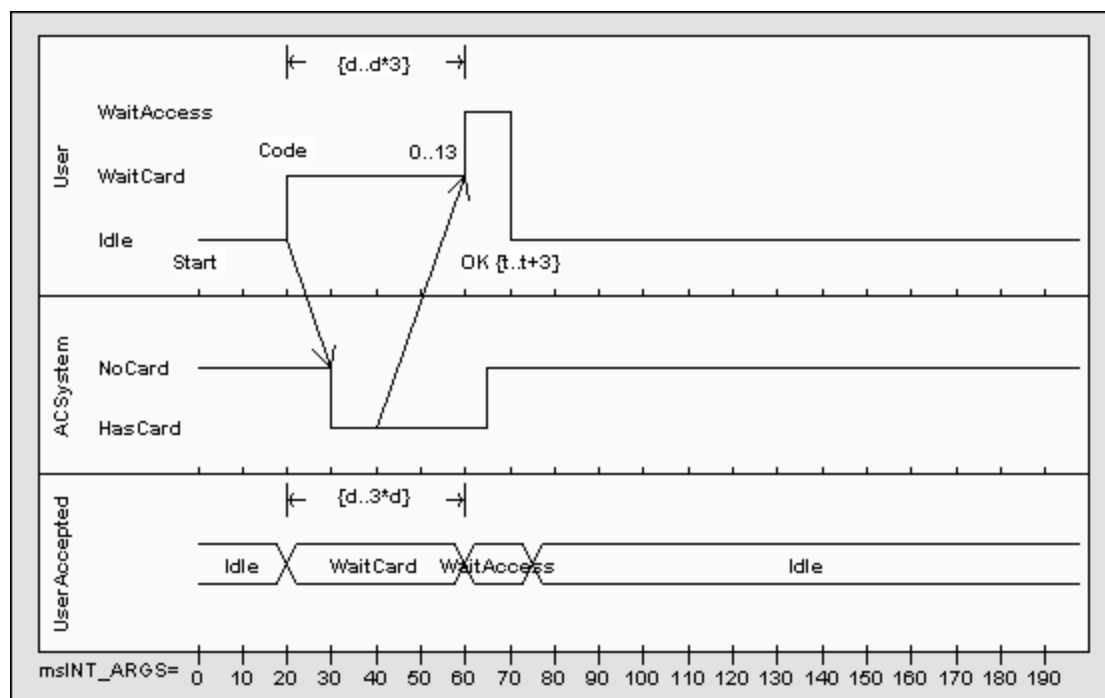
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

The *Timing diagram* defines the behavior of different objects within a time-scale. It provides a visual representation of objects changing state and interacting over time.

Timing diagrams can be used for defining hardware-driven or embedded software components. This may be embedded software such as that used in a fuel injection system, a microwave controller, etc. They can also be used for specifying time-driven business processes.

Example Diagram







Below is an example Timing diagram:



Refer to figures 351 and 352 (*UML 2.0 Superstructure*, p. 454).

Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Timing Diagram Elements | Timing Diagram Connectors |
|--|----------------------------------|
|  State Lifeline | |
|  Value Lifeline | |
|  Message Label | |
|  Message Endpoint | |
|  Diagram Gate | |
|  Message | |

Related Topics

- [Timing Message](#)
- [State Lifeline](#)
- [Value Lifeline](#)
- [Timing Message](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 450) states:

"Timing Diagrams are used to show interactions when a primary purpose of the diagram is to reason about time. Timing diagrams focus on conditions changing within and among Lifelines along a linear time axis. Timing diagrams describe behavior of both individual classifiers and interactions of classifiers, focusing attention on time of occurrence of events causing changes in the modeled conditions of the Lifelines."

The OMG UML specification (*UML 2.0 Superstructure*, p. 453) also states:

"The primary purpose of the timing diagram is to show the change in state or condition of a lifeline (representing a Classifier Instance or Classifier Role) over linear time. The most common usage is to show the change in state of an object over time in response to accepted events or stimuli. The received events are annotated as shown when it is desirable to show the event causing the change in condition or state."

5.2.1.4.2 Sequence Diagram

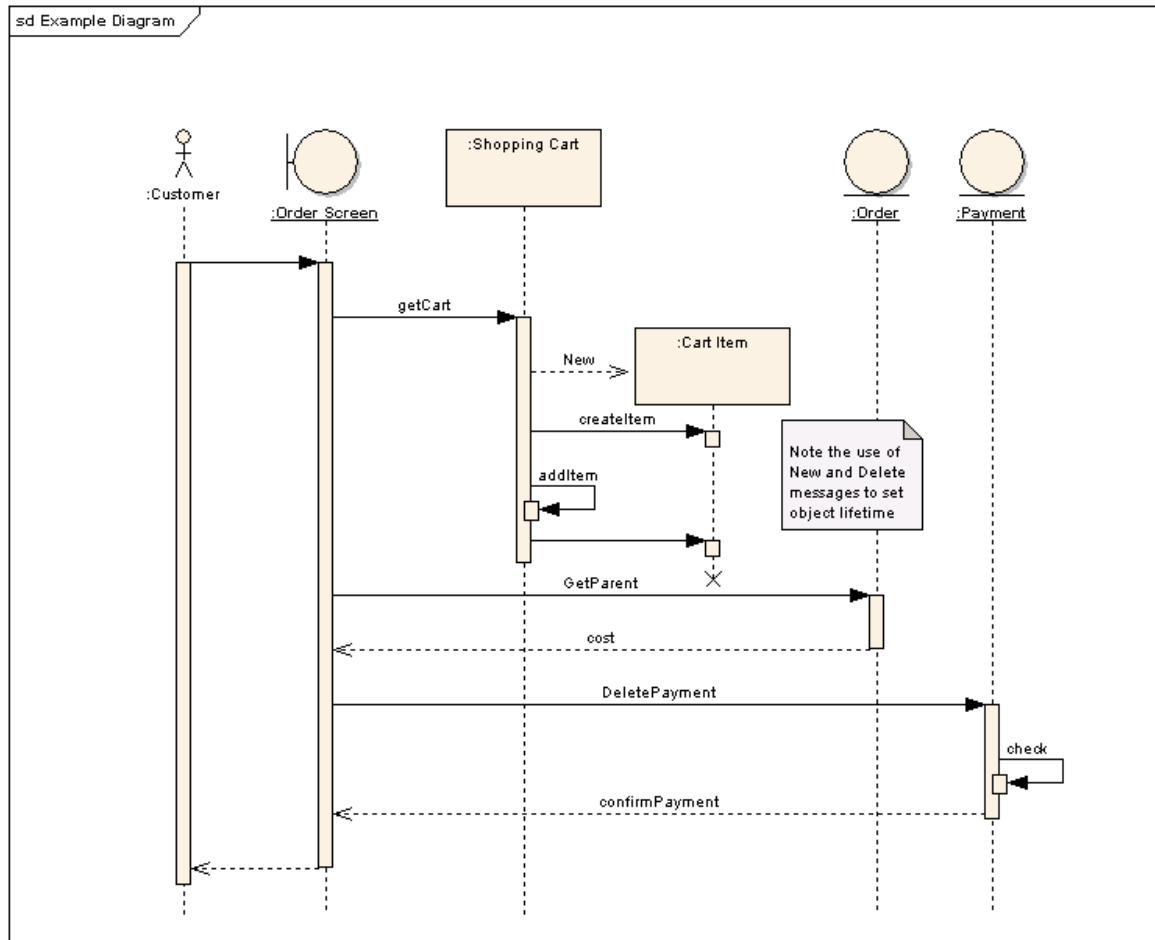
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

A *Sequence diagram* is a structured representation of behavior as a series of sequential steps over time. It is used to depict work flow, message passing and how elements in general cooperate over time to achieve a result.

- Each [sequence element](#) is arranged in a horizontal sequence, with messages passing back and forward between elements.
- An actor element may be used to represent the user initiating the flow of events.
- Stereotyped elements, such as [boundary](#), [control](#) and [entity](#), may be used to illustrate screens, controllers, and database items, respectively.
- Each element has a dashed stem called a lifeline, where that element exists and potentially takes part in the interactions.

Example Diagram

Below is an example Sequence diagram demonstrating several different elements:



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Sequence Diagram Elements | Sequence Diagram Connectors |
|---------------------------|-----------------------------|
| Actor | Message |
| Lifeline | Self-Message |
| Boundary | Recursion |
| Control | Call |
| Entity | |
| Fragment(s) | |
| Endpoint | |
| Diagram Gate | |

| | |
|--|--|
| <input type="radio"/> State/Continuation | |
|--|--|

Related Topics

- [Denote the Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Message Label Visibility](#)
- [Changing the Top Margin](#)
- [Changing the Timing Details](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 14) states:

"A diagram that depicts an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding event occurrences on the lifelines."

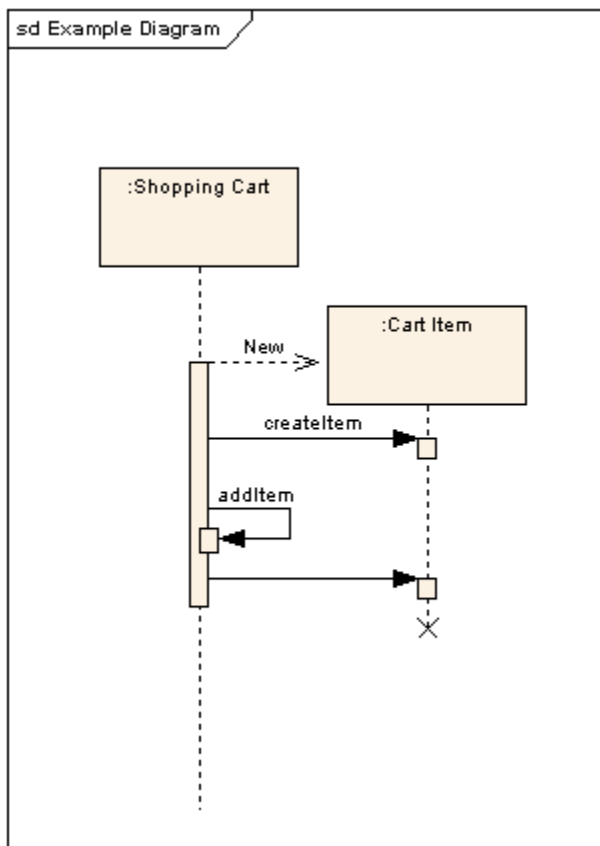
"Unlike a communication diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and communication diagrams express similar information, but show it in different ways."

5.2.1.4.2.1 Denote Lifecycle of an Element

You can capture element lifetimes using messages which are denoted as *New* or *Delete* message types. To do this, follow the steps below:

1. Right-click on a message within a Sequence diagram to view the context menu.
2. Select the *Message Properties...* option to view the *Message Properties* dialog.
3. From the *Lifecycle* selection menu, choose *New* or *Delete*.
4. Press *OK* to save changes.

The example below shows two elements that have specific creation and deletion times. (*Note: to show the termination "X" on the lifeline in the example below, you will need to switch on garbage collection: [Tools](#) | [Options](#) | [Diagram](#) | [Sequence](#) | [Garbage Collect](#)*).



Related Topics

- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)
- [Changing the top margin](#)
- [Changing the Timing Details](#)

5.2.1.4.2.2 Layout of Sequence Diagrams

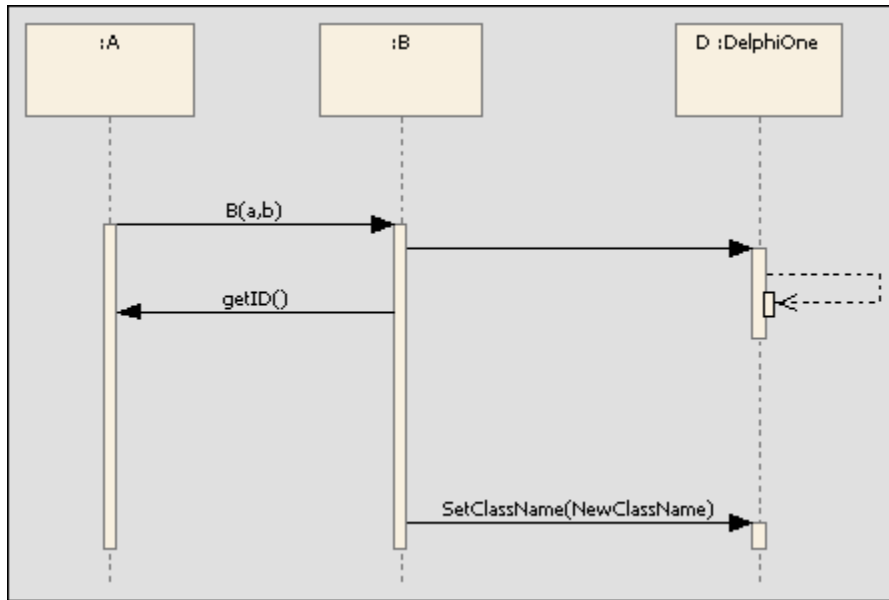
Related Topics

You can modify the vertical height of sequence messages to get an attractive and effective layout. To offset message positions, follow the steps below:

1. Select the appropriate message in a sequence diagram.
2. Use the mouse to drag the message up or down as required.

As you drag a message up or down a lifeline, any messages or fragments below that message are shifted up or down the same amount. However, be aware that if you drag up or down past the next or previous message, EA will interpret that as the desire to swap positions, rather than simply offset a message position.

The example below shows an economical use of space in a Sequence diagram.



Related Topics

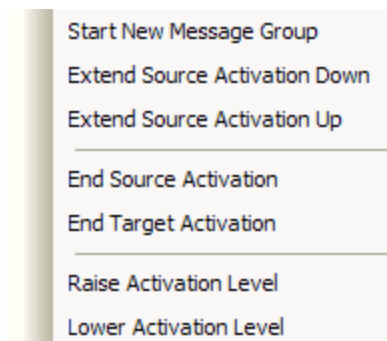
- [Denote Lifecycle of an Element](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)
- [Message Label Visibility](#)

5.2.1.4.2.3 Sequence Element Activation

[Related Topics](#)

Sequence elements in a [Sequence diagram](#) have *Activation* rectangles drawn along their lifelines. These rectangles describes the time the element is active during the overall period of processing. This visual representation can be suppressed by right-clicking the sequence diagram, and selecting "Suppress Activations".

In general, EA will calculate this period of Activation for you - but in some cases you may wish to fine tune the rectangle length. There are several context menu options on a sequence message that you can use to accomplish this. To access the following context menu, *right-click* on the message and select the Activations sub-menu.



Start New Message Group: will start off a new round of processing in the current diagram. This allows you to describe more than one processing scenario in a single diagram.

Extend Source Activation Down: Use this option to force an element to stay active beyond the normal processing period. This could be used to express an element which continues its own processing concurrently with other processes.

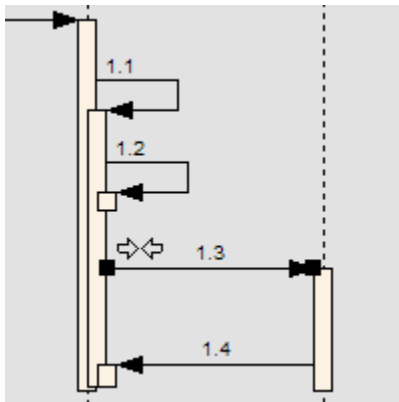
Extend Source Activation Up: Use this option to force an element's activation upwards.

End Source Activation: This option will truncate the activation of the source element after the current message. This is useful where you need to express an asynchronous message after which the source element becomes idle.

End Target Activation: Use this to end a Forced Activation started by the "Extend Source Activation" command.

The commands **Raise Activation Level** and **Lower Activation Level** will appear on the context menu only where they can be used. So, for example, after a self-message the next message will start by default at a lower activation level but will provide an **Raise Activation Level** command on its context menu to allow you to raise its level. **Lower Activation Level** also appears where appropriate.

A more convenient way to change activation levels is directly on the diagram. Whenever appropriate, left and/or right arrows will appear on selected connectors. See connector 1.3. Click the arrow to raise or lower the activation level.



Note: Program flow can more accurately be depicted with nested activation levels for callback messages.

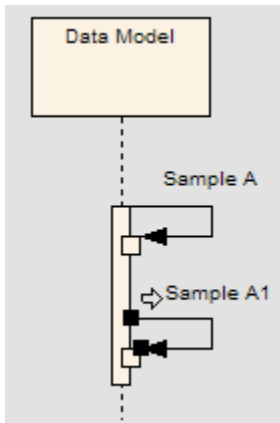
Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)

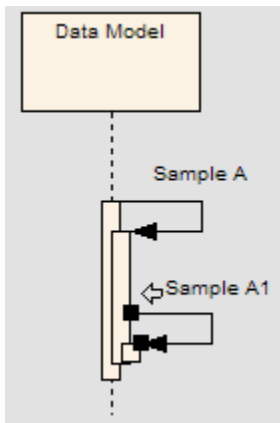
5.2.1.4.2.4 Lifeline Activation Levels

Related Topics

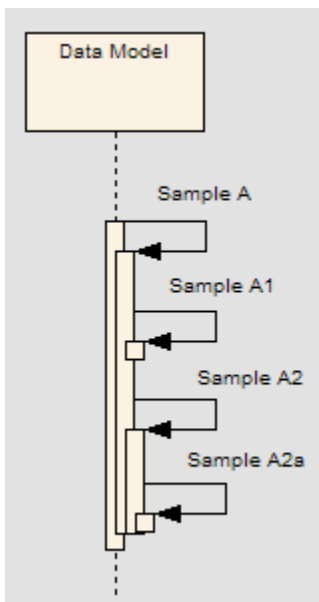
Complicated processing systems can be easily negotiated and reflected in Sequence diagrams, by adding activation layers on a single lifeline. For example, the below class invokes the method sample A, which in turn calls sample A1. By dragging the self-message connectors onto the lifeline, the following will appear:



In order to raise the Activation level of Sample A1, click on the raise arrow of the selected connector. The lifeline now visually depicts that method Sample A1 is called during the processing of Sample A.



In the example below, a few more self-messages have been added. The message Sample A2a is called from Sample A2 which in turn is called from Sample A, not Sample A1. Sample A1 is called from Sample A.



Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)
- [Message Label Visibility](#)

5.2.1.4.2.5 Sequence Elements[Related Topics](#)

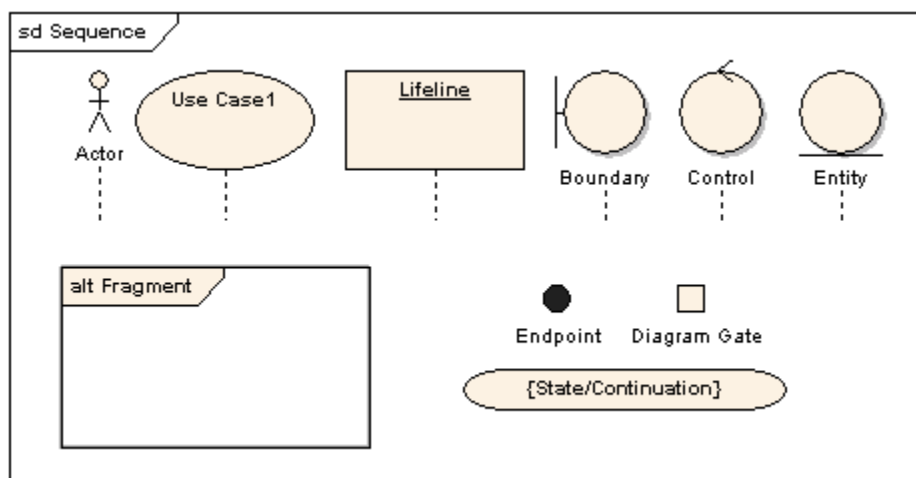
A [Sequence diagram](#) models a dynamic view of the interactions between model elements at runtime.

Sequence diagrams are commonly used as explanatory models for use case scenarios. By creating a sequence diagram with an actor and elements involved in the use case, you can model the sequence of steps the user and the system undertake to complete the required tasks. An element in a Sequence diagram is usually either an actor (the stimulus that starts the interaction) or collaborating elements.

Note: A Sequence diagram is often attached directly under a use case to which it refers. This helps keep elements together, both in the model and when documentation is produced. To do this, right-click the use case on the diagram and select **Composite Element**. Alternatively, from the Project Browser, right-click the use case and select **New Child Diagram**.

The example below shows some possible elements of sequence diagrams and their stereotyped display.

- **Actor** - An instance of an actor at runtime.
- **Use Case** - An instance of a use case at runtime
- **Object** - A standard element - not typed.
- **Boundary** - Represents a user interface screen or input/output device
- **Entity** - A persistent element - typically implemented as a database table or element.
- **Controller** - The active component that controls what work gets done, when and how.



Tip: Use Sequence diagrams early in analysis to capture the flow of information and responsibility throughout the system. Messages between elements will eventually become method calls in the class model.

Related Topics

- [Denote Lifecycle of an Element](#)

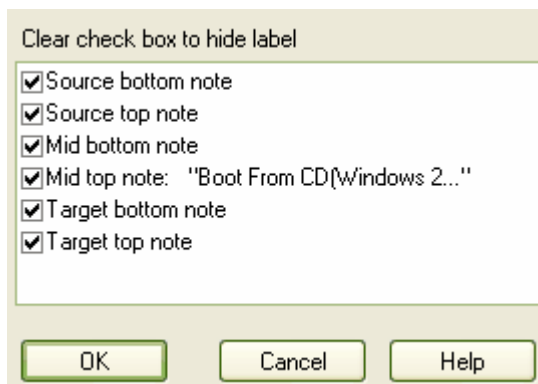
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Diagram](#)
- [Changing the Top Margin](#)
- [Changing the Timing Details](#)

5.2.1.4.2.6 Sequence Message Label Visibility

[Related Topics](#)

Sequence messages may have the control of the label visibility controlled via the message context menu. To hide and show the labels used in sequence messages use the following instructions:

1. Right click on the message within the sequence diagram.
2. From the message context menu select *Set Label Visibility*.
3. Show or hide message labels by checking and clearing the label from the list as appropriate.



Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Diagram](#)
- [Changing the Top Margin](#)
- [Changing the Timing Details](#)

5.2.1.4.2.7 Changing the Top Margin

In order to change the Top Margin from the default 50 units, right click on a sequence diagram and select *Set Top Margin*. Top Margins may be set anywhere from 30 to 250 units.

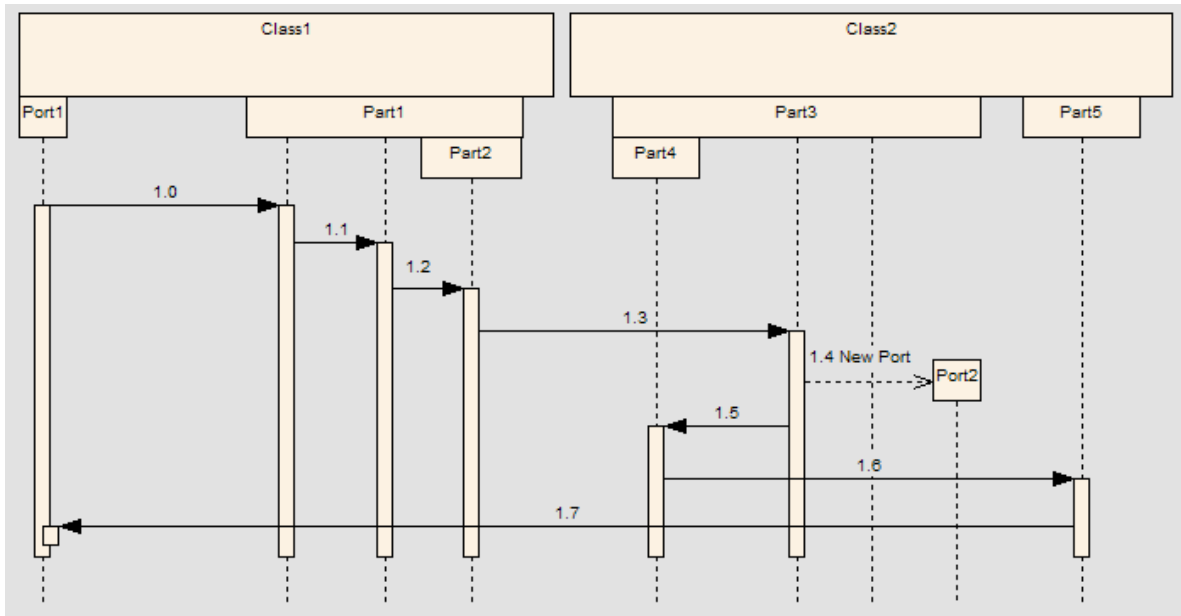
Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Diagram](#)

- [Changing the Top Margin](#)
- [Changing the Timing Details](#)

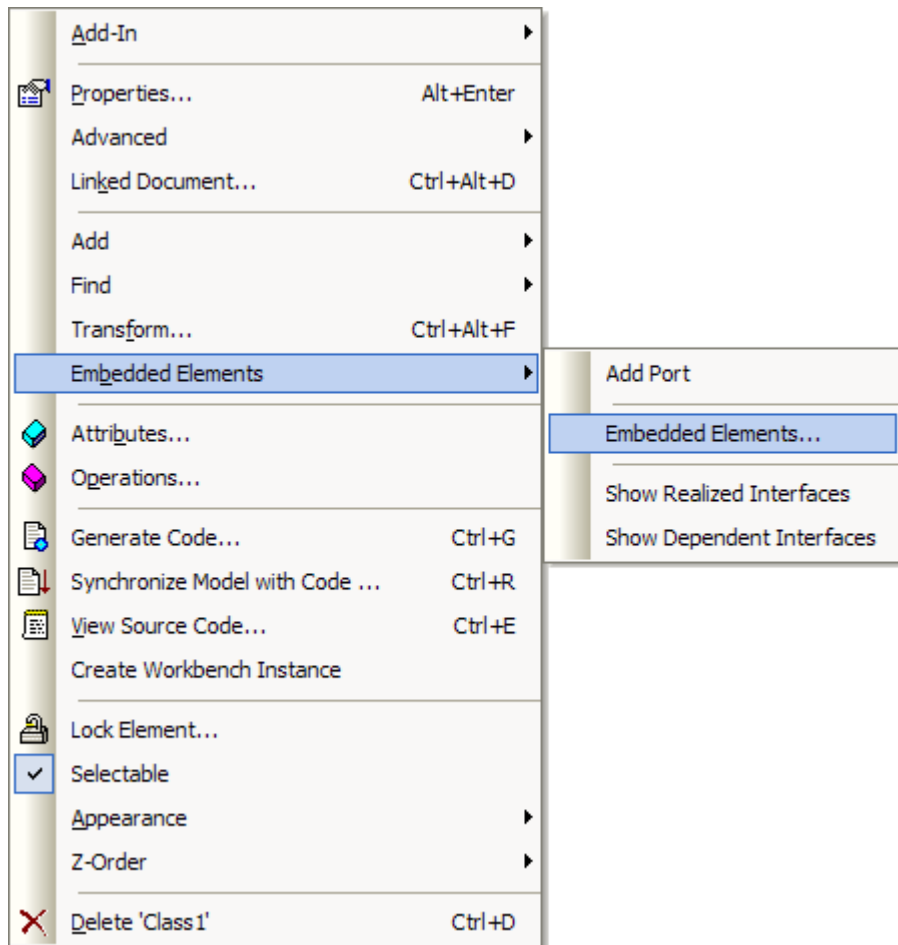
5.2.1.4.2.8 *Inline Sequence Elements*

It is possible to represent Part and Port elements on a sequence diagram. Child Parts and ports appear as inline sequence elements under their parent class sequence element.



1. *Right Click* on the sequence elements containing the Child Ports or Parts and select *Embedded Elements | Embedded Elements...*

2. Check the Parts or Ports you wish to show and close the dialog.



5.2.1.4.3 Communication Diagram (formerly Collaboration Diagram)

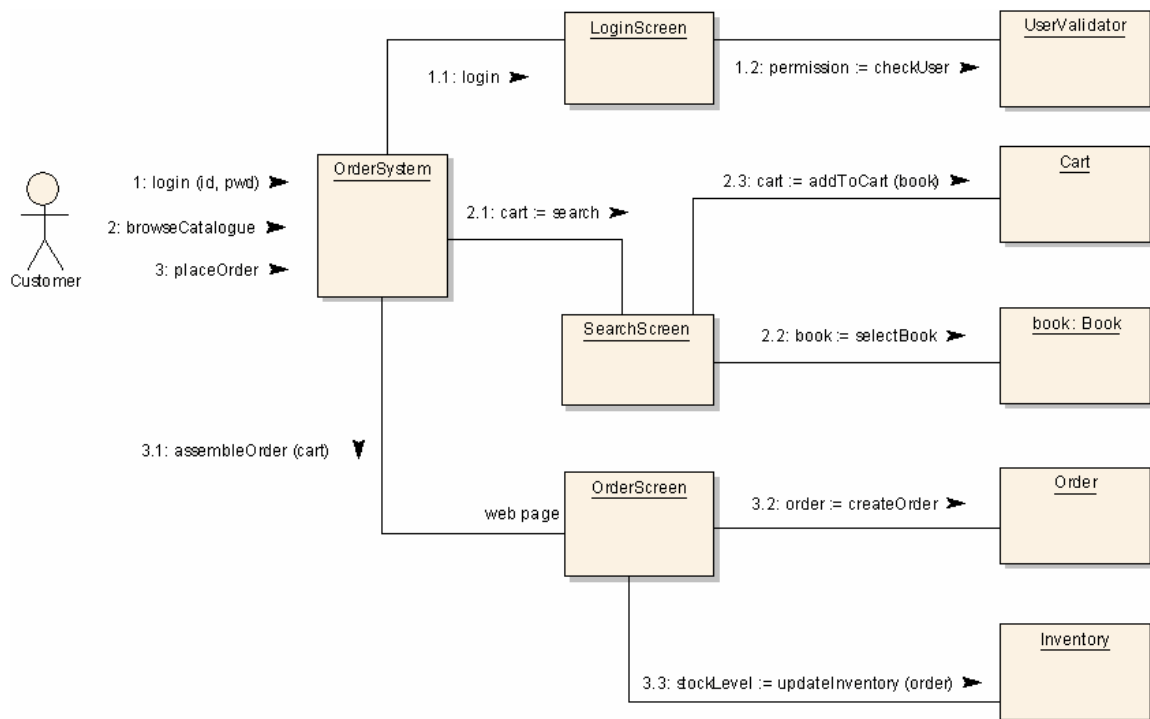
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

A *Communication diagram* shows the interactions between elements at run-time in much the same manner as a Sequence diagram. However, Communication diagrams are used to visualize inter-object relationships, while Sequence diagrams are more effective at visualizing processing over time.

Communication diagrams employ ordered, labeled associations to illustrate processing. Numbering is important to indicate the order and nesting of processing. A numbering scheme could be 1, 1.1, 1.1.1, 1.1.2, 1.2, etc. A new number segment begins for a new layer of processing, and would be equivalent to a method invocation.

Example Diagram

The example below illustrates a Communication diagram among cooperating object instances. Note the use of message levels to capture related flows.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Communication Diagram Elements | Communication Diagram Connectors |
|--------------------------------|----------------------------------|
| Actor | Associate |
| Object | Dependency |
| Boundary | Realize |
| Control | Nesting |
| Entity | |
| Package | |

Related Topics

- [Messages for Communication Diagrams](#)
- [Communication Diagrams in Color](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 14) states:

"A diagram that focuses on the interaction between lifelines where the architecture of the internal structure and how this corresponds with the message passing is central. The sequencing of messages is given through a sequence numbering scheme. Sequence diagrams and communication diagrams express similar information, but show it in different ways."

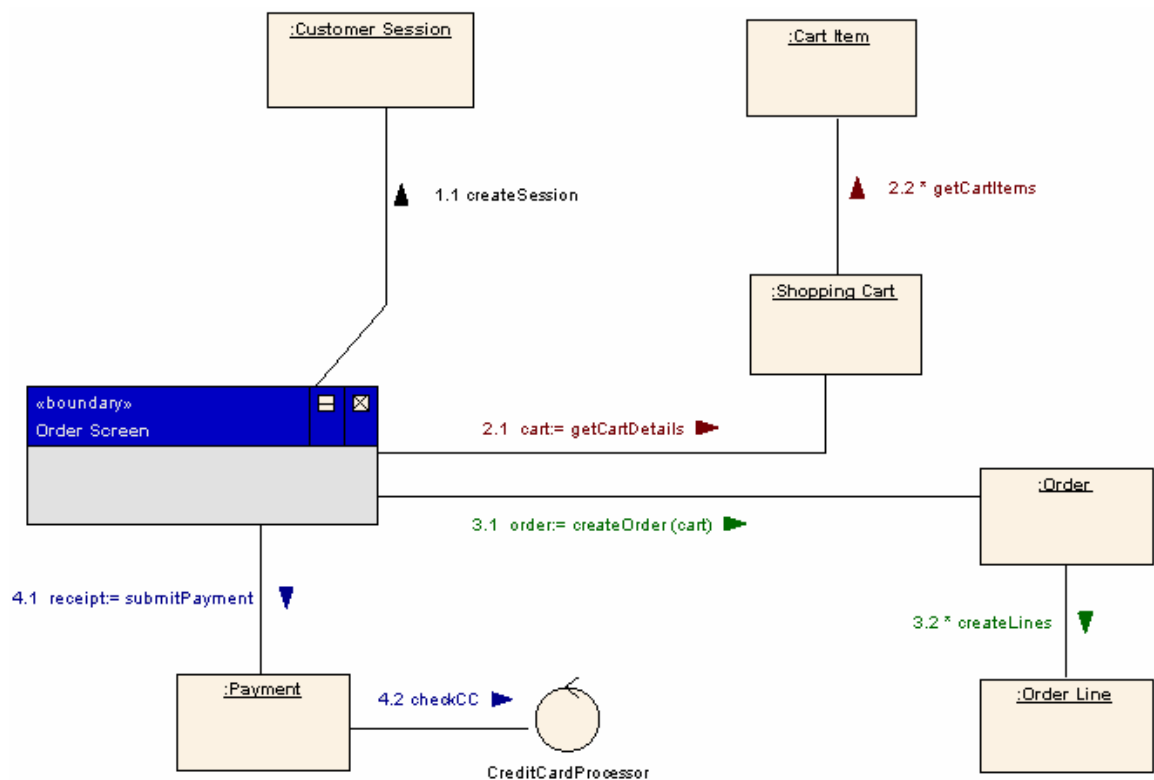
Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.2.1.4.3.1 Communication Diagrams in Color

Enterprise Architect allows you to highlight particular message flows in a Communication diagram using different colors for each message set.

To highlight the colors in a Communication diagram, follow the steps below:

1. Open the **Tools | Options** dialog and go to the **Communication Colors** tab.
2. Make sure the **Use Communication Color** check box is ticked.
3. Select the desired colors.
4. On your Communication diagram, each sequence group of messages will now appear in a different color as shown below.



Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.2.1.4.4 Interaction Overview Diagram

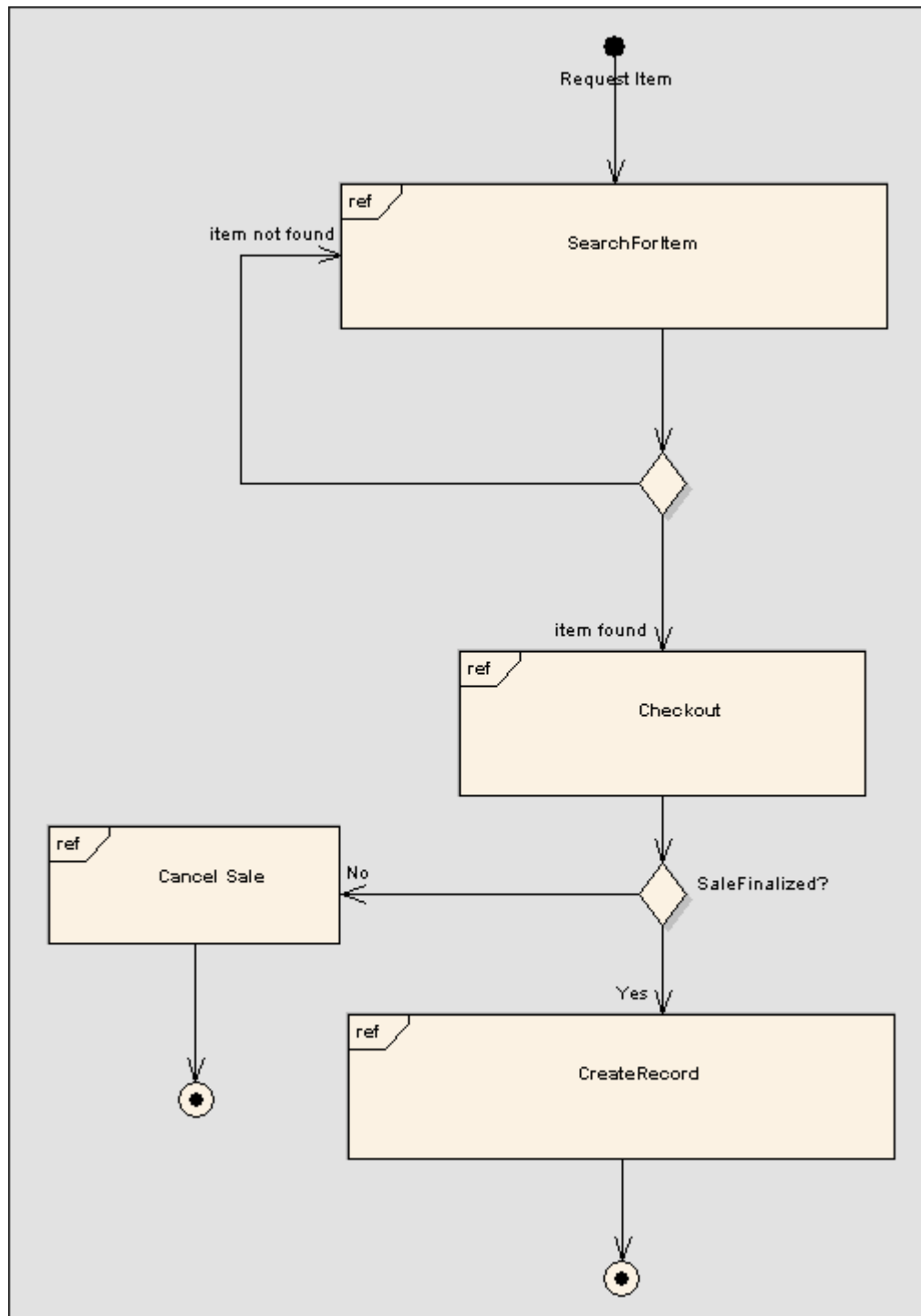
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

Interaction Overview diagrams visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose. As Interaction Overview diagrams are a variant of activity diagrams, most of the diagram notation is similar, as is the process in constructing the diagram. Decision points, forks, joins, start points, and end points are the same. Instead of activity elements, however, rectangular elements are used. There are two types of these elements, interaction elements and interaction occurrence elements. Interaction elements display an inline Interaction diagram, which can be a Sequence diagram, Communication diagram, Timing diagram, or Interaction Overview diagrams. Interaction occurrence elements are references to an existing Interaction diagram. They are visually represented by a frame, with "ref" in the frame's title space. The diagram name is indicated in the frame contents.

To create an interaction occurrence, simply drag an Interaction diagram from the Project Browser onto your Interaction Overview diagram. The "ref" frame will appear, encapsulating an instance of the Interaction diagram.

Example Diagram

The following example depicts a sample sale process, shown in an Interaction Overview diagram, with sub-processes abstracted within interaction occurrences. The diagram appears very similar to an activity diagram, and is conceptualized the same way; as the flow moves into an interaction, that respective interaction's process must be followed before the interaction overview's flow can advance.



















Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

[Interaction Overview Diagram Elements](#)

[Interaction Overview Diagram Connectors](#)

| | |
|--|--|
|  Partition |  Fork/Join |
|  Decision |  Fork/Join |
|  Send |  Control Flow |
|  Receive |  Object Flow |
|  Synch |  Dependency |
|  Initial |  Interrupt Flow |
|  Final | |
|  Flow Final | |
|  Region | |
|  Exception | |

Related Topics

- [Activity Diagram](#)
- [Sequence Diagram](#)
- [Communication Diagram](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 447) states:

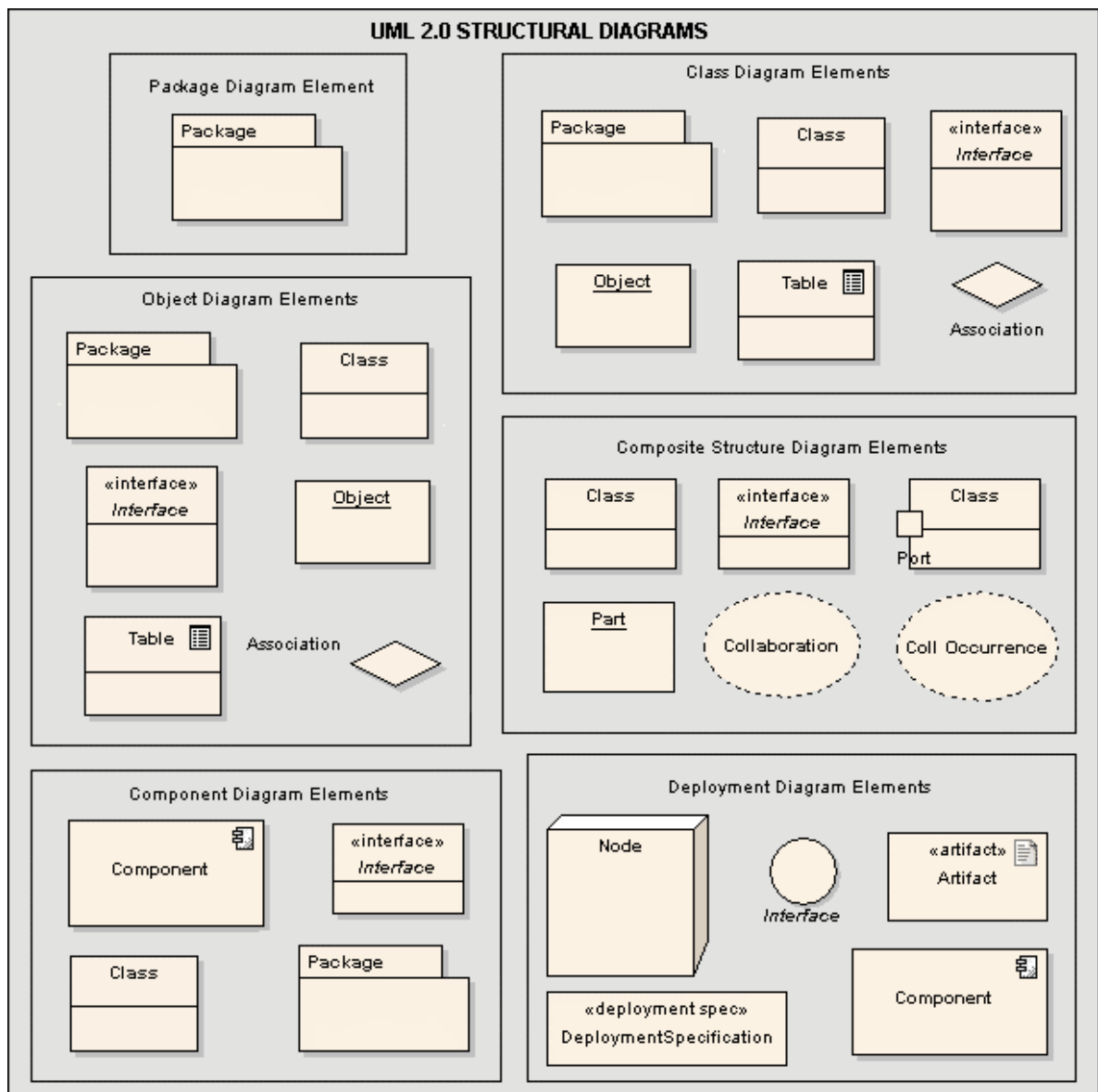
"Interaction Overview Diagrams define Interactions (described in Chapter 14, "Interactions") through a variant of Activity Diagrams (described in Chapter 6, "Activities") in a way that promotes overview of the control flow. Interaction Overview Diagrams focus on the overview of the flow of control where the nodes are Interactions or InteractionOccurrences. The Lifelines and the Messages do not appear at this overview level."

5.2.2 Structural Diagrams

Structural diagrams depict the structural elements composing a system or function. These diagrams can reflect the static relationships of a structure, as do class or package diagrams, or run-time architectures, such as Object or Composite Structure diagrams.

Structural diagrams include:

- [Class diagrams](#)
- [Composite Structure diagrams](#)
- [Component diagrams](#)
- [Deployment diagrams](#)
- [Object diagrams](#)
- [Package diagrams](#)



5.2.2.1 Package Diagram

[Example Diagram](#) | [Elements/Connectors](#) | [OMG UML Specification](#)

Package diagrams are used to reflect the organization of packages and their elements, and provide a visualization of their corresponding namespaces.

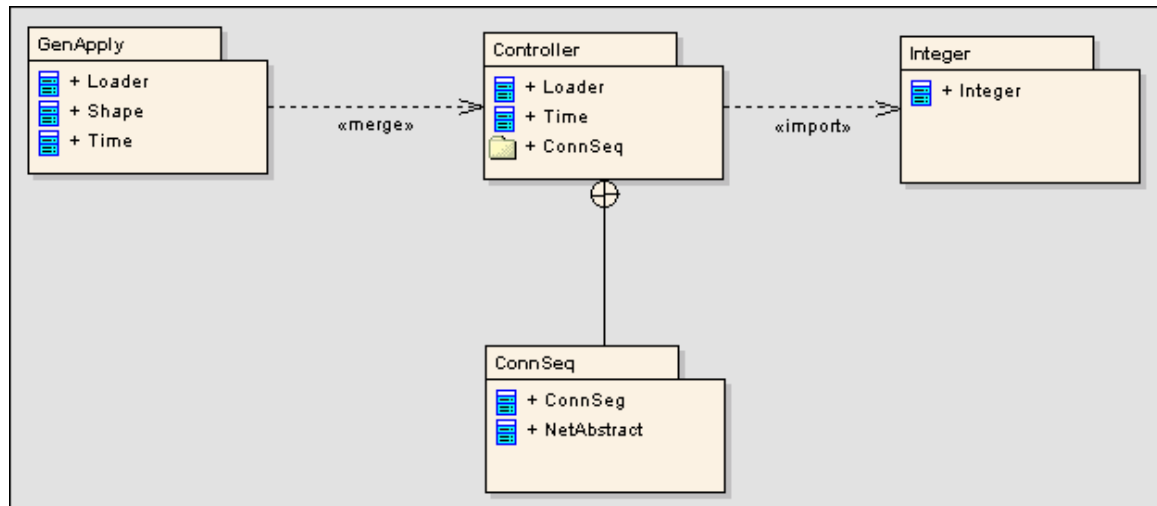
Example Diagram

The following example demonstrates a basic Package diagram. The nesting connector between ConnSeq and Controller reflects what the package contents reveal. Package contents can be listed by accessing the diagram's property window, and selecting "Show Package Contents".

The <<import>> connector indicates that the elements within the target Integer package, which in this example is a single class, Integer, will be imported into the package Controller. The Controller's namespace will gain access to the Integer class; the Integer namespace is not affected.

The <<merge>> connector indicates that the package Controller's elements will be imported into GenApply, including Controller's nested and imported contents. If an element already exists within GenApply, such as Loader and Time, these elements' definitions will be expanded by those included in the package Controller. All elements added or updated by the merge are noted by a generalization relationship back to that package.

Note: Private elements within a package cannot be imported or merged.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Package Diagram Elements | Package Diagram Connectors |
|--------------------------|----------------------------|
| Package | Association |
| Class | Associate |
| Interface | Generalize |
| Object | Compose |
| Table | Aggregate |
| | Association Class |
| | Assembly |
| | Dependency |
| | Realize |
| | Trace |
| | Nesting |
| | Pkg Merge |



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

"A diagram that depicts how model elements are organized into packages and the dependencies among them, including package imports and package extensions."

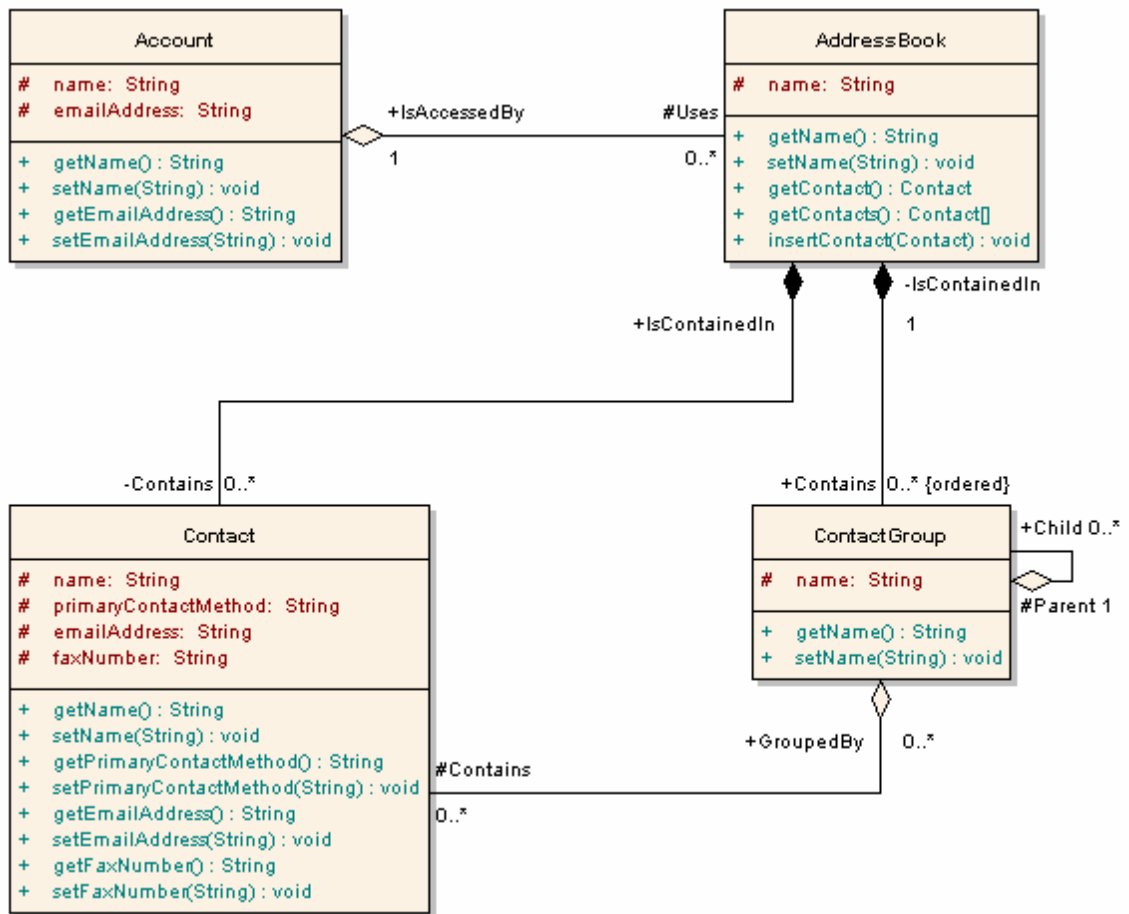
5.2.2.2 Class Diagram

[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

The *Class diagram* captures the logical structure of the system - the classes and things that make up the model. It is a static model, describing what exists and what attributes and behavior it has, rather than how something is done. Class diagrams are most useful to illustrate relationships between classes and interfaces. Generalizations, aggregations, and associations are all valuable in reflecting inheritance, composition or usage, and connections, respectively.

Example Diagram




The lighter aggregation indicates that the class Account uses AddressBook, but does not necessarily contain AddressBook. The strong, composite aggregation by the other connectors indicate ownership or containment by the target classes (at the diamond end) of the source classes.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Class Diagram Elements | Class Diagram Connectors |
|------------------------|--------------------------|
| Package | Association |
| Class | Associate |
| Interface | Generalize |
| Object | Compose |
| Table | Aggregate |
| | Association Class |
| | Assembly |
| | Dependency |
| | Realize |
| | Trace |

| | |
|--|--|
| |  Nesting |
| |  Pkg Merge |
| |  Pkg Import |

Related Topics

- [Parameterized Classes \(Templates\)](#)
- [Active Classes](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 6) states:

"A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships."

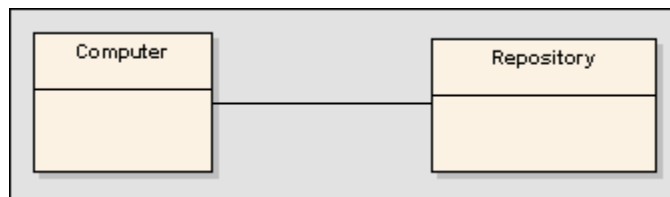
5.2.2.3 Object Diagram

[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

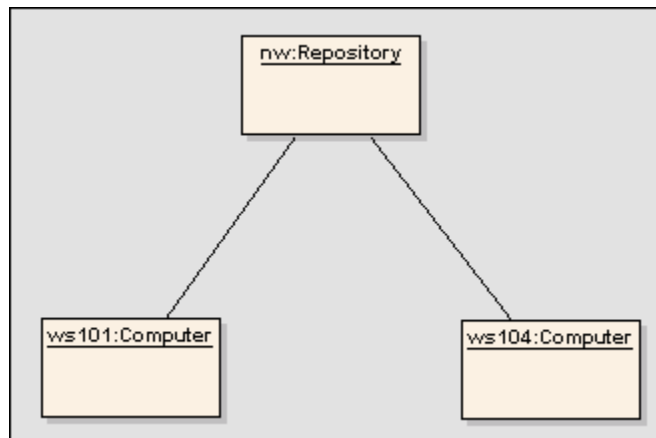
An *Object diagram* is closely related to a Class diagram, with the distinction that it depicts object instances of classes at a point in time. This might appear similar to a Composite Structure diagram, which also models run-time behavior; the difference is that Object diagrams exemplify the static Class diagrams, whereas Composite Structure diagrams reflect run-time architectures different from their static counterparts. Object diagrams do not reveal architectures varying from their corresponding Class diagrams, but reflect multiplicity and the roles instantiated classes could serve. They are useful in understanding a complex Class diagram, by creating different cases in which the relationships and classes are applied. An Object diagram can also be a kind of Communication diagram, which also models the connections between objects, but additionally sequences events along each path.

Example Diagram

The following example first shows a simple Class diagram, with two class elements connected.




















The classes above are instantiated below as objects in an Object diagram. There are two instances of Computer in this model, which can prove useful for considering the relationships and interactions classes play in practice, as objects.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Object Diagram Elements | Object Diagram Connectors |
|---|---|
|  Package |  Association |
|  Interface |  Associate |
|  Object |  Generalize |
|  Table |  Compose |
| |  Aggregate |
| |  Association Class |
| |  Assembly |
| |  Dependency |
| |  Realize |
| |  Trace |
| |  Nesting |
| |  Pkg Merge |
| |  Pkg Import |

Related Topics

- [Composite Structure Diagram](#)
- [Class Diagram](#)
- [Communication Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

"A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a communication diagram."

Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.2.2.4 Composite Structure Diagram

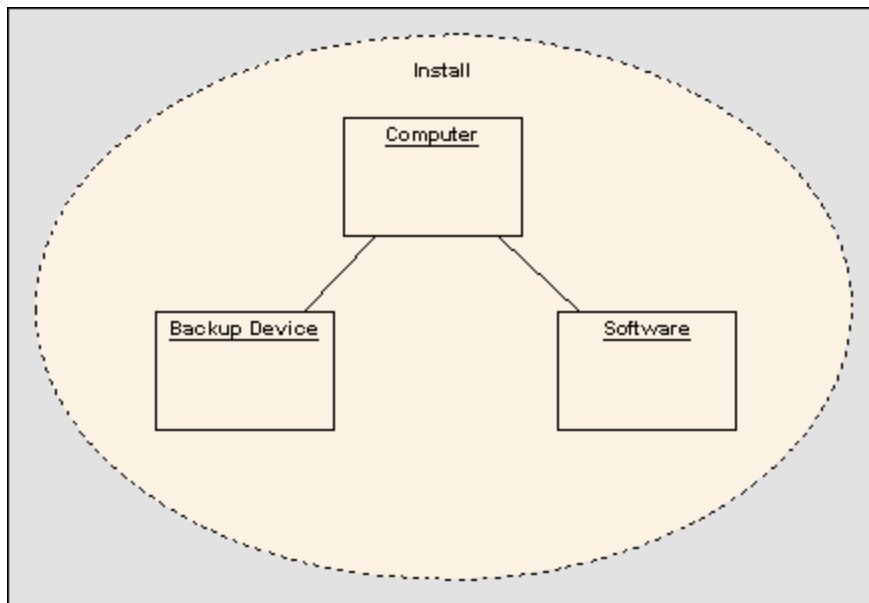
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

A *Composite Structure diagram* reflects the internal collaboration of classes, interfaces or components to describe a functionality. Composite Structure diagrams are similar to Class diagrams, except that they model a specific usage of the structure. Class diagrams model a static view of class structures, including their attributes and behaviors. A Composite Structure diagram is used to express run-time architectures, usage patterns, and the participating elements' relationships, which might not be reflected by static diagrams.

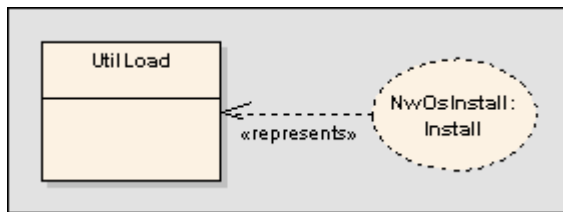
In a Composite Structure diagram, classes are accessed as parts or run-time instances fulfilling a particular role. These parts can have multiplicity, if the role filled by the class requires multiple instances. Ports defined by a part's class should be represented in the composite structure, maintaining that all connecting parts provide the required interfaces specified by the port. There is extensive flexibility, and an ensuing complexity, that come with modeling composite structures. To optimize your modeling, consider building collaborations to represent reusable patterns responding to your design issues.

Example Diagram

The following diagram shows a collaboration used in Composite Structure diagrams to model common patterns. This particular example shows a relationship for performing an installation.














The following diagram uses the Install collaboration in a collaboration occurrence, and applies it to the UtilLoad class via a <<represents>> relationship. This indicates that the classifier UtilLoad uses the collaboration pattern within its implementation. For further examples on composite structure diagrams, refer to the toolbox elements listed below.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| Composite Structure Diagram Elements | Composite Structure Diagram Connectors |
|--|--|
|  Class |  Connector |
|  Interface |  Assembly |
|  Part |  Role Binding |
|  Port |  Represents |
|  Collaboration |  Occurrence |
|  Expose Interface | |

Related Topics

- [Properties](#)
- [Collaboration Occurrence](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 7) states:

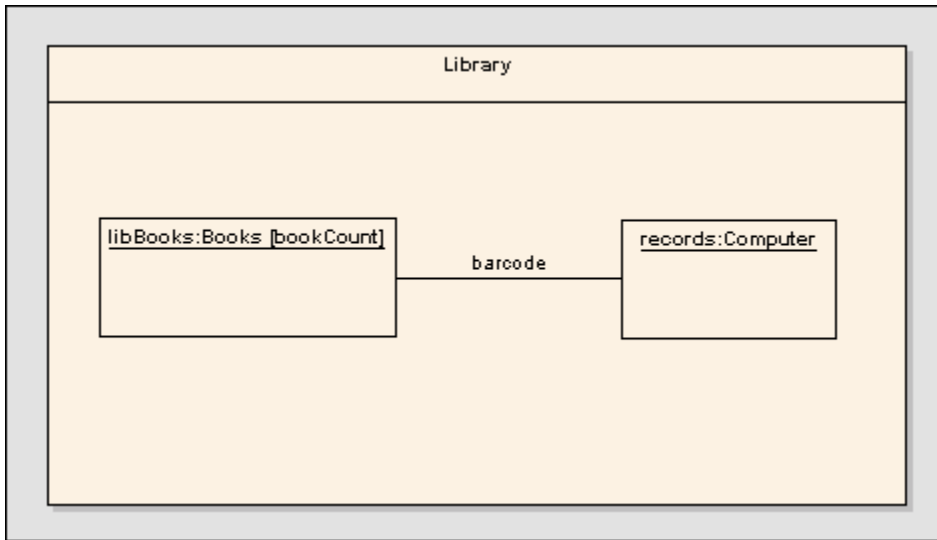
"A diagram that depicts the internal structure of a classifier, including the interaction points of the classifier to other parts of the system. It shows the configuration of parts that jointly perform the behavior of the containing classifier. The architecture diagram specifies a set of instances playing parts (roles), as well as their required relationships given in a particular context."

5.2.2.4.1 Properties

[Related Topics](#)

A property is a nested structure within a classifier, which is usually a class or an interface. The contained structure reflects instances and relationships reflected within the containing classifier. Properties can have multiplicity.

To demonstrate properties, consider the following diagram, which demonstrates some properties belonging to the library class.

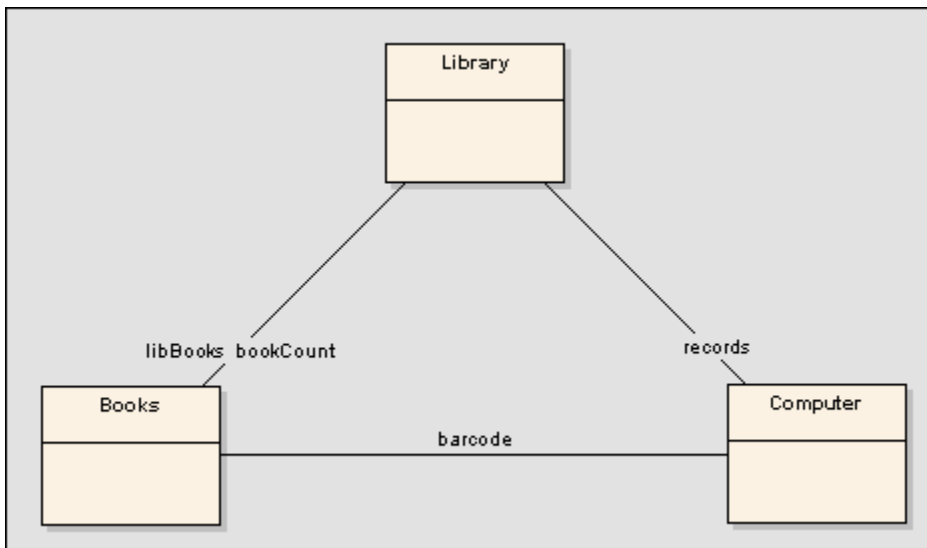


There are two parts, libBooks and records, which are instances corresponding to the classes Books and Computer respectively. After dragging parts from the toolbox out to the workspace, right-click on a part and select *Advanced | Set Property Type* to link to a classifier.

Note: If parts 'disappear' when dragged onto the class, adjust the Z-order of the class (right-click, select *Z-Order*).

The relationship between the two parts is indicated by the connector, reflecting that communication between the parts is done via the barcode. This contained structure and its parts are properties owned by the library class. To indicate a property that is not owned by composition to the containing classifier, use a box symbol with a dashed outline, indicating association. To do this, right-click the part and select *Advanced | Custom Properties*. Set the *IsReference* option to true.

Properties can also be reflected using a normal composite structure (without containing it in a class), with the appropriate connectors, parts and relationships indicated through connections to the class. This alternate representation is shown below. However, this depiction fails to express the ownership immediately reflected by containing properties within a classifier.



Related Topics

- [Composite Structure Diagram](#)

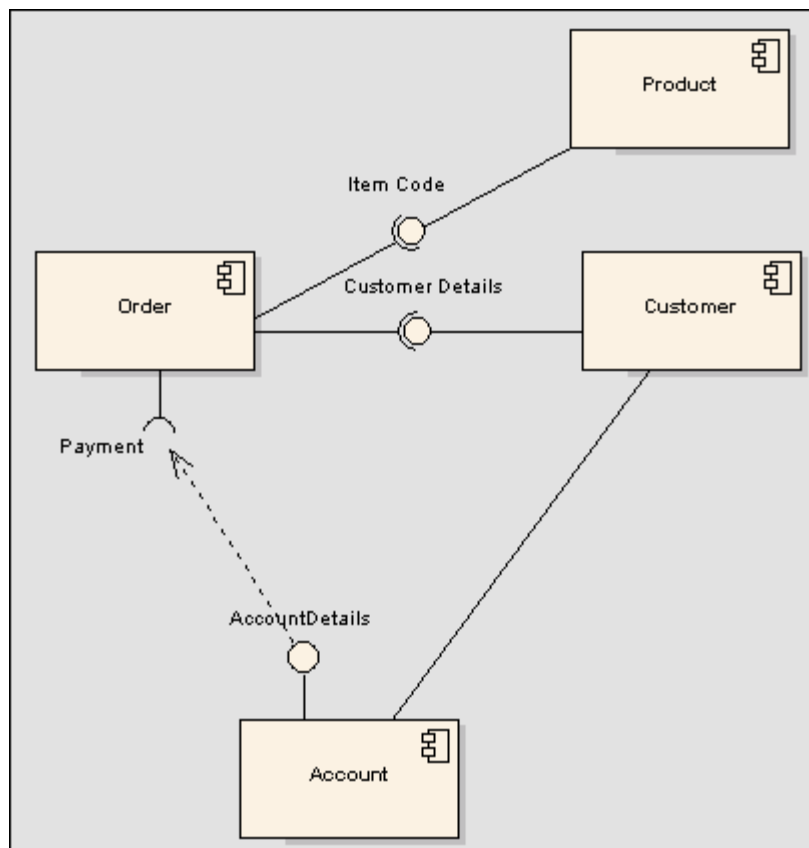
5.2.2.5 Component Diagram

[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)
















A *Component diagram* illustrates the pieces of software, embedded controllers, etc. that will make up a system. A Component diagram has a higher level of abstraction than a Class diagram - usually a component is implemented by one or more classes (or objects) at runtime. They are building blocks, such that eventually a component can encompass a large portion of a system.

Example Diagram

The diagram below demonstrates some components and their inter-relationships. Assembly connectors 'link' the provided interfaces supplied by Product and Customer to the required interfaces specified by Order. A dependency relationship maps a customer's associated account details to the required interface, 'Payment', indicated by Order.

**Toolbox Elements and Connectors**

Tip: Click the elements and connectors below for more information.

| <i>Component Diagram Elements</i> | <i>Component Diagram Connectors</i> |
|--|--|
|  Package |  Assembly |
|  Component |  Delegate |
|  Class |  Associate |
|  Interface |  Realize |
|  Object |  Dependency |
|  Port |  Trace |
|  Expose Interface |  Generalize |
|  Artifact | |

Related Topics

- [Class Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure, p. 7*) states:

"A diagram that shows the organizations and dependencies among components."

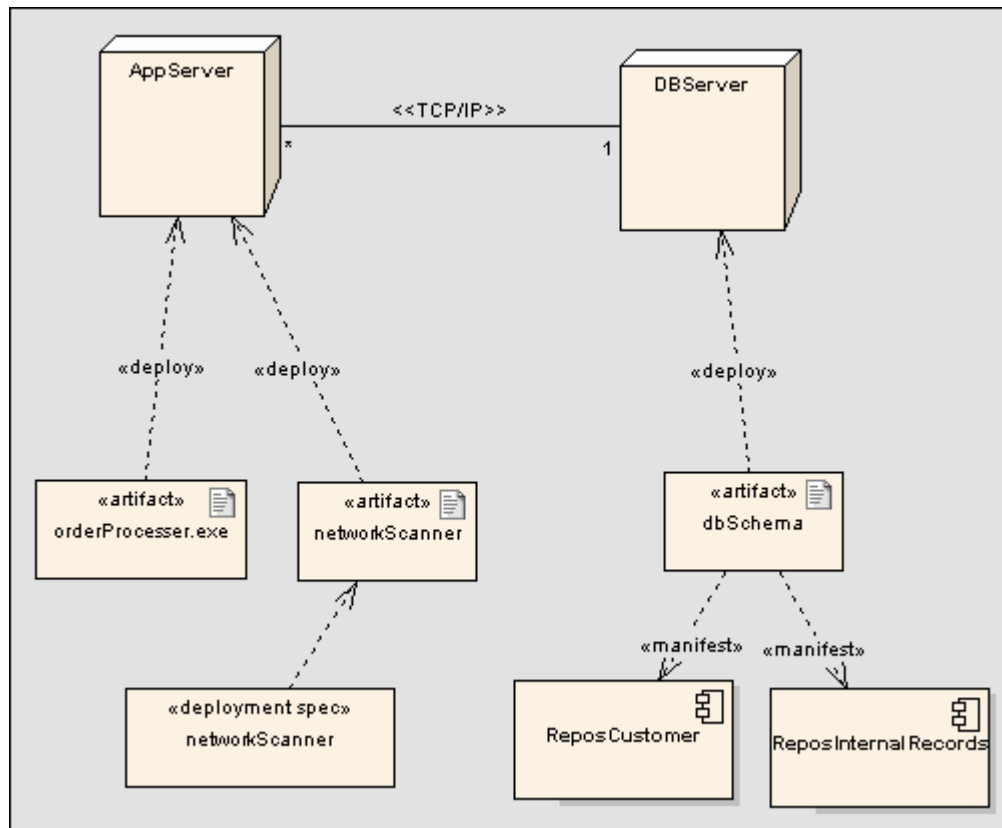
5.2.2.6 Deployment Diagram

[Example Diagram](#) | [Elements/Connectors](#) | [OMG UML Specification](#)

A *Deployment diagram* shows how and where the system will be deployed. Physical machines and processors are reflected as nodes, and the internal construction can be depicted by embedding nodes or artifacts. As artifacts are allocated to nodes to model the system's deployment, the allocation is guided by the use of deployment specifications.

Example Diagram

Below is an example Deployment diagram. The two nodes have a TCP/IP communication path indicated. Deployment relationships indicate the deployment of artifacts. Furthermore, a deployment spec defines the process of deployment for the networkScanner artifact. The manifestation relationships reveals the physical implementation of components ReposCustomer and ReposInternalRecords.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

| <i>Deployment Diagram Elements</i> | <i>Deployment Diagram Connectors</i> |
|---|---|
| Node | Associate |
| Component | Association Class |
| Interface | Generalize |
| Artifact | Realize |
| Deployment Spec | Deployment |
| Package | Manifest |
| | Dependency |
| | Trace |
| | Object Flow |
| | Nesting |

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 8) states:

"A diagram that depicts the execution architecture of systems. It represents system artifacts as nodes, which are connected through communication paths to create network systems of arbitrary complexity. Nodes are typically defined in a nested manner, and represent either hardware devices or software execution environments."

5.2.3 Additional Diagrams

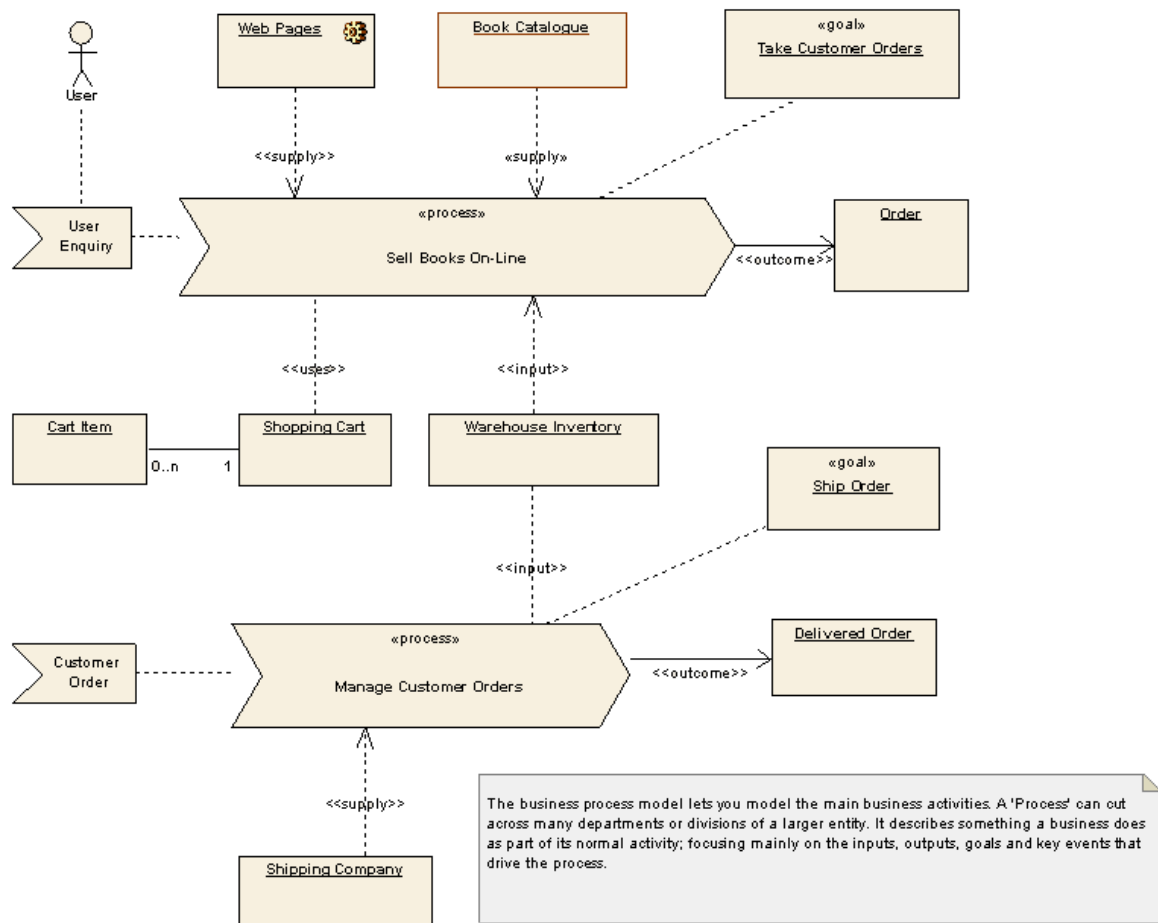
In addition to diagrams defined by the UML, EA provides some additional diagram platforms to model business processes or custom diagrams.

- [Analysis Diagram](#)
- [Custom Diagram](#)
- [Requirements Diagram](#)
- [Maintenance Diagram](#)
- [User Interface Diagram](#)
- [Database Schema](#)
- [Robustness Diagram](#)

5.2.3.1 Analysis Diagram

An *Analysis diagram* is a simplified Activity diagram, which is used to capture high level business processes and early models of system behavior and elements. It is less formal than some other diagrams, but provides a good means of capturing the essential business characteristics and needs.

EA supports some of the Eriksson-Penker Business Extensions which facilitate business process modeling. The complete Eriksson-Penker Business Extensions UML Profile may also be loaded into EA and used to create detailed process models.

**See Also**

- [Business Modeling](#)

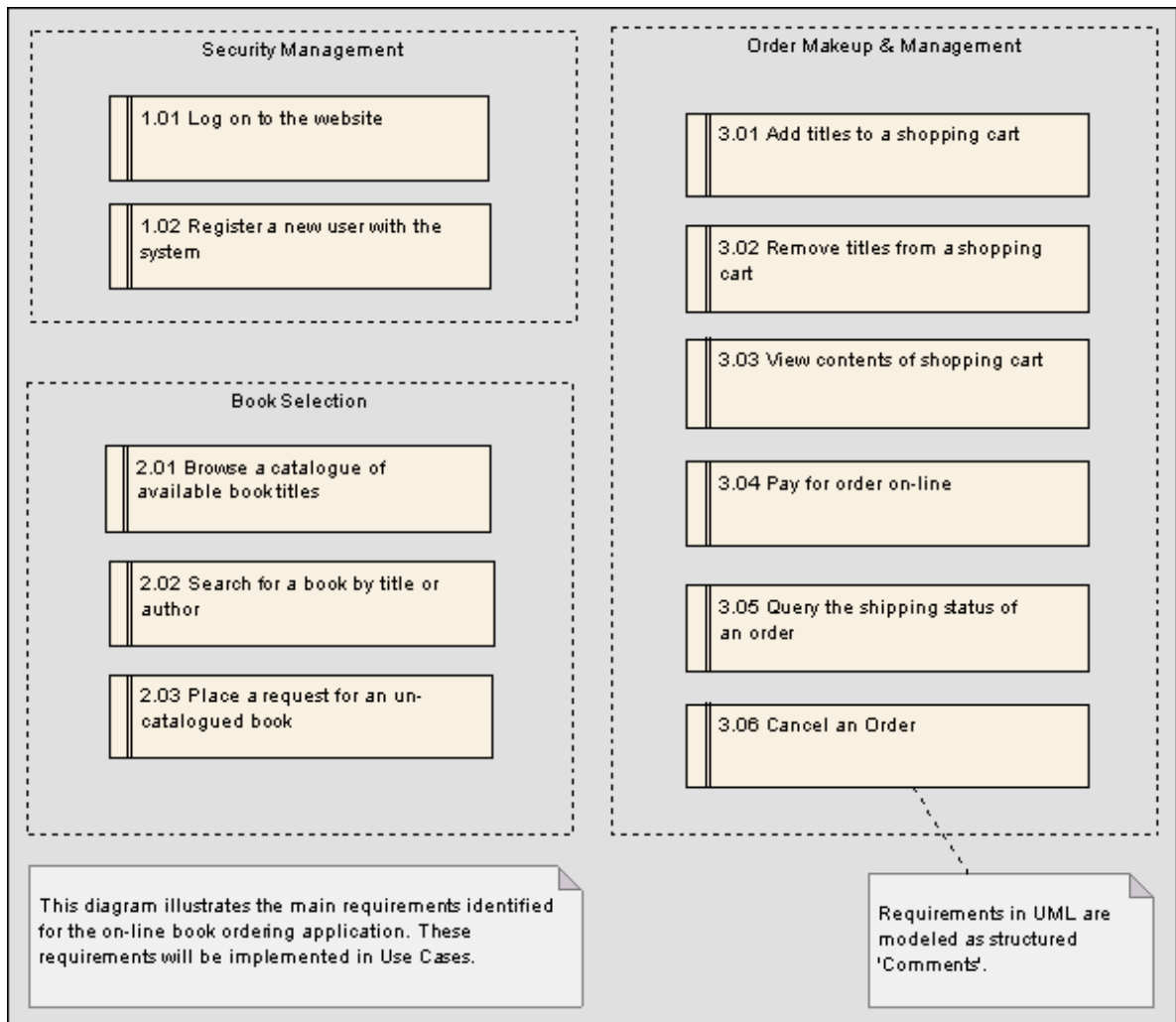
5.2.3.2 Custom Diagram

A *Custom diagram* is an extended class diagram which is used to capture requirements, user interfaces or custom-design models.

The below example reflects a requirements diagram. Requirement elements may then be linked back to use cases and components in the system to illustrate how a particular system requirement is met.

Screen design is supported through a stereotyped screen element and UI Controls. Use this model to design high level system prototypes.

Custom models provide a few extensions to the UML model and allow for some exploratory and non-rigorous experimentation with model elements and diagrams.

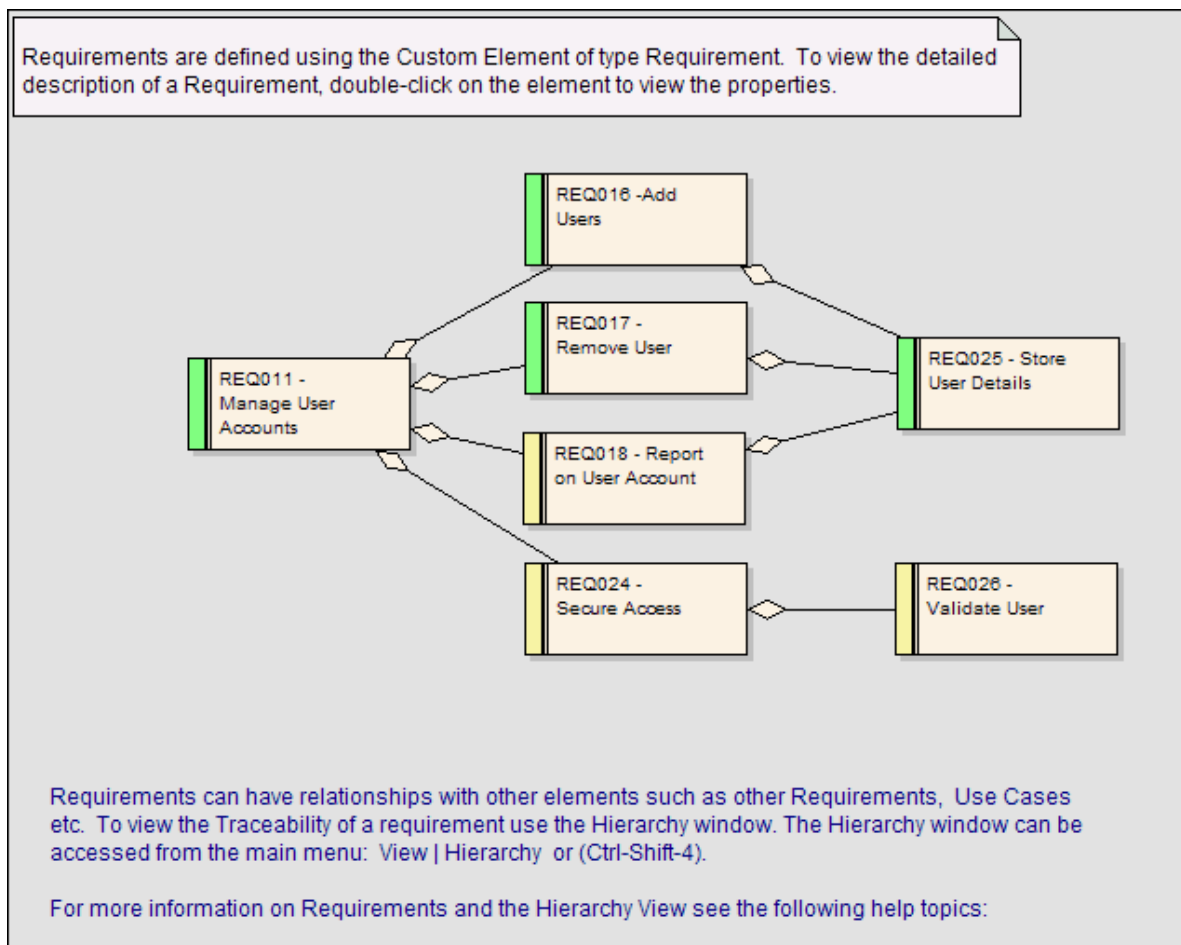


5.2.3.3 Requirements Diagram

A *Requirements diagram* is a custom diagram used to describe a system's requirements or features as a visual model.

The below example reflects a requirements diagram. Requirement elements may then be linked back to use cases and components in the system to illustrate how a particular system requirement is met.

Requirements models provide extensions to the UML model and allow for traceability between specifications and design requirements, and the model elements which realize them.



5.2.3.4 Maintenance Diagram

A *Maintenance diagram* is a custom diagram used to describe change requests and issue items within a system model.

The below example illustrates an example maintenance diagram. Change, task and issue elements may then be linked back to other model elements in the system to illustrate how they need to be modified, fixed, or updated.

Maintenance models provide extensions to the UML model and allow for change management between change items, and the model elements which require the changes to be made on them.

Changes

EA supports custom Elements of type 'Change'. These can be linked to other elements in the repository or used as a separate lists of any changes proposed for the model.

Below are a set of Change elements for the shopping basket with test definitions listed against them. They are ready for checking once they are confirmed as corrected.

The color markings reflect the Status of the element.

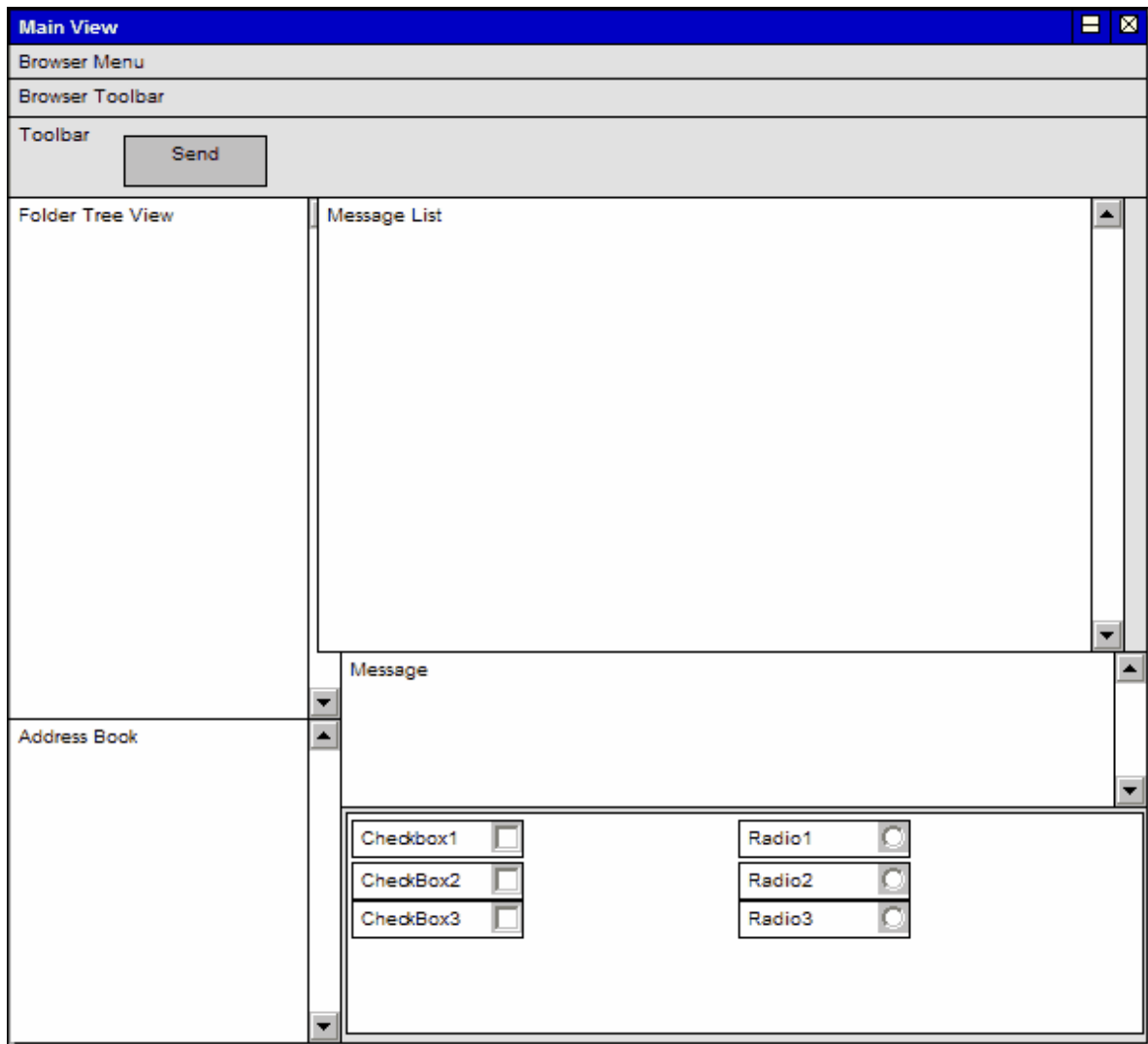
| | |
|--|--|
| View Basket - add: alter quantity against the entries. | test scripts |
| | Unit : (Not Run) Alter Quantity |
| View Basket - Add button to update Change | test scripts |
| | Unit : (Deferred) Update Cart Button operative |
| Create Account: password confirmation fails | test scripts |
| | Unit : (Not Run) Password Confirmation |

5.2.3.5 User Interface Diagram

User Interface Diagrams are used to visually mock-up a system's user interface using forms, controls and labels.

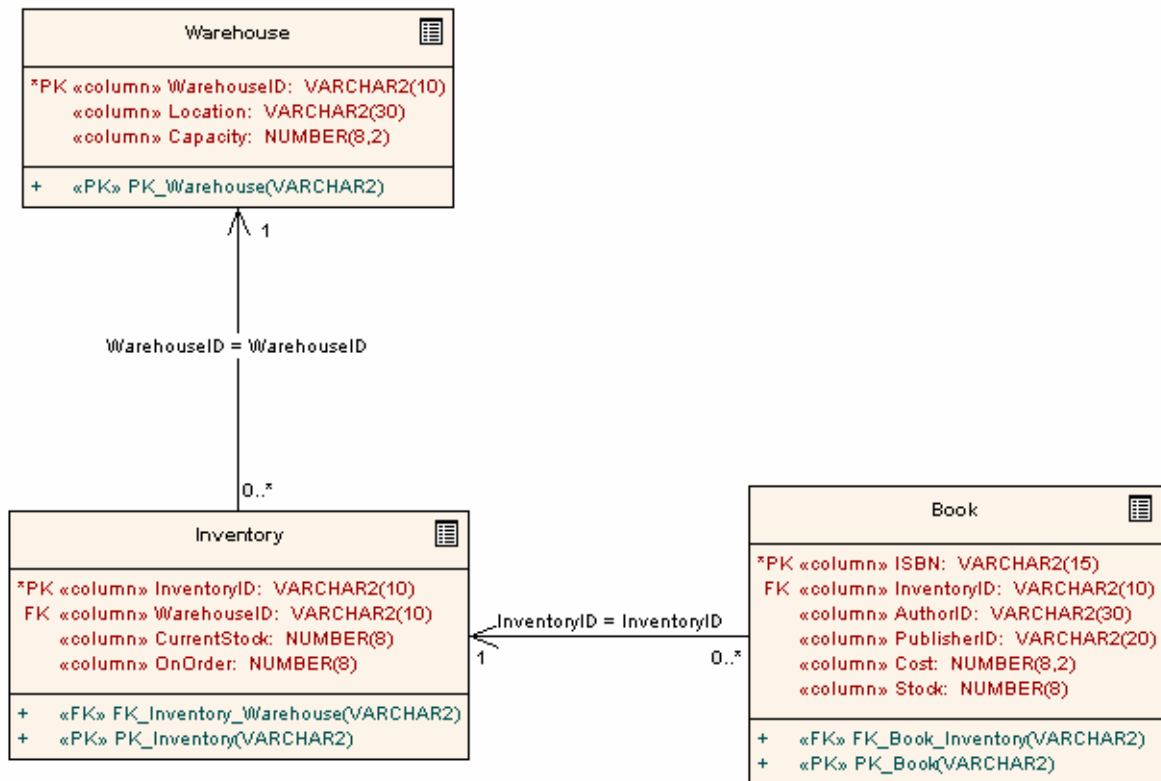
A *User Interface diagram* is a custom diagram used to describe the visual mock-up of a system's user interface.

The below example illustrates an example User Interface diagram. Forms, controls and labels are arranged on the diagram to describe its appearance. UI elements can also be traced to other model elements linking the UI with the underlying implementation.



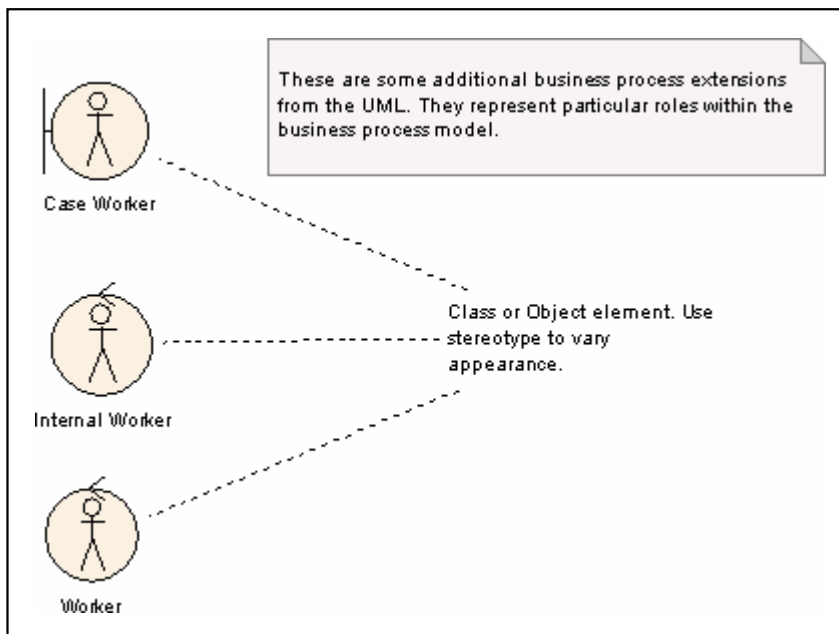
5.2.3.6 Database Schema

Below is an example *Database Schema*:



5.2.3.7 Robustness Diagram

Enterprise Architect supports business process modeling extensions from the UML business process model profile. Examples of these are below:



Robustness diagrams are used in the Iconix Process - you can read more about this at www.sparxsystems.com.au/iconix/iconixsw.htm.

5.3 UML Elements

Models in the UML are constructed from elements. Each element has a different purpose, different rules and different notation. Model elements are used at different stages of the design process for different purposes. Elements include [Classes](#), [Objects](#), [Interfaces](#), [Use Cases](#), [Components](#) and [Nodes](#).

The features of Enterprise Architect cannot be fully utilized without a good understanding of UML:

- During early analysis, use cases, activities, business processes, objects and collaborations are used to capture the problem domain.
- During elaboration, sequence diagrams, objects, classes and state machines are used to refine the system specification.
- Components and nodes are used to model larger parts of the system as well as the physical entities that will be created and deployed into a production environment.

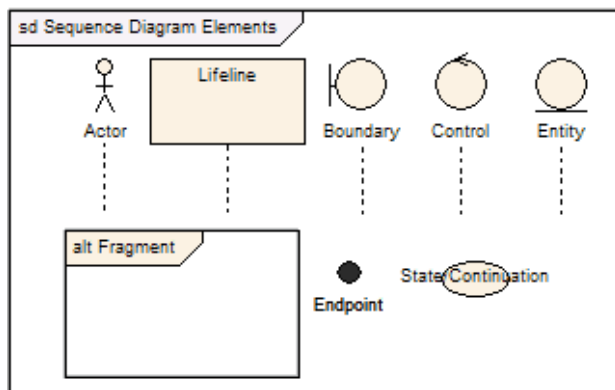
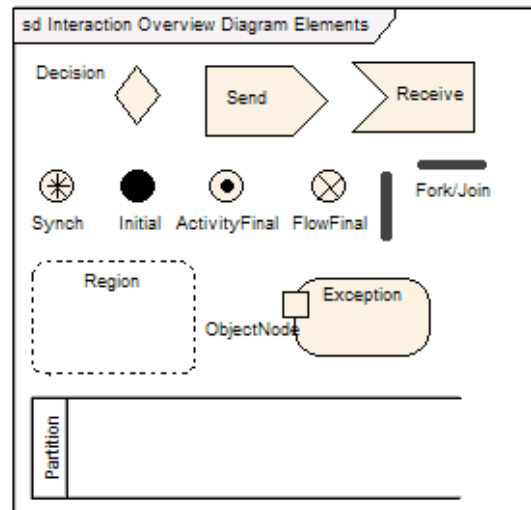
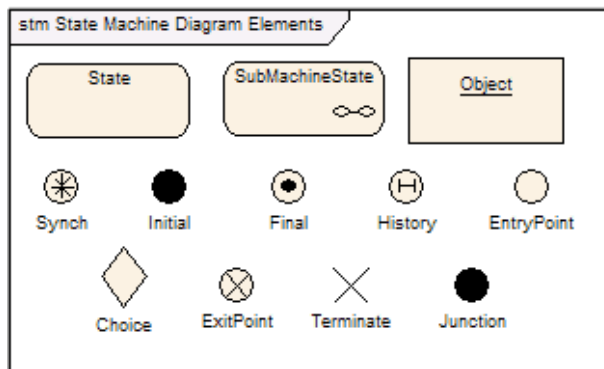
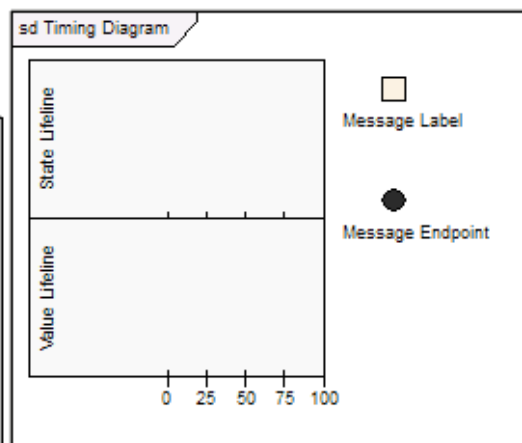
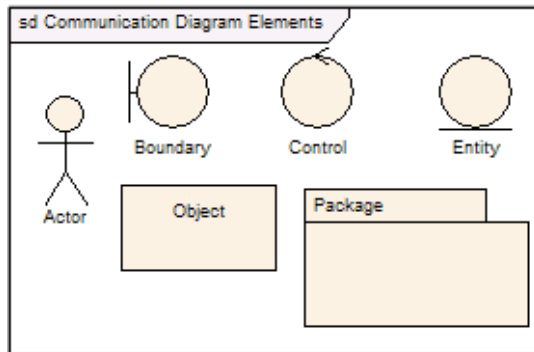
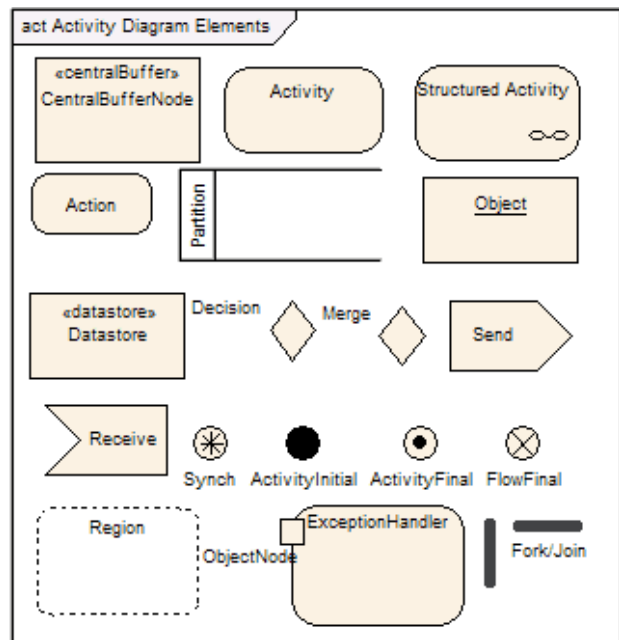
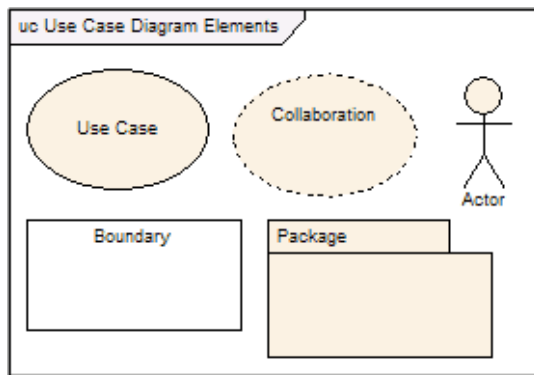
UML Elements can be divided into two categories: those used on [Behavioral Diagrams](#) and those used on [Structural Diagrams](#). This basic set can be extended almost without limit using [Stereotypes](#) and [UML Profiles](#).

5.3.1 Behavioral Diagram Elements

The following figure illustrates the main UML Elements that are used in Behavioral Diagrams. For more insight into using these elements, click on a link:

- A: [Action](#), [Activity](#), [Actor](#)
- B: [Boundary](#)
- C: [Choice](#), [Collaboration](#), [Combined Fragment](#), [Continuation](#)
- D: [Datastore](#), [Decision](#)
- E: [Endpoint](#), [Entry Point](#), [Exception](#), [Expansion Region](#), [Exit Point](#)

- F: [Final](#), [Flow Final](#), [Fork](#)
- H: [History](#)
- I: [Initial](#), [Interaction Occurrence](#), [Interruptible Activity Region](#)
- J: [Join](#), [Junction](#)
- L: [Lifeline](#)
- O: [Object](#)
- P: [Package](#), [Partition](#)
- R: [Receive](#), [Region](#)
- S: [Send](#), [State](#), [State Lifeline](#), [State/Continuation](#), [SubActivity](#), [SubMachine](#), [Synch](#), [System Boundary](#)
- T: [Terminate](#)
- U: [Use Case](#)
- V: [Value Lifeline](#)



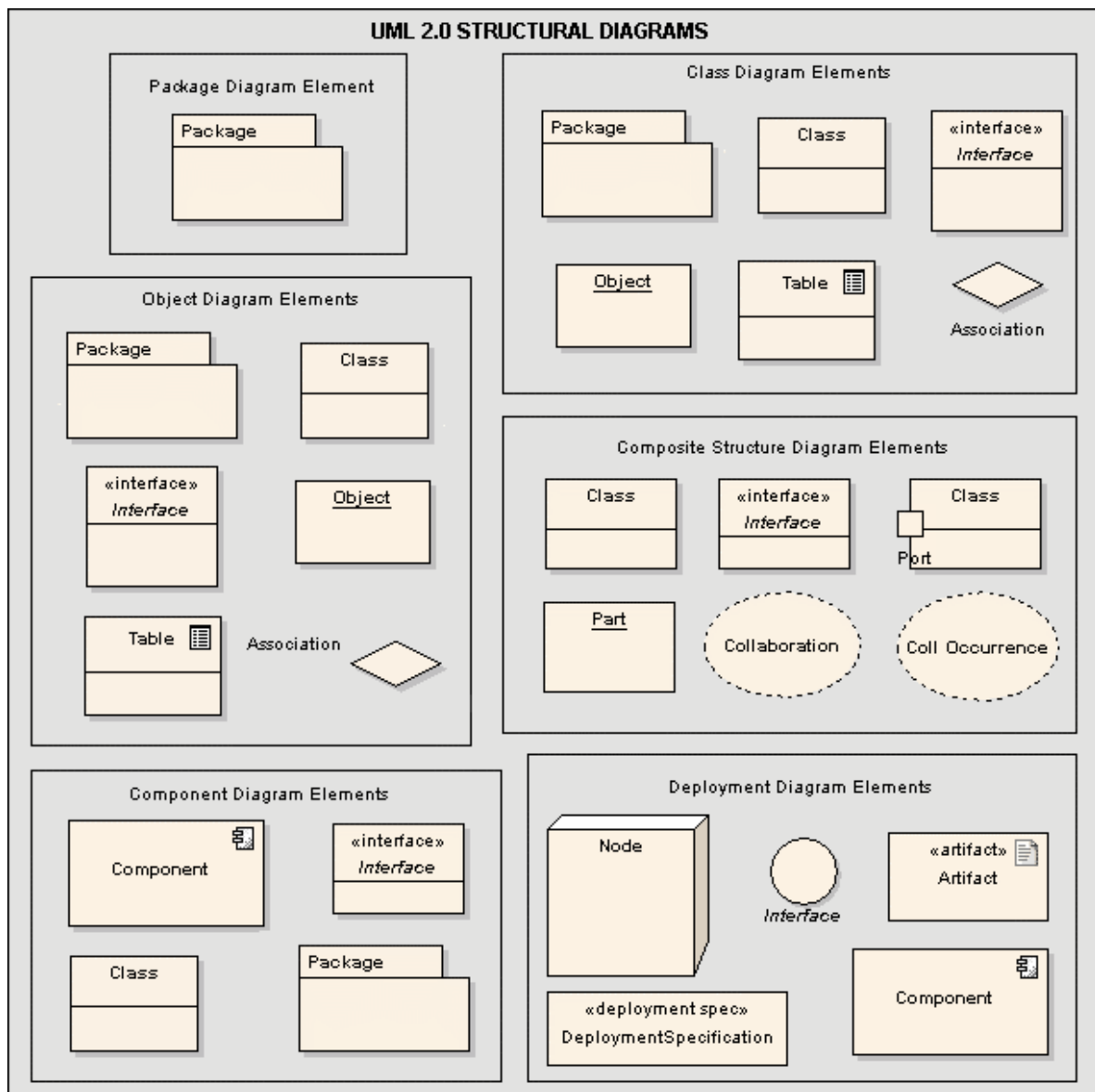
See Also

- [Introduction to the UML Language](#)
- [Introduction to UML Elements](#)
- [Structural Diagram Elements](#)

5.3.2 Structural Diagram Elements

The following figure illustrates the main UML Elements that are used in Structural Diagrams. For more insight into using these elements, click on a link:

- A: [Artifact](#)
- C: [Class](#), [Collaboration](#), [Collaboration Occurrence](#), [Component](#)
- D: [Deployment Specification](#)
- I: [Interface](#)
- N: [Node](#)
- O: [Object](#)
- P: [Package](#), [Part](#), [Port](#)
- Q: [Qualifiers](#)

**See Also**

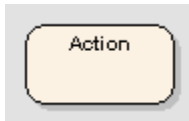
- [Introduction to the UML Language](#)
- [Introduction to UML Elements](#)
- [Behavioral Diagram Elements](#)

5.3.3 Basic Elements

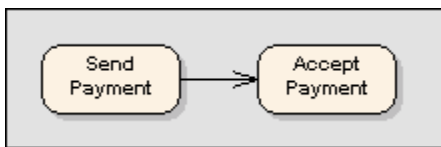
An introduction to elements defined by the UML follows, which together compose the backbone of modeling. The most conceivable modeling elements are stereotypes or extensions of the elements introduced in this section.

5.3.3.1 Action

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *action* element describes a basic process or transformation that occurs within a system. It is the basic functional unit within an Activity diagram. Actions can be thought of as children of [activities](#). Both represent processes, but activities can contain multiple steps or decomposable processes, each of which can be embodied in an action. An action cannot be further broken down or decomposed.



Common Usage

- [Activity Diagram](#)

☐ Action

Further Information

- [Action Notation](#)
- [Action Expansion Node](#)
- [Activity Pre and Post Conditions](#)
- [Action Pin](#)
- [Activities](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 280) states:

"An action is an executable activity node that is the fundamental unit of executable functionality in an activity, as opposed to control and data flow among actions. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise."

"An action may have sets of incoming and outgoing activity edges that specify control flow and data flow from and to other nodes. An action will not begin execution until all of its input conditions are satisfied. The completion of the execution of an action may enable the execution of a set of successor nodes and actions that take their inputs from the outputs of the action."

5.3.3.1.1 Action Notation

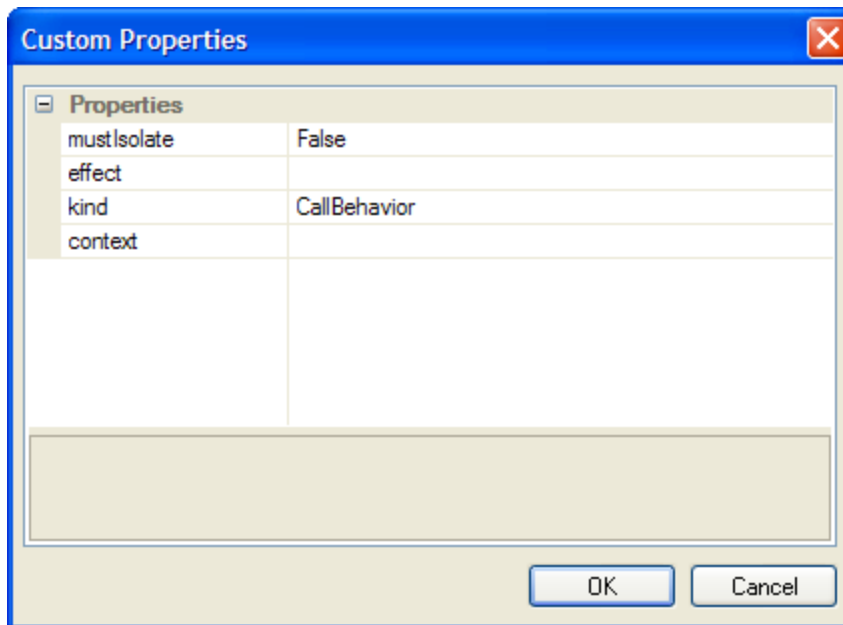
[Further Information](#)

Some properties can be graphically depicted on an action element, as exemplified below.



| | | | | |
|--|---|--|---|---|
| Action Notation Kind: SendSignal | Action Notation Kind: AcceptEvent | Action Notation Kind: AcceptEventTimer | Action Notation Kind: CallOperation | Action Notation Kind: CallBehaviour |
|--|---|--|---|---|

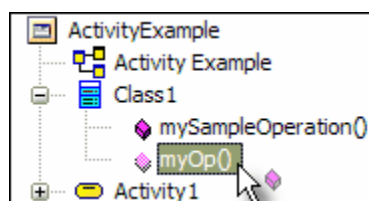
These properties can be defined by right-clicking on the activity and selecting *Advanced Properties*, which opens the following *Custom Properties* dialog box.



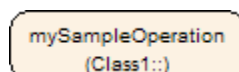
Class Operations in Activity Diagrams

Operations from classes may be displayed on Activity diagrams as an action. When an operation is shown as an action the notation of the action, will display the name of the class which features the operation. To add an operation to an Activity diagram use the following procedure:

1. Open an Activity diagram.
2. From the *Project View* open a class and locate the operation that is to be added to the activity diagram.
3. Drag the operation on to the diagram.

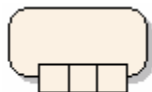


4. When the operation has been added to the Activity diagram the Action will display the namer of the class which features the operation.



See Also

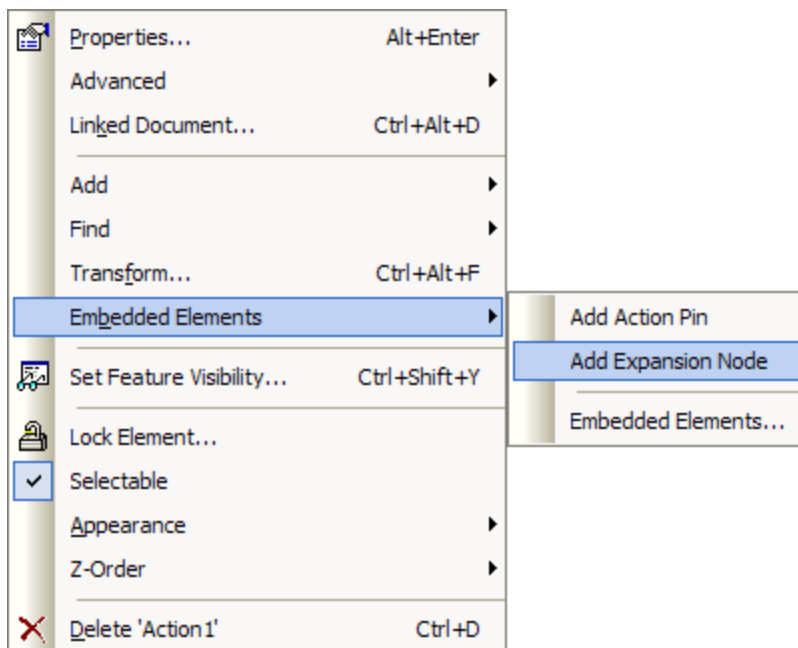
- [Action](#)

5.3.3.1.2 Action Expansion Node[Further Information](#)

ExpansionNode

Representing an action as an *expansion node* is a shorthand notation to indicate that action composes an [expansion region](#).

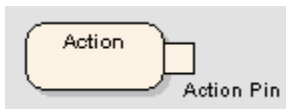
To specify an action is an expansion node, *right-click* on the action, which will render the dialog below. Navigate to the *Expansion Node* option. After designating an action as an expansion node, modifications or deletions can be made by accessing the *Insert Embedded Elements...* option.

**See Also**

- [Expansion Region](#)
- [Action](#)

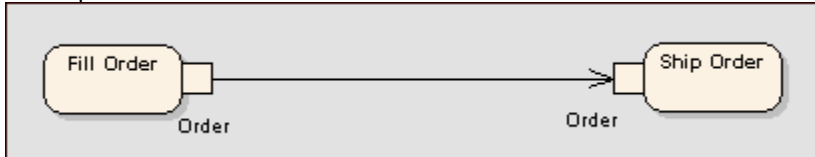
5.3.3.1.3 Action Pin

[Further Information](#) | [OMG UML Specification](#)



An *action pin* is used to define the data flow out of and into an action. An input pin provides values to the action, whereas an output pin contains the results from that action.

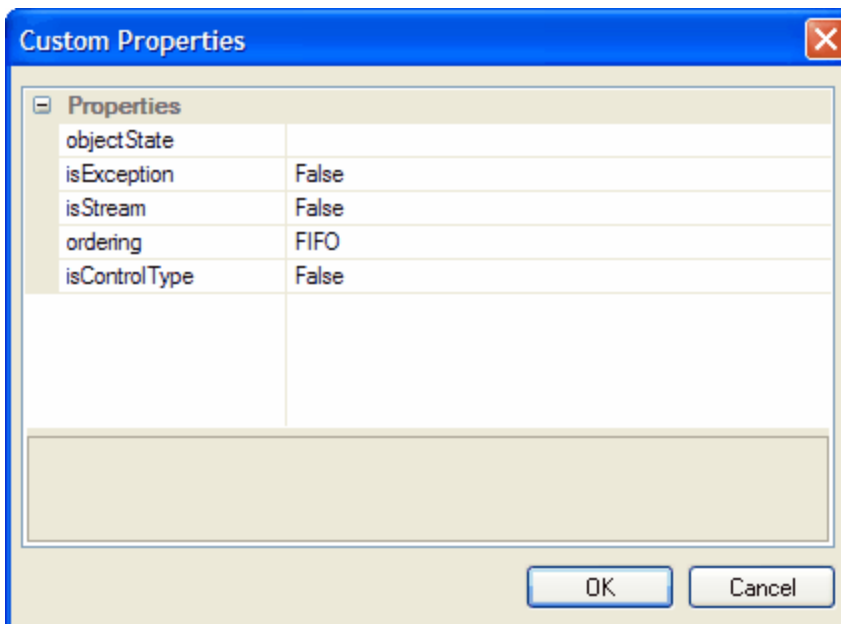
Action pins are used below to connect two actions:



Refer to figure 271 (*UML 2.0 Superstructure*, p. 327).

Action pins can be further characterized as defining exception parameters, streams, or states. Associating a state with a pin defines the state of input or output values. For instance, the pin could be called "Orders", but the state could be "Validated" or "Canceled".

To add an action pin to an action, *right click* on the action and from the context menu select *Embedded Element* | *Add Action Pin*. To change the type of an action pin *right-click* on the pin and select *Advanced* | *Custom Properties*. The following properties can be set:



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 13) states:

"A model element that represents the data values passed into a behavior upon its invocation as well as the data values returned from a behavior upon completion of its execution."

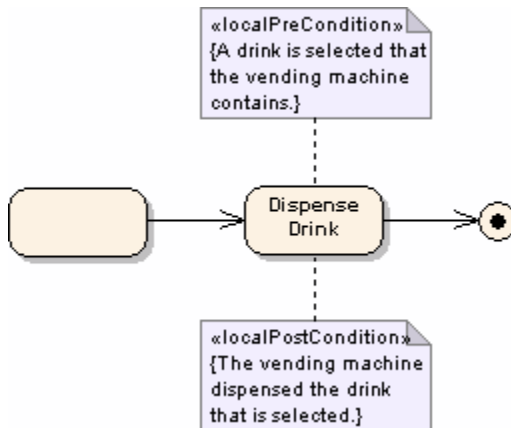
See Also

- [Action](#)

5.3.3.1.4 Local Pre/Post Conditions

[Further Information](#)

Actions can be further defined with *pre-condition* and *post-condition notes*, which constrain an action's entry and exit. These notes can be added to an action as defined below.

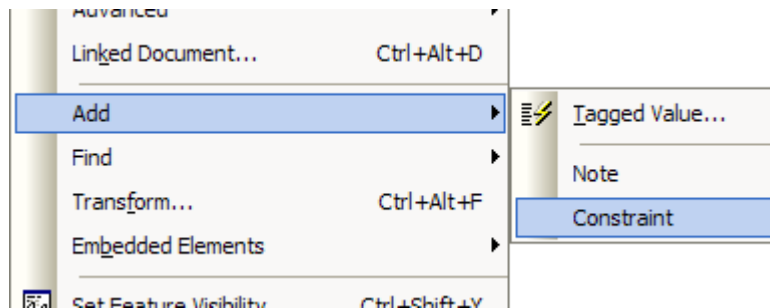


Refer to figure 200 (*UML 2.0 Superstructure*, p. 283).

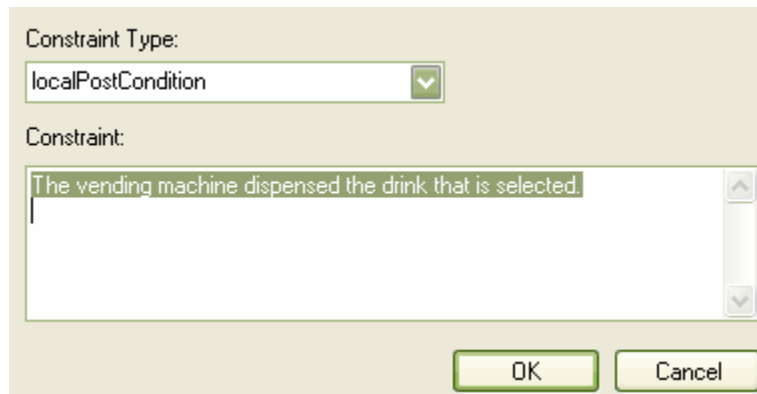
Creating a Constraint

To attach a constraint to an action use the following steps:

1. Right-click on the action. This will give the following:



2. Select *Add*. From the sub-menu select *Constraint*.
3. Right-click on the newly created Note.
4. Select *Properties*.



Constraint Type:
localPostCondition

Constraint:
The vending machine dispensed the drink that is selected.

OK Cancel

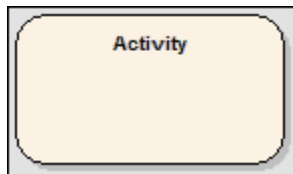
5. Select the *Constraint Type*.
6. Add the text for the constraint.
7. Select *OK* to save it.

See Also

- [Action](#)

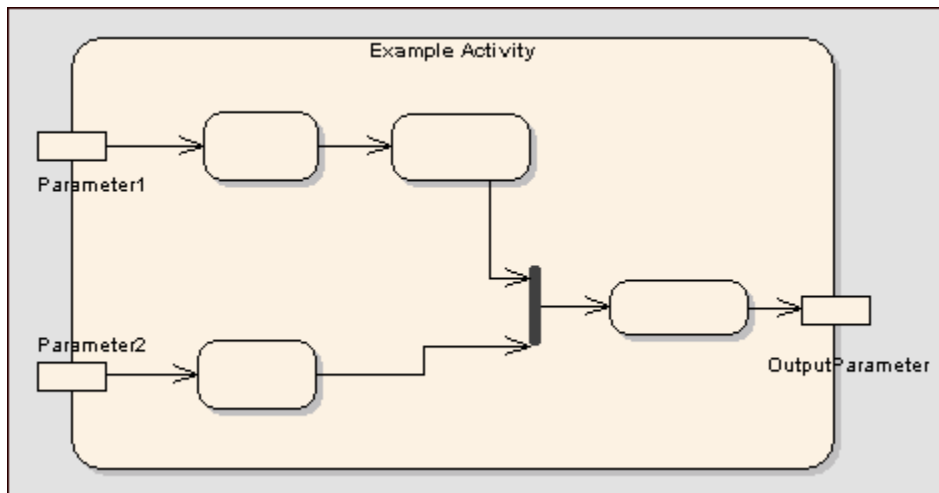
5.3.3.2 Activity


[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *activity* organizes and specifies the participation of subordinate behaviors, such as sub-activities or actions, to reflect the control and data flow of a process. Activities are used for various modeling purposes, from procedural-type application development for system design, to business process modeling of organizational structures or workflow.

Below is a simple diagram of an activity containing action elements and including input parameters and output parameters.



 Activity

Common Usage

- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (UML 2.0 Superstructure, p. 284) states:

"An activity specifies the coordination of executions of subordinate behaviors, using a control and data flow model. The subordinate behaviors coordinated by these models may be initiated because other behaviors in the model finish executing, because objects and data become available, or because events occur external to the flow. The flow of execution is modeled as activity nodes connected by activity edges. A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents. Activity nodes also include flow-of-control constructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to operations that are directly invoked.

"Activities may describe procedural computation. In this context, they are the methods corresponding to operations on classes. Activities may be applied to organizational modeling for business process engineering and workflow. In this context, events often originate from inside the system, such as the finishing of a task, but also from outside the system, such as a customer call. Activities can also be used for information system modeling to specify system level processes.

"Activities may contain actions of various kinds:

- occurrences of primitive functions, such as arithmetic functions.
- invocations of behavior, such as activities.
- communication actions, such as sending of signals.
- manipulations of objects, such as reading or writing attributes or associations.

"Actions have no further decomposition in the activity containing them. However, the execution of a single action may induce the execution of many other actions. For example, a call action invokes an operation which is implemented by an activity containing actions that execute before the call action completes."

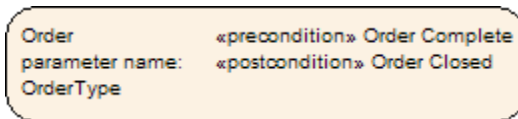
See Also

- [Activity Pre and Post Conditions](#)
- [Action Pin](#)
- [Activities](#)

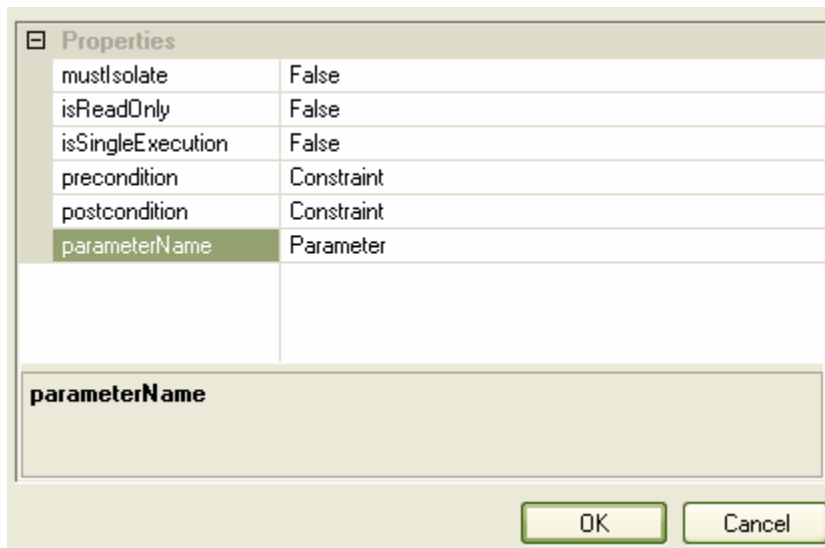
5.3.3.2.1 Activity Notation

[Further Information](#)

Some properties can be graphically depicted on an activity element, as shown below.



These properties can be defined by right-clicking on the activity and selecting *Advanced | Custom Properties*, which opens the following dialog.



Further Information

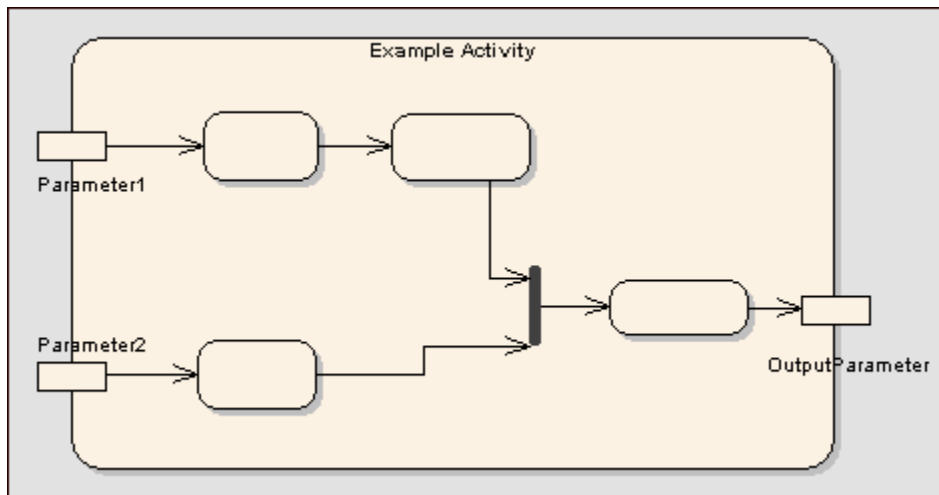
- [Activity](#)

5.3.3.2.2 Activity Parameter Nodes

[Further Information](#) | [OMG UML Specification](#)

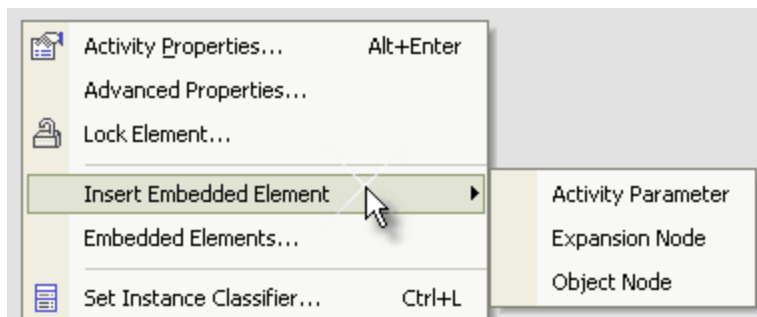
An activity parameter node is used for accepting input to an activity and providing output from an activity.

The following example depicts two entry parameters and one output parameter defined for the activity.



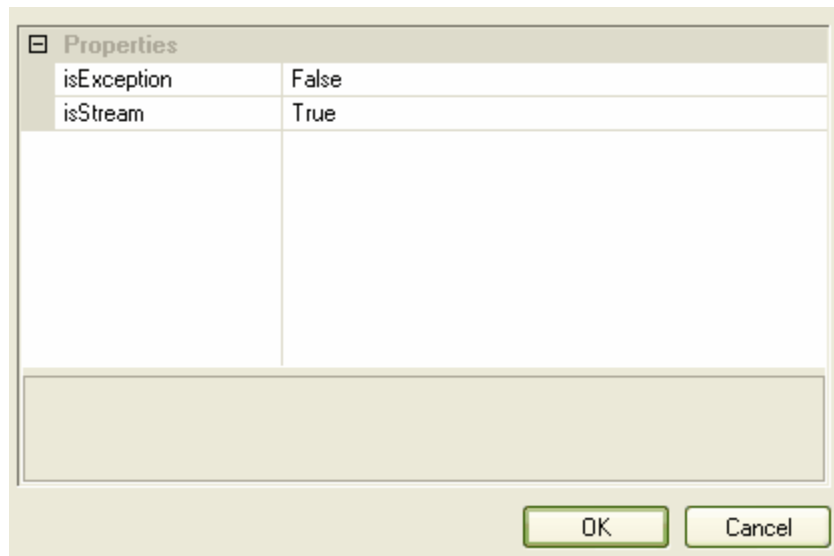
An activity parameter node can be defined for an activity by following these steps:

1. Right-click on the element, selecting *Insert Embedded Element* and *Activity Parameter*.



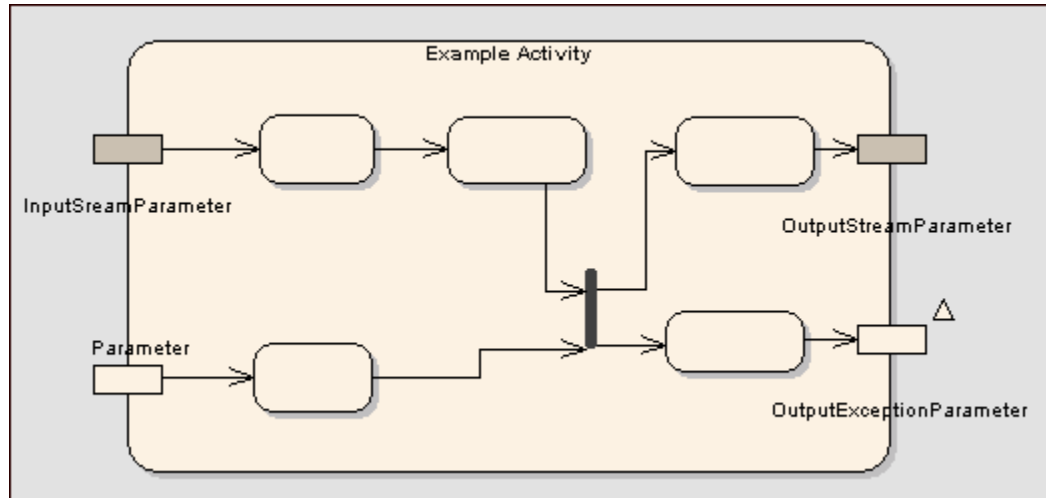
2. You will be prompted with the *Properties* dialog, which asks for the Name, etc. of the embedded element.

After closing this dialog, you can further define the new activity parameter. Right-click on the activity parameter, and select *Advanced Properties*, which opens the following dialog:



Similar to characterizing action pins, activity parameters also have the *isException* and *isStream* options. *isException* indicates that a parameter can emit a value at the exclusion of other outputs, usually because of some error. *isStream* indicates whether a parameter can accept or post values during the execution of the activity.

The following is an example using the above settings:



Further Information

- [Activity](#)
- [Action Pin](#)

OMG UML Specification

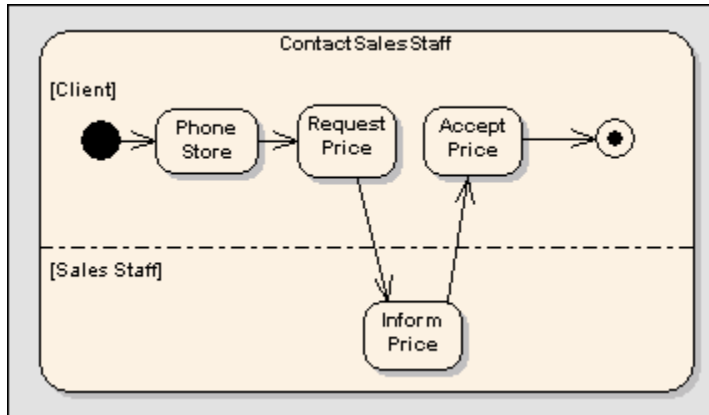
The OMG UML specification (*UML 2.0 Superstructure*, p. 304) states:

"An activity parameter node is an object node for inputs and outputs to activities....Activity parameters are object nodes at the beginning and end of flows, to accept inputs to an activity and provide outputs from it. (CompleteActivities) Activity parameters inherit support for streaming and exceptions from Parameter."

5.3.3.2.3 Activity Partition

[Further Information](#)

Activity partitions are used to logically organize an activity. They do not affect the token flow of an activity diagram, but help structure the view or parts of an activity. An example of a partitioned activity follows:



To define partitions:

1. Right-click on the activity element.
2. Select **Advanced | Partition Activity**. The **Activity Partitions** window will appear, as shown below.



- Activi
3. Enter the name of a partition and click **Save**.
 4. Repeat Step 3 for each partition to be created.

Further Information

- [Activity](#)

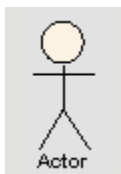
OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 307) states:

"Partitions divide the nodes and edges to constrain and show a view of the contained nodes. Partitions can share contents. They often correspond to organizational units in a business model. They may be used to allocate characteristics or resources among the nodes of an activity."

5.3.3.3 Actor

[Common Usage](#) | [OMG UML Specification](#)



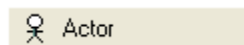
An actor is a user of the system; "user" can mean a human user, a machine, or even another system. Anything that interacts with the system from the outside or system boundary is termed an actor. Actors are typically associated with [Use Cases](#).

Actors may use the system through a graphical user interface, through a batch interface or through some other media. An actor's interaction with a use case is documented in a use case scenario and details the functions a system must provide to satisfy the user requirements.

Note: Remember, an actor can be a human user -OR- a machine -OR- another system -OR- another subsystem in the model.

Common Usage

- [Use Case](#)
- [Sequence Diagram](#)



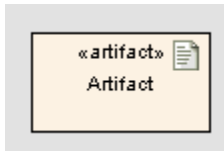
OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 5) states:

"A construct that is employed in use cases that define a role that a user or any other system plays when interacting with the system under consideration. It is a type of entity that interacts, but which is itself external to the subject. Actors may represent human users, external hardware, or other subjects. An actor does not necessarily represent a specific physical entity. For instance, a single physical entity may play the role of several different actors and, conversely, a given actor may be played by multiple physical entities."

5.3.3.4 *Artifact*

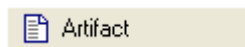
[Common Usage](#) | [OMG UML Specification](#)



An *artifact* is any physical piece of information used or produced by a system. Artifacts can have associated properties or operations, and can be instantiated or associated with other artifacts. Examples of artifacts include model files, source files, database tables, development deliverables or support documents.

Common Usage

- [Deployment Diagram](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 185) states:

"An Artifact defined by the user represents a concrete element in the physical world. A particular instance (or 'copy') of an artifact is deployed to a node instance. Artifacts may have composition associations to other artifacts that are nested within it. For instance, a deployment descriptor artifact for a component may be contained within the artifact that implements that component. In that way, the component and its descriptor are deployed to a node instance as one artifact instance."

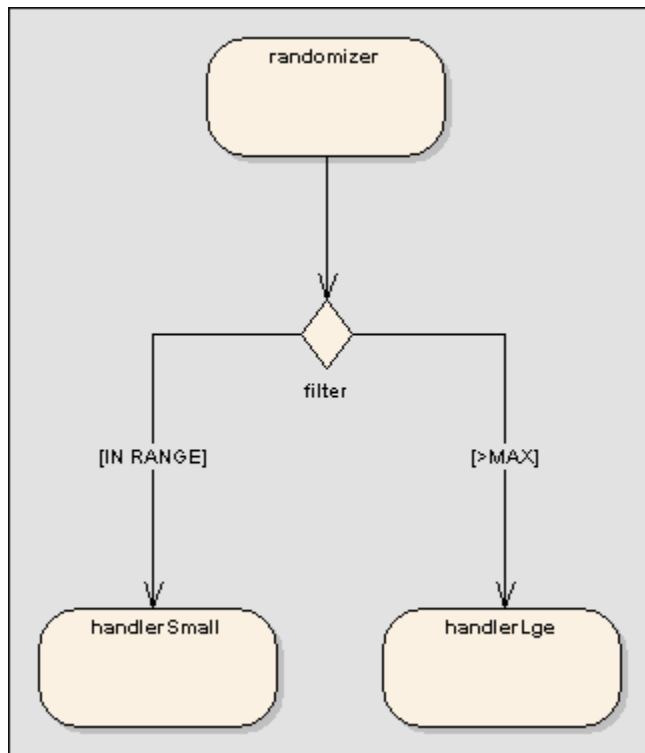
5.3.3.5 *Choice*

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



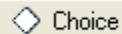
The *choice* pseudo-state is used to compose complex transitional paths, where the outgoing transition path is decided by dynamic, run-time conditions. The run-time conditions are determined by the actions performed by the state machine on the path leading to the choice.

The following example depicts the choice element. Upon reaching the filter pseudo-state, a transition will fire to the appropriate state based on the run-time value passed to the filter. Very similar in form to a junction pseudo-state, the choice pseudo-state's distinction is deciding transition paths at run-time.



Common Usage

- [State Machine Diagram](#)



Further Information

- [Pseudo-States](#)

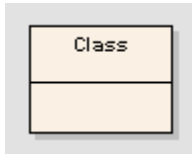
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"choice vertices which, when reached, result in the dynamic evaluation of the guards of the triggers of its outgoing transitions. This realizes a dynamic conditional branch. It allows splitting of transitions into multiple outgoing paths such that the decision on which path to take may be a function of the results of prior actions performed in the same run-to-completion step. If more than one of the guards evaluates to true, an arbitrary one is selected. If none of the guards evaluates to true, then the model is considered ill-formed. (To avoid this, it is recommended to define one outgoing transition with the predefined "else" guard for every choice vertex.) Choice vertices should be distinguished from static branch points that are based on junction points."

5.3.3.6 Class

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

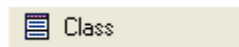


A *class* is a representation of object(s), that reflects their structure and behavior within the system. It is a template from which actual running instances are created. A class may have attributes (data) and methods (operations or behavior). Classes may inherit characteristics from parent classes and delegate behavior to other classes. Class models usually describe the logical structure of the system and are the building blocks from which components are built.

The top section of the class below shows the attributes (or data elements) associated with a class. These hold the 'state' of an object at run-time. If the information is saved to a data store and can be reloaded, it is termed 'persistent'. The lower section contains the class operations (or methods at run-time). Operations describe the behavior a class offers to other classes, and the internal behavior it has (private methods).

Common Usage

- [Class Diagram](#)
- [Composite Structure Diagram](#)



Further Information

- [Parameterized Classes \(Templates\)](#)
- [Active Classes](#)
- [Attributes](#)
- [Operations](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 87) states:

"The purpose of a class is to specify a classification of objects and to specify the features that characterize the structure and behavior of those objects.

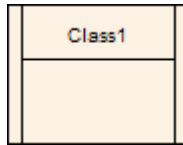
"Objects of a class must contain values for each attribute that is a member of that class, in accordance with the characteristics of the attribute, for example its type and multiplicity.

"When an object is instantiated in a class, for every attribute of the class that has a specified default, if an initial value of the attribute is not specified explicitly for the instantiation, then the default value specification is evaluated to set the initial value of the attribute for the object.

"Operations of a class can be invoked on an object, given a particular set of substitutions for the parameters of the operation. An operation invocation may cause changes to the values of the attributes of that object. It may also return a value as a result, where a result type for the operation has been defined. Operation invocations may also cause changes in value to the attributes of other objects that can be navigated to, directly or indirectly, from the object on which the operation is invoked, to its output parameters, to objects navigable from its parameters, or to other objects in the scope of the operation's execution. Operation invocations may also cause the creation and deletion of objects."

5.3.3.6.1 Active Classes

[Further Information](#) | [OMG UML Specification](#)



An *active class* indicates that, when instantiated, it will control its own execution. Rather than being invoked or activated by other objects, it can operate standalone, and define its own thread of behavior.

To define an active class in EA:

1. Highlight a class, and open its *Properties* dialog.
2. Press the *Advanced* button.
3. Check the *IsActive* check box.
4. Press *OK* to save the details.

Further Information

- [Class](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 4) states:

"An object that may execute its own behavior without requiring method invocation. This is sometimes referred to as "the object having its own thread of control." The points at which an active object responds to communications from other objects are determined solely by the behavior of the active object and not by the invoking object. This implies that an active object is both autonomous and interactive to some degree."

5.3.3.6.2 Parameterized Classes (Templates)

[Further Information](#) | [OMG UML Specification](#)

Enterprise Architect supports *template* or *parameterized classes*, which specify parameters that must be defined by any binding class. A template class allows for its functionality to be reused by any bound class. If a default value is specified for a parameter, and a binding class doesn't provide a value for that parameter, the default is used. Parameterized classes are commonly implemented in C++.

EA will import and generate templated classes for C++. Template classes are shown with the parameters in a dashed outline box in the upper right corner of a class.

To create a parameterized class:

1. Open the *Class Property* dialog for a class.
2. Select the *Detail* tab.
3. Under "Templates", select *Type* to be "Parameterized".
4. Define your parameters in the provided list dialog.

General **Detail** Require Constraints Link Scenario Files

Cardinality:

Visibility:

Concurrency

Sequential

Guarded

Active

Synchronous

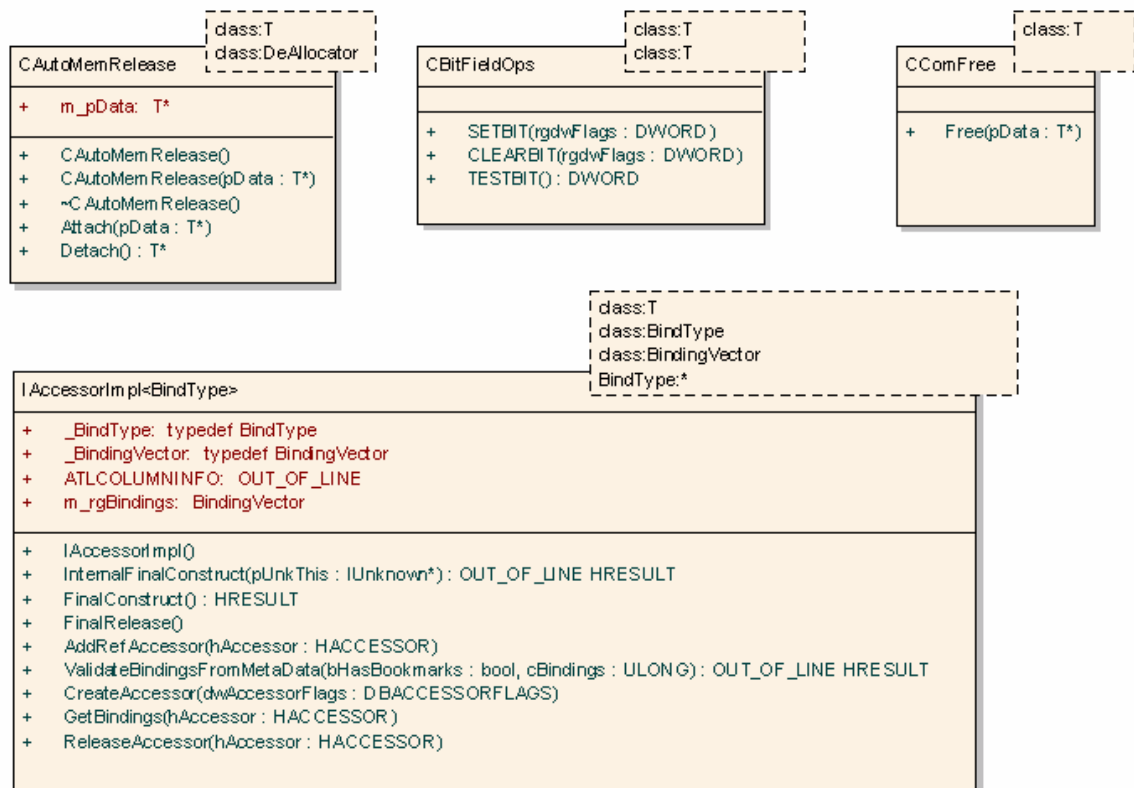
Templates

Type

| Parameter | Type | Default |
|-----------|-------|---------|
| BindType | Class | |

Arguments:

Notation example



Further Information

- [Class](#)

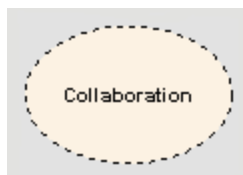
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 545) states:

"A template is a parameterized element that can be used to generate other model elements using TemplateBinding relationships. The template parameters for the template signature specify the formal parameters that will be substituted by actual parameters (or the default) in a binding."

5.3.3.7 Collaboration

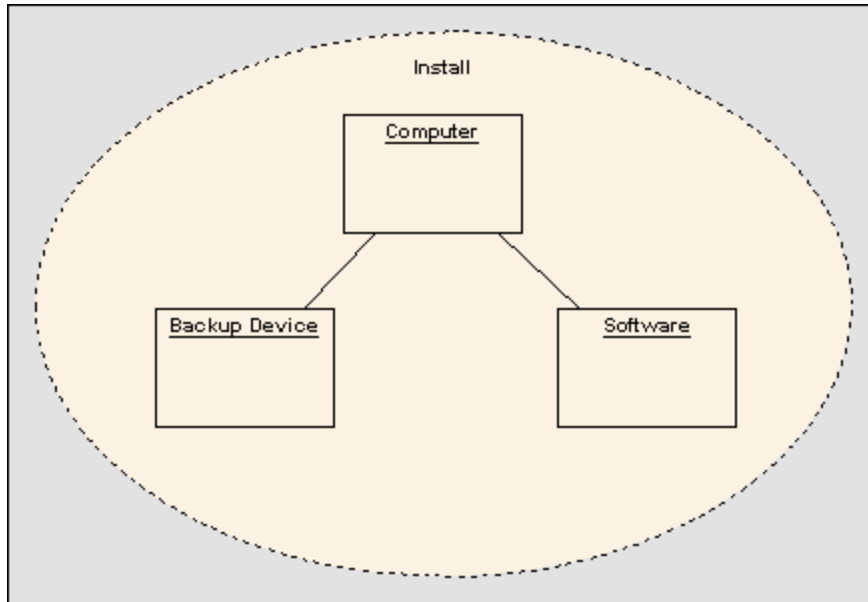
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *collaboration* defines a set of cooperating roles and their connectors. These are used to collectively illustrate a specific functionality. A collaboration should specify only the roles and attributes needed to accomplish a specific task or function. Although in practice a behavior and its roles could involve many tangential attributes

and properties, isolating the primary roles and their requisites simplifies and clarifies the behavior, as well as providing for reuse. A collaboration often implements a pattern to apply to various situations.

The following example illustrates an "Install" collaboration, with three roles connected as shown. The process for this collaboration can be demonstrated by attaching an interaction diagram.



To understand referencing a collaboration in a specific situation, see the topic [Collaboration Occurrence](#).

Common Usage

- [Composite Structure Diagrams](#)

 Collaboration

Further Information

- [Collaboration Occurrence](#)

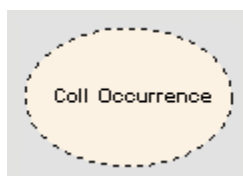
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 6) states:

"The specification of how an operation or classifier, such as a use case, is realized by a set of classifiers and associations playing specific roles used in a specific way. The collaboration defines an interaction."

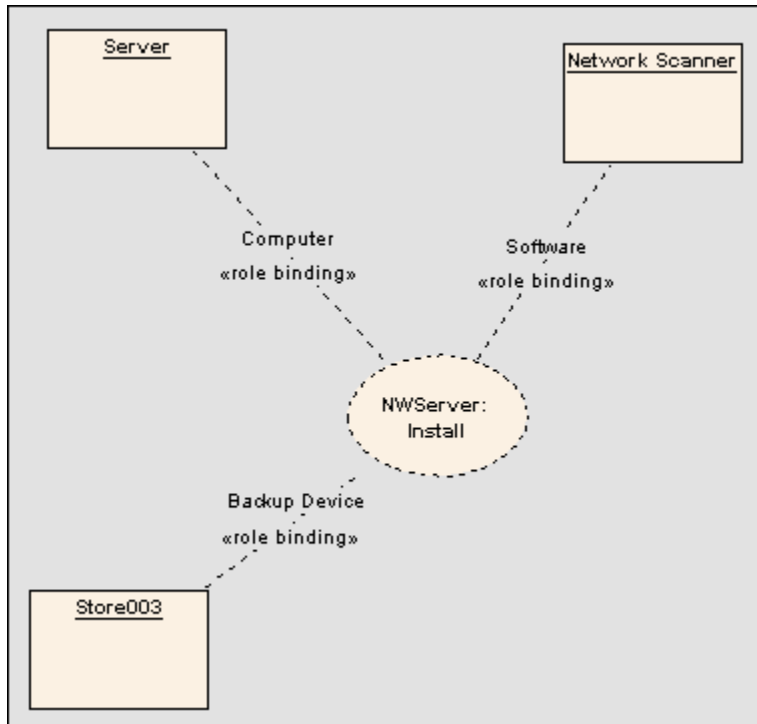
5.3.3.8 Collaboration Occurrence

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Use a *collaboration occurrence* to apply a pattern defined by a collaboration to a specific situation.

The following example uses an occurrence, 'NWServer', of the collaboration 'Install', to define the installation process of a network scanner. This process can be defined by an interaction attached to the collaboration. Refer to the [Collaboration](#) topic to see the 'Install' collaboration, depicted below.



Common Usage

- [Composite Structure Diagrams](#)

Further Information

- [Collaboration](#)

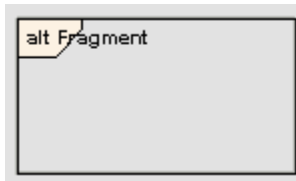
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 6) states:

"A particular use of a collaboration to explain the relationships between the parts of a classifier or the properties of an operation. It may also be used to indicate how a collaboration represents a classifier or an operation. A collaboration occurrence indicates a set of roles and connectors that cooperate within the classifier or operation according to a given collaboration, indicated by the type of the collaboration occurrence. There may be multiple occurrences of a given collaboration within a classifier or operation, each involving a different set of roles and connectors. A given role or connector may be involved in multiple occurrences of the same or different collaborations."

5.3.3.9 Combined Fragment

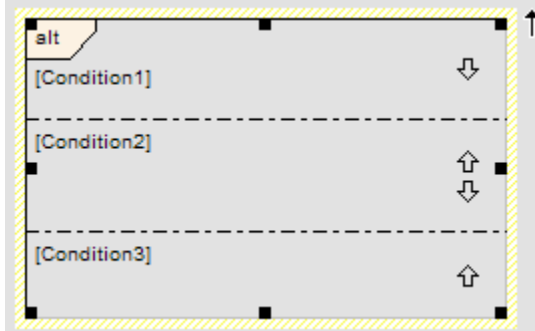
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *combined fragment* reflects a piece or pieces of interaction (called interaction operands) controlled by an interaction operator, whose corresponding boolean conditions are known as interaction constraints. It appears graphically as a transparent window, divided by horizontal dashed lines for each operand.

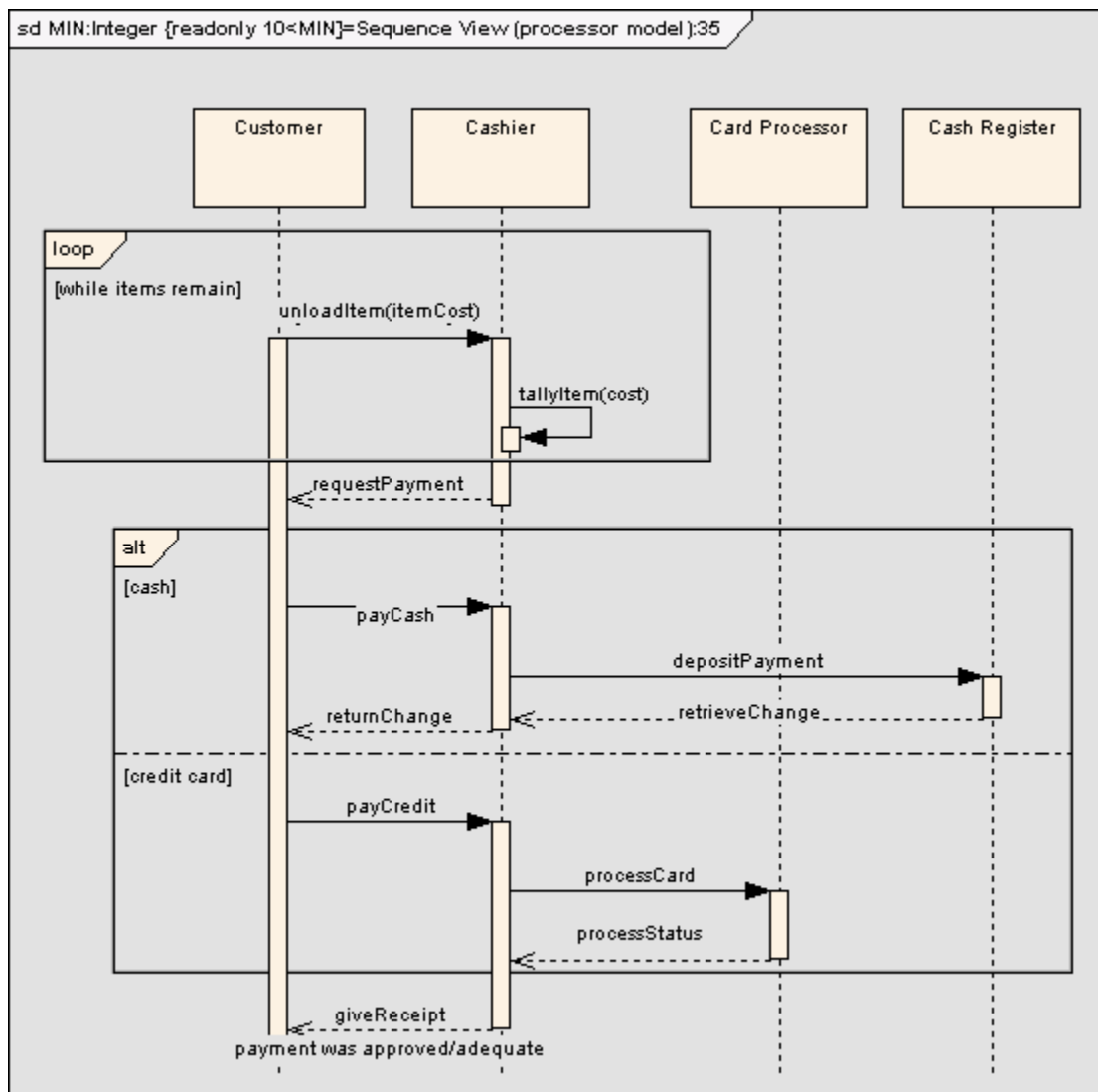
The following diagram exemplifies the use of combined fragments, with a sequence diagram modeling a simplified purchasing process. A loop fragment is utilized to iterate through an unknown amount of items for purchase, after which the cashier requests payment. At this point, two payment options are considered, and an alternative fragment is created, divided to show the two operands, cash and credit card. After the fragment completes its trace, the cashier gives a receipt to the customer, under the fulfilled condition that payment requirements were met.

The order of interaction fragment conditions can be changed directly on the diagram. Select an interaction fragment with more than one condition defined. Up and down arrows will appear on the right hand side of the each condition. Just click on the arrow to change the order.



Note: In order to select an interaction fragment, you need to click near the inside edge or drag a selection rectangle around the fragment. This was changed to prevent accidental selection when moving connectors inside the interaction fragment.

Tips & Tricks: By holding the *Alt* key, a combined fragment can be moved independently of its contents.

**Common Usage**

Fragment(s)

- [Sequence Diagram](#)

Further Information

- [Create a Combined Fragment](#)
- [Interaction Operators](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 409) states:

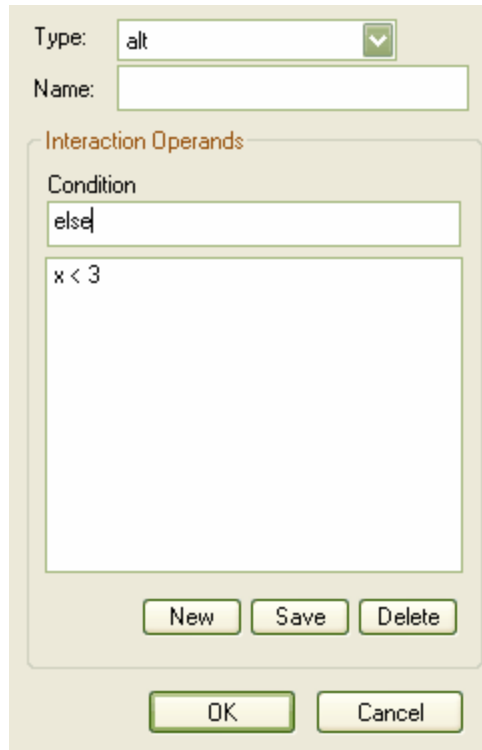
"A combined fragment defines an expression of interaction fragments. A combined fragment is defined by an interaction operator and corresponding interaction operands. Through the use of CombinedFragments the user will be able to describe a number of traces in a compact and concise manner."

5.3.3.9.1 Create a Combined Fragment

[Further Information](#)

Use the following guidelines to create a combined fragment in EA.

1. Drag the fragment element from the Interaction section of the toolbox. The following dialog will appear:



The dialog box is titled 'Create a Combined Fragment'. It features a 'Type' dropdown menu with 'alt' selected. Below it is a 'Name' text field. The 'Interaction Operands' section contains a 'Condition' text field with 'else' and a larger text area with 'x < 3'. At the bottom are buttons for 'New', 'Save', 'Delete', 'OK', and 'Cancel'.

2. Specify the interaction operator in the *Type* drop-down. See the [Interaction Operators](#) topic for an explanation of the various types of combined fragments.
3. A condition or interaction constraint must be specified for each operand.
4. A rectangular frame will appear, partitioned by dashed lines into segments for each operand.
5. Adjust the frame to encompass the desired event occurrences for each operand.

Further Information

- [Combined Fragment](#)
- [Interaction Operators](#)

5.3.3.9.2 Interaction Operators

[Further Information](#)

When creating combined fragments, you must apply an appropriate interaction operator to characterize the fragment. The following table provides rough guidance on the various operators, and their corresponding descriptions.

| Interaction Operator | Description |
|----------------------|--|
| alt | Divides up interaction fragments based on Boolean conditions. |
| opt | Encloses an optional fragment of interaction. |
| par | Indicates operands will operate in parallel. |
| loop | The operand will repeat a number of times, as specified by interaction constraints. |
| critical | Indicates a sequence that cannot be interrupted by other processing. |
| neg | Asserts that a fragment is invalid, and implies that all other interaction is valid. |
| assert | Specifies the only valid fragment to occur. Often enclosed within a consider or ignore operand. |
| strict | Indicates the behaviors of the operands must be processed in strict sequence. |
| seq | The combined fragment is weakly sequenced. This means that the ordering within operands is maintained, but the ordering between operands is undefined, so long as an event occurrence of the first operand precedes that of the second operand, if the event occurrences are on the same lifeline. |
| ignore | Indicates which messages should be ignored during execution, or can appear anywhere in the execution trace. |
| consider | Specifies which messages should be considered in the trace. This is often used to specify the resulting event occurrences with the use of an Assert operator. |
| ref | Provides a reference to another diagram. Note: The ref fragment is not created using the method described above. To create a ref fragment, simply drag an existing Diagram from the Project View onto the current diagram. |

Further Information

- [Combined Fragment](#)

OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 410-412) states:

"The semantics of a CombinedFragment is dependent upon the interactionOperator as explained below.

"Alternatives

The interactionOperator alt designates that the CombinedFragment represents a choice of behavior. At most one of the operands will execute. The operand that executes must have an explicit or implicit guard expression that evaluates to true at this point in the interaction. An implicit true guard is implied if the operand has no guard. The set of traces that defines a choice is the union of the (guarded) traces of the operands. An operand guarded by else designates a guard that is the negation of the disjunction of all other guards in the enclosing CombinedFragment.

"Option

The interactionOperator opt designates that the CombinedFragment represents a choice of behavior where either the (sole) operand happens or nothing happens. An option is semantically equivalent to an alternative CombinedFragment where there is one operand with non-empty content and the second operand is empty.

"Break

The interactionOperator break designates that the CombinedFragment represents a breaking scenario in the sense that the operand is a scenario that is performed instead of the remainder of the enclosing

InteractionFragment. Thus the break operator is a shorthand for an Alternative operator where one operand is given and the other assumed to be the rest of the enclosing *InteractionFragment*. Break *CombinedFragments* must be global relative to the enclosing *InteractionFragment*.

"Parallel

The interactionOperator *par* designates that the *CombinedFragment* represents a parallel merge between the behaviors of the operands. The eventoccurrences of the different operands can be interleaved in any way as long as the ordering imposed by each operand as such is preserved. A parallel merge defines a set of traces that describes all the ways that eventoccurrences of the operands may be interleaved without obstructing the order of the eventoccurrences within the operand.

"Weak Sequencing

The interactionOperator *seq* designates that the *CombinedFragment* represents a weak sequencing between the behaviors of the operands. Weak sequencing is defined by the set of traces with these properties:

1. The ordering of eventoccurrences within each of the operands are maintained in the result.
2. Eventoccurrences on different lifelines from different operands may come in any order.
3. Eventoccurrences on the same lifeline from different operands are ordered such that an eventoccurrence of the first operand comes before that of the second operand.

Thus weak sequencing reduces to a parallel merge when the operands are on disjunct sets of participants. Weak sequencing reduces to strict sequencing when the operands work on only one participant.

"Strict Sequencing

The interactionOperator *strict* designates that the *CombinedFragment* represents a strict sequencing between the behaviors of the operands. The semantics of the strict operation defines a strict ordering of the operands on the first level within the *CombinedInteraction* with operator *strict*. Therefore eventoccurrences within contained *CombinedInteractions* will not directly be compared with other eventoccurrences of the enclosing *CombinedInteraction*.

"Negative

The interactionOperator *neg* designates that the *CombinedFragment* represents traces that are defined to be invalid. The set of traces that defined a negative *CombinedFragment* is equal to the set of traces given by its (sole) operand, only that this set is a set of invalid rather than valid traces. All *InteractionFragments* that are different from *Negative* are considered positive meaning that they describe traces that are valid and should be possible.

"Critical Region

The interactionOperator *critical* designates that the *CombinedFragment* represents a critical region. A critical region means that the traces of the region cannot be interleaved by other Eventoccurrences (on those Lifelines covered by the region). This means that the region is treated atomically by the enclosing fragment when determining the set of valid traces. Even though enclosing *CombinedFragments* may imply that some Eventoccurrences may interleave into the region, such as e.g. with *par*-operator, this is prevented by defining a region. Thus the set of traces of enclosing constructs are restricted by critical regions.

"Ignore / Consider

The interactionOperator *ignore* designates that there are some message types that are not shown within this combined fragment. These message types can be considered insignificant and are intuitively ignored if they appear in a corresponding execution. Alternatively one can understand *ignore* to mean that the messages that are ignored can appear anywhere in the traces. Conversely the interactionOperator *consider* designates which messages should be considered within this *CombinedFragment*. This is equivalent to defining every other message to be ignored.

"Assertion

The interactionOperator *assert* designates that the *CombinedFragment* represents an assertion. The sequences of the operand of the assertion are the only valid continuations. All other continuations result in an invalid trace. Assertions are often combined with *Ignore* or *Consider* as shown in Figure 345.

"Loop

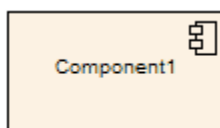
The interactionOperator *loop* designates that the *CombinedFragment* represents a loop. The loop operand will be repeated a number of times. The Guard may include a lower and an upper number of iterations of the loop as well as a Boolean expression. The semantics is such that a loop will iterate

minimum the 'minint' number of times (given by the iteration expression in the guard) and at most the 'maxint' number of times. After the minimum number of iterations have executed, and the boolean expression is false the loop will terminate. The loop construct represent a recursive application of the seq operator where the loop operand is sequenced after the result of earlier iterations.

"The Semantics of Gates (see also 'Gate (from Fragments)' on page 418)
The gates of a CombinedFragment represent the syntactic interface between the CombinedFragment and its surroundings, which means the interface towards other InteractionFragments. The only purpose of gates is to define the source and the target of Messages or General Order relations."

5.3.3.10 Component

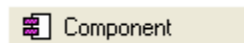
[Common Usage](#) | [OMG UML Specification](#)



A *component* is a modular part of a system, whose behavior is defined by its provided and required interfaces; the internal workings of the component should be invisible and its usage environment-independent. Source code files, DLLs, Java beans and other artifacts defining the system may be manifested in components.

A component may be composed of multiple classes, or components pieced together. As smaller components come together to create bigger components, the eventual system can be modeled, building-block style. By building the system in discrete components, localization of data and behavior allows for decreased dependency between classes and objects, providing a more robust and maintainable design.

Common Usage



- [Component Diagram](#)

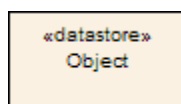
OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 7) states:

"A modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. As such, a component serves as a type, whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics)."

5.3.3.11 Datastore

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



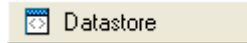
A *datastore* is an element used to define permanently stored data. A token of data that enters into a datastore is stored permanently, updating tokens for that data that already exists. A token of data that comes out of a datastore is a copy of the original data.

Use object flow connectors to link elements to datastores, as values and information are being passed between nodes. Selection and transformation behavior, together composing a sort of query, can be specified

as to the nature of data access. For instance, selection behavior determines which objects are affected by the connection to the datastore. Transformation behavior might then further specify the value of an attribute pertaining to a selected object.

To define the behavior of access to a datastore, attach a note to the object flow connector. To do this, right-click on the object-flow and select *Attach Note or Constraint*. A dialog will indicate other flows in the diagram, to which you can select to attach the note, if the behavior applies to multiple flows. To comply with UML 2, preface behavior with the notation <<selection>> or <<transformation>>.

Common Usage



- [Activity Diagrams](#)

Further Information

- [Activity Diagrams](#)
- [Activity](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 318) states:

"A data store node is a central buffer node for non-transient information... A data store keeps all tokens that enter it, copying them when they are chosen to move downstream. Incoming tokens containing a particular object replace any tokens in the object node containing that object."

5.3.3.12 Decision

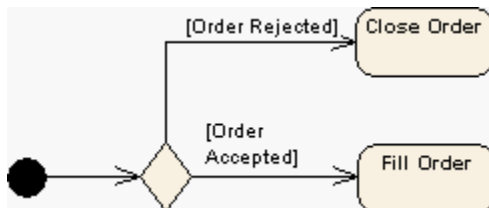
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *decision* is an element of an Activity diagram that indicates a point of conditional progression: if a condition is true, then processing continues one way, if not, then another.

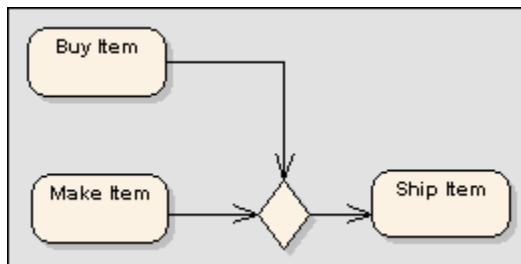
This can also be used as a merge node in that multiple alternate flows can be merged (but not synchronized) to form one flow. The following are examples of these two modes of using the decision element.

Used as a decision:



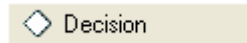
Refer to figure 238 (UML 2.0 Superstructure, p. 321).

Used as a merge:



Refer to figure 266 (UML 2.0 Superstructure, p. 344).

Common Usage



- [Activity Diagrams](#)
- [Interaction Overview](#)

Further Information

- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 319 (*Decision symbol*)) states:

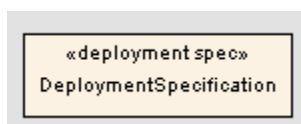
"A decision node is a control node that chooses between outgoing flows. A decision node has one incoming edge and multiple outgoing activity edges."

The OMG UML specification (*UML 2.0 Superstructure*, p. 343 (*Merge symbol*)) also states:

"A merge node is a control node that brings together multiple alternate flows. It is not used to synchronize concurrent flows but to accept one among several alternate flows...A merge node has multiple incoming edges and a single outgoing edge."

5.3.3.13 Deployment Spec

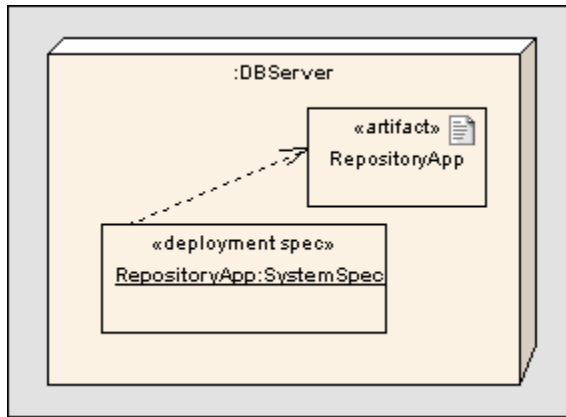
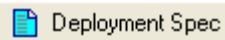
[Common Usage](#) | [OMG UML Specification](#)



A *deployment specification (spec)* specifies parameters guiding deployment of an artifact, as is necessary with most hardware and software technologies. A specification lists those properties that must be defined for deployment to occur. An instance of this specification specifies the values for the parameters; a single specification can be instantiated for multiple artifacts.

These specifications can be extended by certain component profiles. Examples of standard tagged values that a profile might add to a deployment specification are «concurrencyMode» with tagged values {thread, process, none} or «transactionMode» with tagged values {transaction, nestedTransaction, none}.

The following example depicts the artifact RepositoryApp deployed on the server node, per specifications of RepositoryApp, instantiated from the deployment specification SystemSpec.

**Common Usage**

- [Deployment Diagrams](#)

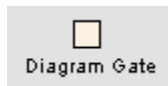
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 190) states:

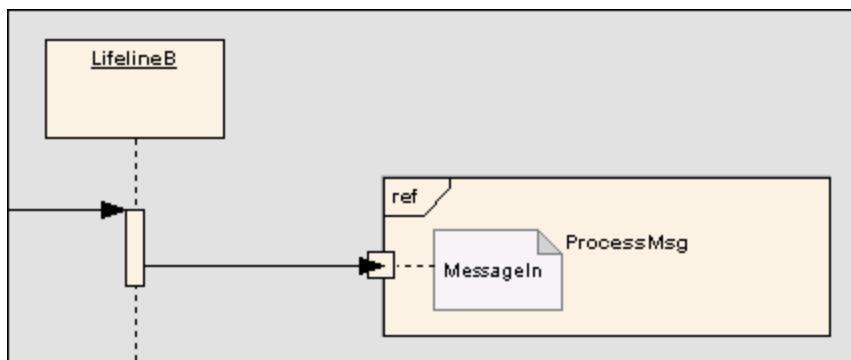
"A deployment specification specifies a set of properties which determine execution parameters of a component artifact that is deployed on a node. A deployment specification can be aimed at a specific type of container. An artifact that reifies or implements deployment specification properties is a deployment descriptor."


5.3.3.14 Diagram Gate

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *diagram gate* is a simple graphical way to indicate the point at which messages can be transmitted into and out of interaction fragments. A fragment might need to receive or deliver a message; internally, an ordered message reflects this need, with a gate indicated on the boundary of the fragment's frame. Any external messages 'synching' with this internal message must correspond appropriately. Gates can appear on interactions, interaction occurrences and combined fragments (to specify the expression).



Common Usage □ Diagram Gate

- [Timing Diagrams](#)
- [Sequence Diagrams](#)

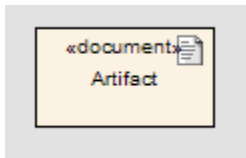
Further Information

- [Interactions](#)
- [Interaction Occurrences](#)
- [Combined Fragments](#)

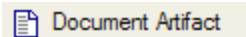
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 418) states:

"A Gate is a connection point for relating a Message outside an InteractionFragment with a Message inside the InteractionFragment... Gates are connected through Messages. A Gate is actually a representative of an EventOccurrence that is not in the same scope as the Gate. Gates play different roles: we have formal gates on Interactions, actual gates on InteractionOccurrences, expression gates on CombinedFragments."

5.3.3.15 Document Artifact[Common Usage](#) | [OMG UML Specification](#)

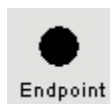
A *document artifact* is an [artifact](#) having a stereotype of *document*. The *document artifact* associates with an RTF document. By double clicking on this element, you will be confronted with a RTF word processor. see [Linking an RTF Document to an Element](#).

Common Usage Document Artifact

- [Component Diagram](#)
- [Deployment Diagram](#)

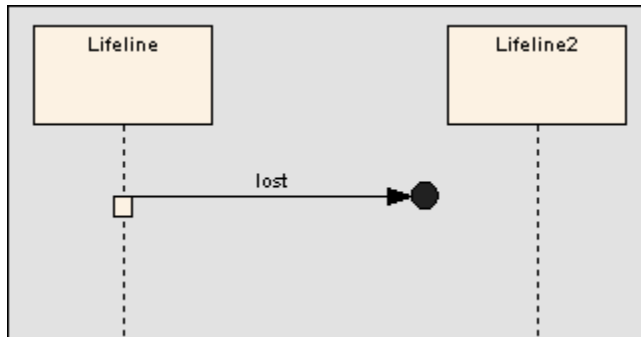
See Also

- [Artifact](#)
- [Linking an RTF Document to an Element](#).


5.3.3.16 Endpoint[Common Usage](#) | [OMG UML Specification](#)

An *endpoint* is used in interaction diagrams to reflect a lost or found message in sequence. To model this, drag an endpoint element onto the workspace. With Sequence diagrams, drag a message from the appropriate lifeline to the endpoint. With Timing diagrams, the message connecting the lifeline to the endpoint will require some timing specifications to draw the connection.

The following example depicts an lost message in a Sequence diagram.



Common Usage

 Endpoint

- [Sequence Diagram](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 429) states:

"A lost message is a message where the sending event occurrence is known, but there is no receiving event occurrence. We interpret this to be because the message never reached its destination."

"A found message is a message where the receiving event occurrence is known, but there is no (known) sending event occurrence. We interpret this to be because the origin of the message is outside the scope of the description. This may for example be noise or other activity that we do not want to describe in detail."

5.3.3.17 Entry Point

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Entry points are used to define the beginning of a state machine. An entry point exists for each region, directing the initial concurrent state configuration.

Common Usage

 Entry

- [State Machine Diagram](#)

Further Information

- [Pseudo-States](#)

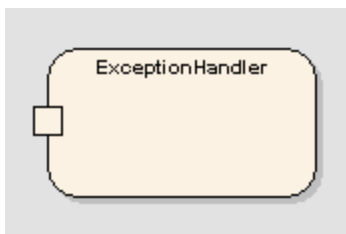
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"An entry point pseudostate is an entry point of a state machine. In each region of the state machine it has a single transition to a vertex within the same region."


5.3.3.18 Exception

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



The *exception handler* element defines the group of operations to carry out when an exception occurs. The protected element may contain a set of operations and is linked to the exception handler via an interrupt flow connector. Any defined error contained within an element's parts can trigger the flow to move to an exception.

Common Usage

 Exception

- [Activity Diagrams](#)

Further Information

- [Interruptible Activity Region](#)
- [Interrupt Flow](#)
- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 8) states:

"A special kind of signal, typically used to signal fault situations. The sender of the exception aborts execution and execution resumes with the receiver of the exception, which may be the sender itself. The receiver of an exception is determined implicitly by the interaction sequence during execution; it is not explicitly specified."

5.3.3.19 Expansion Region

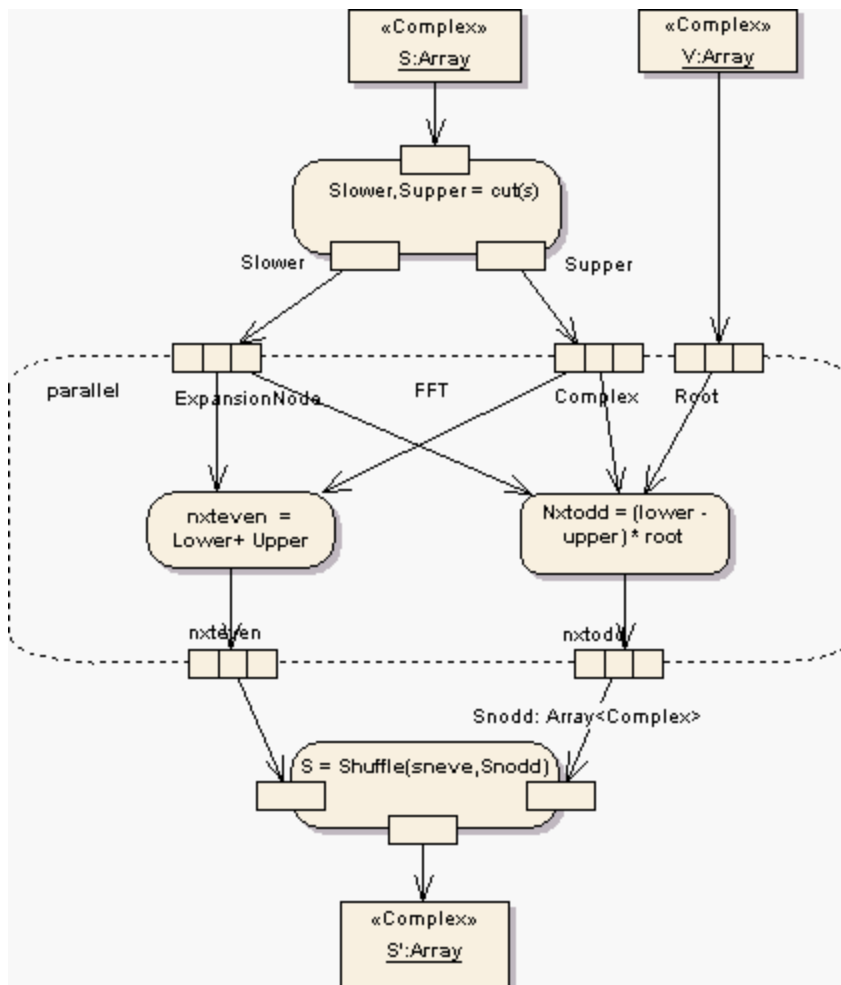
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *expansion region* surrounds a process to be imposed multiple times on the incoming data, once for every element in the input collection. If there are multiple inputs, the collection sizes must match, and the elements within each collection must be of the same type. Similarly, any outputs must be in the form of a collection matching the size of the inputs.

The concurrency of the expansion region's multiple executions can be specified as type parallel, iterative, or stream. Parallel reflects that the elements in the incoming collections can be processed at the same time or overlapping, whereas an iterative concurrency type specifies that execution must occur sequentially. A stream-type expansion region indicates that the input and output come in and exit as streams, and that the expansion region's process must have some method to support streams.

Modify the mode of an expansion region by *Right-Clicking* and selecting *Advanced | Custom Properties* on an expansion region.



Refer to figure 247 (UML 2.0 Superstructure, p. 330).

Common Usage  Region

- [Activity Diagrams](#)

Further Information

- [Add an Expansion Region](#)
- [Activity Diagrams](#)
- [Regions](#)

The OMG UML specification states:

The OMG UML specification (*UML 2.0 Superstructure*, p. 325) states:

"An expansion region is a structured activity region that executes multiple times corresponding to elements of an input collection."

5.3.3.19.1 Add Expansion Region[Further Information](#)

After adding a region element to the diagram, the following prompt appears:

By default this is set to InterruptibleActivityRegion.

1. Select the type: *ExpansionRegion*.
2. Choose the concurrency attribute by selecting an entry from the drop down *Kind*.

Further Information

- [Expansion Region](#)

5.3.3.20 Exit Point

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Exit points are used in submachine states and state machines to denote the point where the machine will be exited and the transition sourcing this exit point, for submachines, will be triggered. Exit points are a type of pseudo-state used in the state machine diagram.

Common Usage

- [State Machine Diagram](#)

Further Information

- [Pseudo-States](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"An exit point pseudostate is an exit point of a state machine. Entering an exit point within any region of the state machine referenced by a submachine state implies the exit of this submachine state and the triggering of the transition that has this exit point as source in the state machine enclosing the submachinestate."

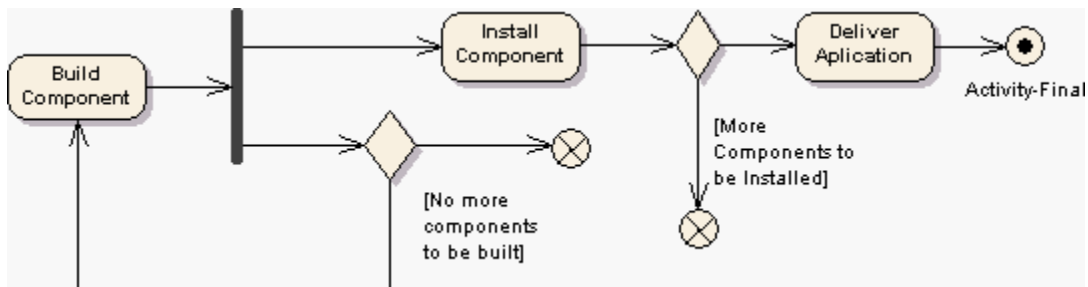
5.3.3.21 Final

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

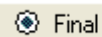


There are two nodes used to define a *final* state in an activity, both defined in UML 2.0 as of type "final node". The final element, shown above, indicates the completion of an activity - upon reaching the final, all execution in the activity diagram is aborted. The other type of final node, [flow final](#), depicts an exit from the system but has no effect on other executing flows in the activity.

The following example illustrates the development of an application. The process comes to a flow final node when there are no more components to be built; note that the fork element indicates a concurrent process with the building of new components and installation of completed components. The flow final terminates only the sub-process building components. Similarly, only those tokens entering the decision branch, for the installation of further components, terminate with the connecting flow final. It is only after the Deliver Application activity is completed, after the control flow reaches the final node, that all flows stop.



Refer to figure 251 (UML 2.0 Superstructure, p. 332).

Common Usage

- [Activity Diagram](#)

Further Information

- [Activity Diagrams](#)
- [Flow Final](#)

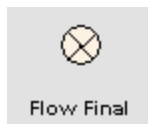
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 298) states:

"An activity may have more than one activity final node. The first one reached stops all flows in the activity."

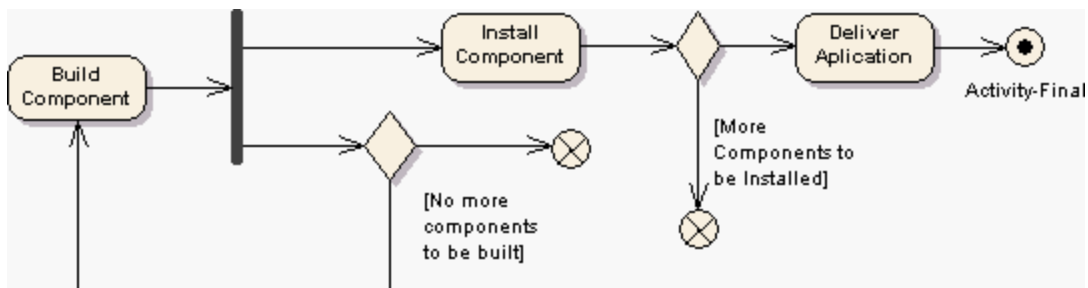
5.3.3.22 Flow Final

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



There are two nodes used to define a *final* state in an activity, both defined in UML 2.0 as of type "final node". The *flow final* element depicts an exit from the system as opposed to the activity final which represents the completion of the activity. Only the flow entering the flow final node exits the activity; other flows continue undisturbed.

The following example illustrates the development of an application. The process comes to a flow final node when there are no more components to be built; note that the fork element indicates a concurrent process with the building of new components and installation of completed components. The flow final terminates only the sub-process building components. Similarly, only those tokens entering the decision branch, for the installation of further components, terminate with the connecting flow final. It is only after the Deliver Application activity is completed, after the control flow reaches the final node, that all flows stop.



Refer to figure 251 (UML 2.0 Superstructure, p. 332).

Common Usage

- [Activity Diagrams](#)

Further Information

- [Activity Diagrams](#)
- [Activity Final](#)

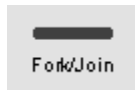
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 333) states:

"A flow final destroys all tokens that arrive at it. It has no effect on other flows in the activity."

5.3.3.23 Fork/Join

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



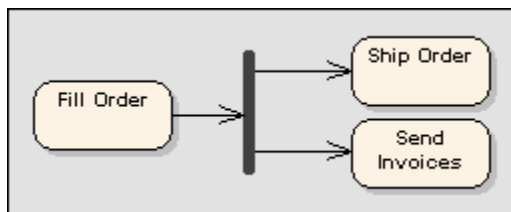
The *fork/join* elements have different modes of use. These are as follows:

- To fork or split the flow into a number of concurrent flows.
- To join the flow of a number of concurrent flows.
- To both join and fork a number of incoming flows to a number of outgoing flows.

With respect to State Machine diagrams, forks and joins are utilized as pseudo-states. Other pseudo-states include history states, entry points and exit points. Forks are used to split an incoming transition into concurrent multiple transitions leading to different target states. Joins are used to merge concurrent multiple transitions into a single transition leading to a single target. They are semantic inverses. To learn more about these individual elements, consult their topics, listed [below](#).

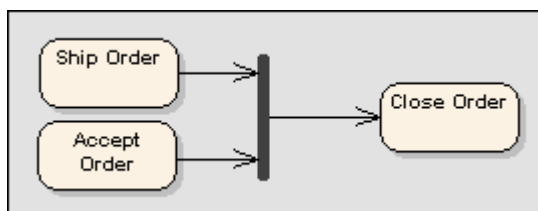
Some examples of fork/join nodes include:

Fork or split the flow into a number of concurrent flows:



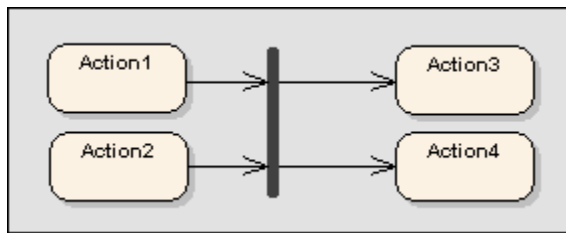
Refer to figure 255 (UML 2.0 Superstructure, p. 335).

Join the flow of a number of concurrent flows:



Refer to figure 263 (UML 2.0 Superstructure, p. 340).

Join and fork a number of incoming flows to a number of outgoing flows:

**Common Usage**

- [Activity Diagrams](#)
- [State Machine Diagram](#)

Further Information

- [Fork](#)
- [Join](#)
- [Pseudo-States](#)

OMG UML Specification**Fork**

The OMG UML specification (*UML 2.0 Superstructure*, p. 333) states:

"A fork node is a control node that splits a flow into multiple concurrent flows... A fork node has one incoming edge and multiple outgoing edges."

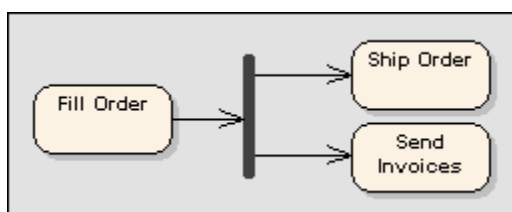
Join

The OMG UML specification (*UML 2.0 Superstructure*, p. 338-339) states:

"A join node is a control node that synchronizes multiple flows...A join node has multiple incoming edges and one outgoing edge."

5.3.3.23.1 Fork

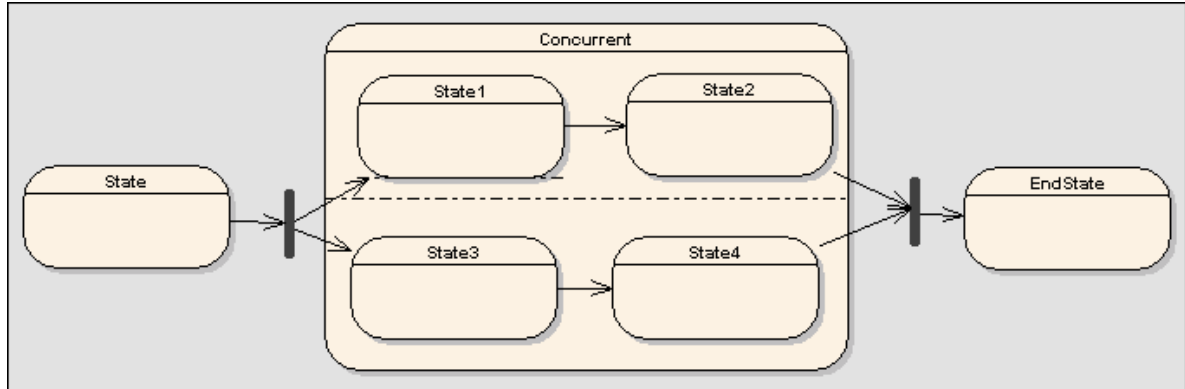
[Further Information](#) | [OMG UML Specification](#)



Refer to figure 255 (UML 2.0 Superstructure, p. 335).

With respect to state machine diagrams, a *fork* pseudo-state signifies that its incoming transition will come

from a single state, and it will have multiple outgoing transitions. These transitions must occur concurrently, requiring the use of concurrent regions, as depicted below in the composite state. Unlike choice or junction pseudo-states, forks may not have triggers or guards. The following diagram demonstrates a fork pseudo-state dividing into two concurrent regions, which then return to the EndState via the join.



Further Information

- [Pseudo-States](#)
- [Fork/Join](#)
- [Regions](#)

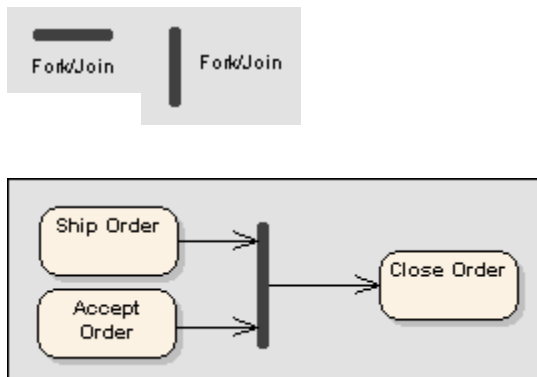
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"fork vertices serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices (i.e. vertices in different regions of a composite state). The segments outgoing from a fork vertex must not have guards or triggers."

5.3.3.2.3.2 Join

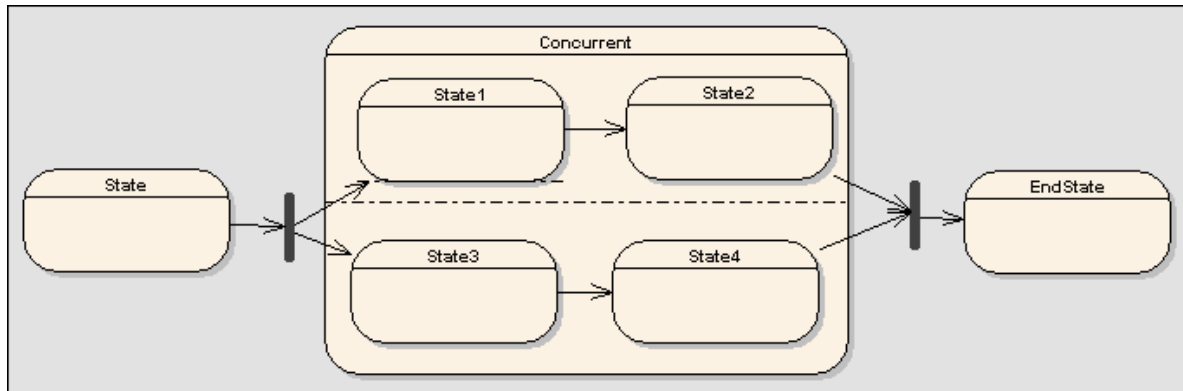
[Further Information](#) | [OMG UML Specification](#)



Refer to figure 263 (UML 2.0 Superstructure, p. 340).

The *join* element is utilized by the activity and state machine diagrams. The above example illustrates a join transition between activities. With respect to state machine diagrams, a join pseudo-state indicates multiple states concurrently transitioning into the join and onto a single state. Unlike choice or junction pseudo-states,

joins may not have triggers or guards. The following diagram demonstrates a fork pseudo-state dividing into two concurrent regions, which then return to the EndState via the join.



Further Information

- [Pseudo-States](#)
- [Fork/Join](#)
- [Regions](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"join vertices serve to merge several transitions emanating from source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers."

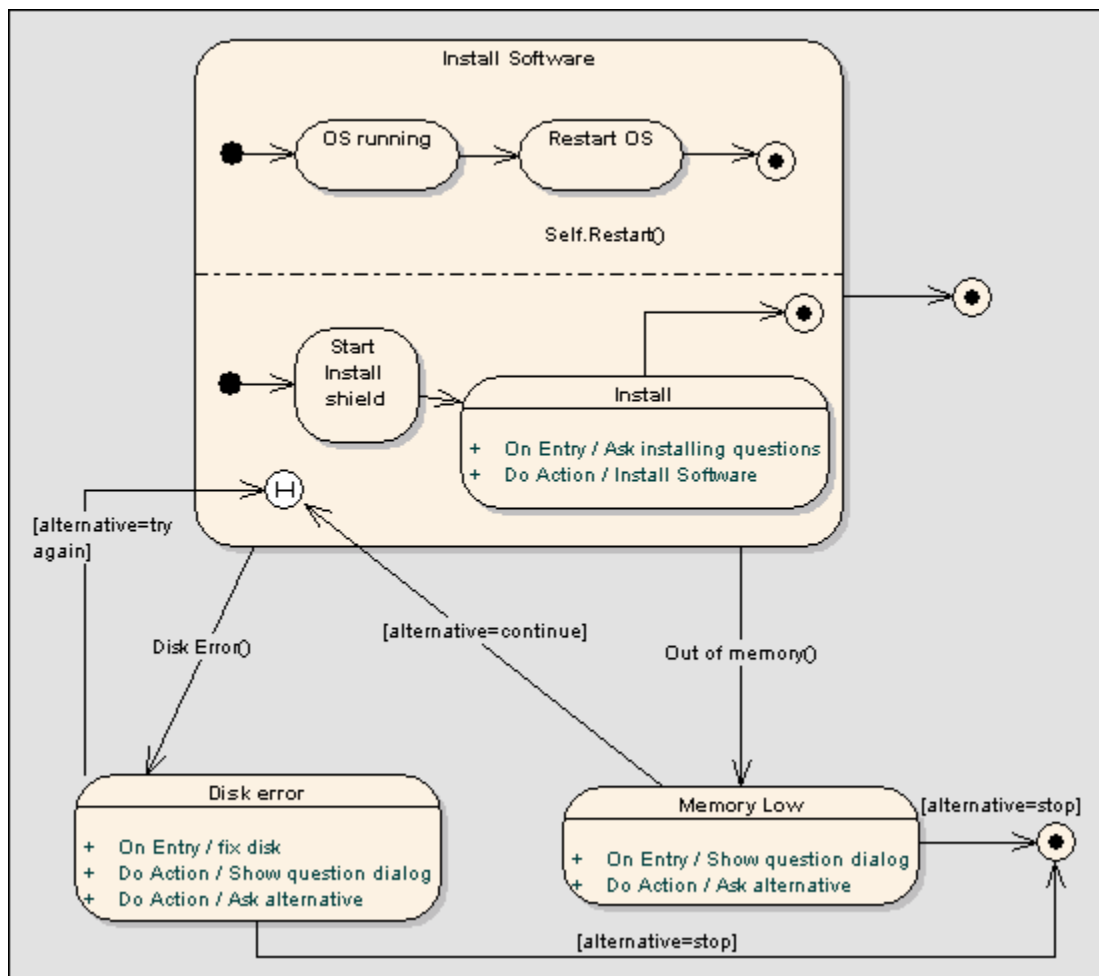
5.3.3.24 History

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



There are two types of *history* pseudo-states defined in UML, shallow and deep history. A shallow history sub-state is used to represent the most recently active sub-state of a composite state; this pseudo-state does not recurse into this sub-state's active configuration, should one exist. A single connector can be used to depict the default shallow history state, in case the composite state has never been entered.

A deep history sub-state, in contrast, reflects the most recent active configuration of the composite state. This includes active sub-states of all regions, and recurses into those sub-states' active sub-states, should they exist. At most one deep history and one shallow history can dwell within a composite state.

**Common Usage**

History

- [State Machine Diagram](#)

Further Information

- [Pseudo-States](#)

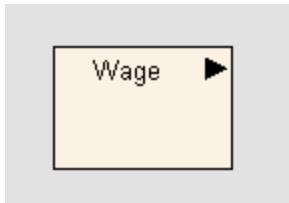
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 470) states:

"deepHistory represents the most recent active configuration of the composite state that directly contains this pseudostate; e.g. the state configuration that was active when the composite state was last exited. A composite state can have at most one deep history vertex. At most one transition may originate from the history connector to the default deep history state. This transition is taken in case the composite state had never been active before. Entry actions of states entered on the path to the state represented by a deep history are performed. shallowHistory represents the most recent active substate of its containing state (but not the substates of that substate). A composite state can have at most one shallow history vertex. A transition coming into the shallow history vertex is equivalent to a transition coming into the most recent active substate of a state. At most one transition may originate from the history connector to the default shallow history state. This transition is taken in case the composite state had never been active before. Entry actions of states entered on the path to the state represented by a shallow history are performed."

5.3.3.25 Information Item

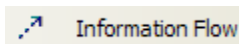
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An InformationItem represents an abstraction of data.

Common Usage

- [Activity Diagram](#)



Further Information

- [Information Flow](#)

OMG UML Specification

The OMG UML specification (*UML 2.1 Superstructure*, p. 636) states:

“An information item is an abstraction of all kinds of information that can be exchanged between objects. It is a kind of classifier intended for representing information at a very abstract way, one which cannot be instantiated.

One purpose of information items is to be able to define preliminary models, before having made detailed modeling decisions on types or structures. One other purpose of information items and information flows is to abstract complex models by a less precise but more general representation of the information exchanged between entities of a system.”

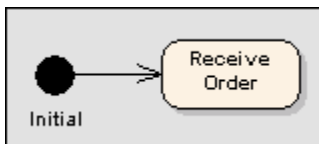
5.3.3.26 Initial

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



The *initial* element is used by the Activity and State Machine diagrams. In Activity diagrams, it defines the start of a flow when an activity is invoked. With State Machines, the initial element is a [pseudo-state](#) used to denote the default state of a composite state; there can be one initial vertex in each region of the composite state.

This simple example shows the start of a flow to receive an order.



Refer to figure 257 (UML 2.0 Superstructure, p. 336).

Common Usage

- [State Machine Diagram](#)
- [Activity Diagram](#)

Further Information

- [Pseudo-States](#)
- [Final](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 470) states:

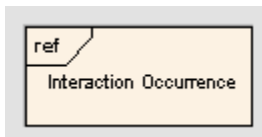
"An initial pseudostate represents a default vertex that is the source for a single transition to the default state of a composite state. There can be at most one initial vertex in a region. The initial transition may have an action."

The OMG UML specification (*UML 2.0 Superstructure*, p. 335) states:

"An initial node is a control node at which flow starts when the activity is invoked."

5.3.3.27 Interaction Occurrence

[Common Usage](#) | [OMG UML Specification](#)



An *interaction occurrence* is a reference to an existing interaction element. Interaction occurrences are visually represented by a frame, with "ref" in the frame's title space. The diagram name is indicated in the frame contents. To create an interaction occurrence, simply drag an interaction diagram onto an open interaction diagram's workspace. A dialog will open, providing configuration options.

Common Usage

- [Sequence Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 423) states:

"An InteractionOccurrence refers to an Interaction. The InteractionOccurrence is a shorthand for copying the contents of the referred Interaction where the InteractionOccurrence is. To be accurate the copying must take into account substituting parameters with arguments and connect the formal gates with the actual ones. It is common to want to share portions of an interaction between several other interactions. An InteractionOccurrence allows multiple interactions to reference an interaction that represents a common portion of their specification."

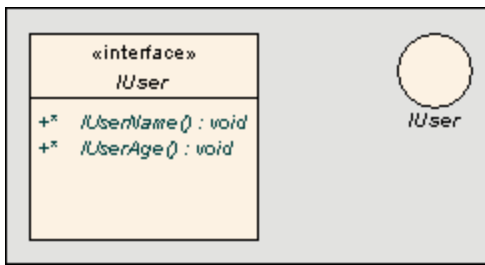
5.3.3.28 Interface

[Common Usage](#) | [OMG UML Specification](#)



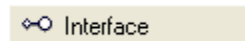
An *interface* is a specification of behavior that implementers agree to meet. It is a contract. By implementing an interface, classes are guaranteed to support a required behavior, which allows the system to treat non-related elements in the same way - ie. through the common interface.

Interfaces may be drawn in a similar method to a class, with operations specified, as shown below. They may also be drawn as a circle with no explicit operations detailed. Use the right-click context menu option *Use Circle Notation* to switch between styles. When drawn as a circle, realization links to the circle form of notation are drawn without target arrows.



Note: An interface cannot be instantiated (ie. you cannot create an object from an Interface). You must create a class that 'implements' the interface specification, and in the class body place operations for each of the interface operations. You can then instantiate the class.

Common Usage



- [Composite Structure Diagram](#)
- [Class Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 114) states:

"An interface is a kind of classifier that represents a declaration of a set of coherent public features and obligations. In a sense, an interface specifies a kind of contract which must be fulfilled by any instance of a classifier that realizes the interface. The obligations that may be associated with an interface are in the form of various kinds of constraints (such as pre- and post-conditions) or protocol specifications, which may impose ordering restrictions on interactions through the interface. Since interfaces are declarations, they are not directly instantiable. Instead, an interface specification is realized by an instance of a classifier, such as a class, which means that it presents a public facade that conforms to the interface specification. Note that a given classifier may realize more than one interface and that an interface may be realized by a number of different classifiers."

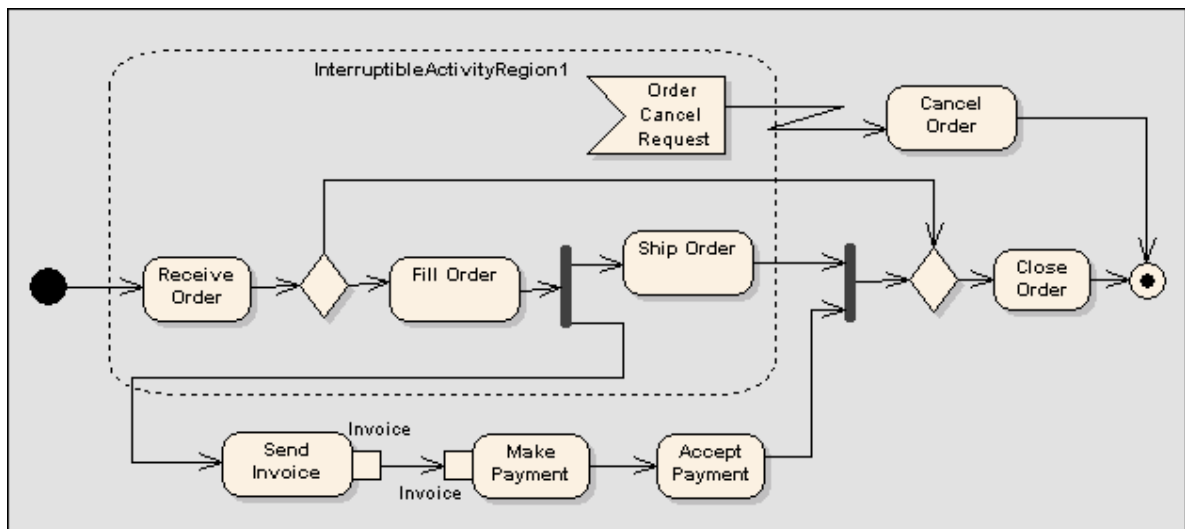
5.3.3.29 Interruptible Activity Region

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *interruptible activity region* surrounds a group of activity elements, all affected by certain interrupts in such a way that all tokens passing within the region are terminated should the interruption(s) be raised. Any processing occurring within the bounds of an interruptible activity region is terminated when a flow is instigated across an interrupt flow to an external element.

The example below illustrates that an order cancellation will kill any processing of the order at the receipt, filling or shipping stage.



Refer to figure 260 (UML 2.0 Superstructure, p. 338).

Common Usage Region

- [Activity Diagram](#)

Further Information

- [Add an Interruptible Activity Region](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 337) states:

"An interruptible region contains activity nodes. When a token leaves an interruptible region via edges designated by the region as interrupting edges, all tokens and behaviors in the region are terminated."

5.3.3.29.1 Add Interruptible Activity Region

[Further Information](#)

After adding a region element to an activity diagram, the following prompt appears:



The dialog box has a title bar and a light beige background. It contains the following elements:

- A label "Select type:" followed by two radio buttons:
 - InterruptibleActivityRegion
 - ExpansionRegion
- A label "Kind:" followed by a dropdown menu showing "iterative".
- Two buttons on the right: "OK" and "Cancel".

By default this is set to `InterruptibleActivityRegion` and the *Kind* option is disabled. Press *OK*.

Further Information

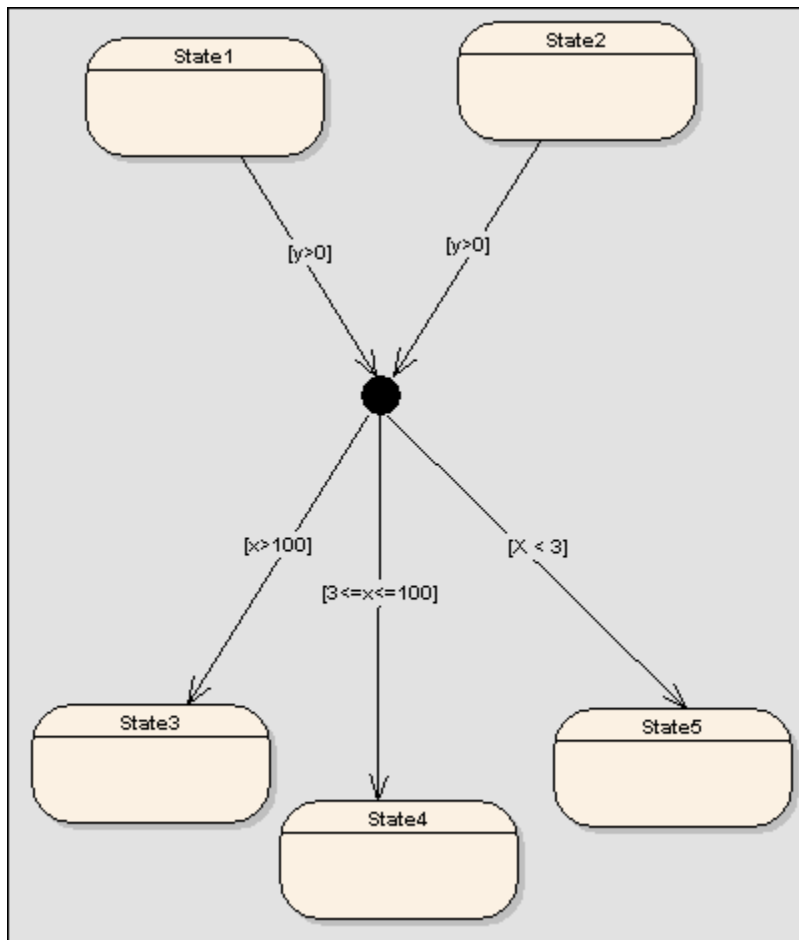
- [Interruptible Region](#)

5.3.3.30 Junction

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

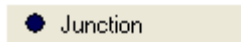


Junction pseudo-states are used to design complex transitional paths. A junction can be used to combine or merge multiple paths into a shared transition path. Alternatively, a junction can split an incoming path into multiple paths, similar to a fork pseudostate. Unlike forks or joins, junctions can apply guards to each incoming or outgoing transition, such that if the guard expression is false, the transition is disabled. The following example illustrates how guards can be applied to transitions coming into or out of a junction pseudo-state.



Common Usage

- [State Machine Diagram](#)



Further Information

- [Pseudo-States](#)

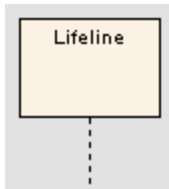
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"junction vertices are semantic-free vertices that are used to chain together multiple transitions. They are used to construct compound transition paths between states. For example, a junction can be used to converge multiple incoming transitions into a single outgoing transition representing a shared transition path (this is known as an merge). Conversely, they can be used to split an incoming transition into multiple outgoing transition segments with different guard conditions. This realizes a static conditional branch. (In the latter case, outgoing transitions whose guard conditions evaluate to false are disabled. A predefined guard denoted "else" may be defined for at most one outgoing transition. This transition is enabled if all the guards labeling the other transitions are false.) Static conditional branches are distinct from dynamic conditional branches that are realized by choice vertices."

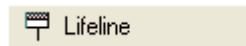
5.3.3.31 Lifeline

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *lifeline* is an individual participant in an interaction (ie. lifelines cannot have multiplicity). A lifeline represents a distinct connectable element. To specify that representation within Enterprise Architect, right-click the lifeline and select *Advanced Settings* | *Set Instance Classifier*. A dialog will appear containing a selectable list of all project classifiers. Lifelines are available in Sequence and Timing diagrams, and although the representation differs between the two, the meaning of a lifeline is the same.

Common Usage



- [Sequence Diagrams](#)
- [Timing Diagrams](#)

Further Information

- [State Lifeline](#)
- [Value Lifeline](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p.427) states:

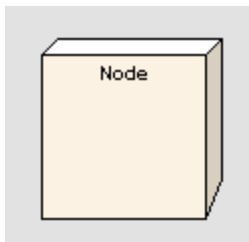
"A lifeline represents an individual participant in the Interaction. While Parts and StructuralFeatures may have multiplicity greater than 1, Lifelines represent only one interacting entity.

"Lifeline is a specialization of NamedElement.

"If the referenced ConnectableElement is multivalued (i.e. has a multiplicity > 1), then the Lifeline may have an expression (the 'selector') that specifies which particular part is represented by this Lifeline. If the selector is omitted this means that an arbitrary representative of the multivalued ConnectableElement is chosen."

5.3.3.32 Node

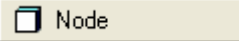
[Common Usage](#) | [OMG UML Specification](#)



A *node* is a physical piece of equipment on which the system will be deployed - for example a workgroup server or workstation. A node usually hosts components and other executable pieces of code, which again may be linked to particular processes or execution spaces. Typical nodes are client workstations, application servers, mainframes, routers, terminal servers, etc.

Nodes are used in deployment diagrams to model the deployment of a system, and to illustrate the physical allocation of implemented artifacts.

Common Usage



- [Deployment Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 195) states:

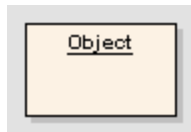
"In the metamodel, a Node is a subclass of Class. It is associated with a Deployment of an Artifact. It is also associated with a set of Elements that are deployed on it. This is a derived association in that these PackageableElements are involved in a Manifestation of an Artifact that is deployed on the Node. Nodes may have an internal structure defined in terms of parts and connectors associated with them for advanced modeling applications."

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) also states:

"A classifier that represents a run-time computational resource, which generally has at least memory and often processing capability. Run-time objects and components may reside on nodes."

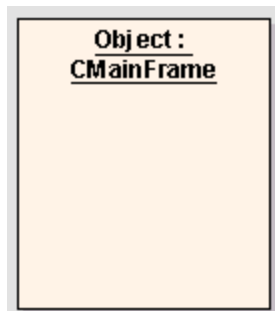
5.3.3.33 Object

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



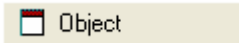
An *object* is an *instance* of a class at run time. For example the car with the license plate "AAA-001" is an instance of the general class of cars (with a license plate number attribute). Objects are often used in analysis to represent the numerous artifacts and items that exist in any business - pieces of paper, faxes, information, etc.

Early in analysis, objects can be used to quickly capture all the things that are of relevance within the system domain. As the model progresses these analysis objects are refined into generic classes from which instances may be derived to represent common business items. Once classes are defined, *objects* may be typed - that is they may have a classifier set that indicates their base type - see [Set Instance Classifier](#).



Common Usage

- [Object Diagram](#)
- [Composite Structure Diagram](#)
- [Communication Diagram](#)



Further Information

- [Setting the Instance Classifier](#)
- [Setting the Run-time State](#)

OMG UML Specification

The OMG UML specification states:

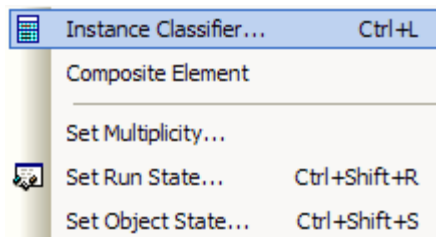
"An object represents a particular instance of a class. It has identity and attribute values. A similar notation also represents a role within a collaboration because roles have instance-like characteristics."

5.3.3.33.1 Instance Classifier

[Further Information](#)

To set the object base type or 'classifier', follow these steps:

1. Select an object in the diagram view.
2. Right click to view the context menu.
3. Select the *Advanced Settings* | *Set Instance Classifier* menu item.



4. From the list of available classes, select the required type.

Further Information

- [Object](#)

5.3.3.33.2 Run-time State

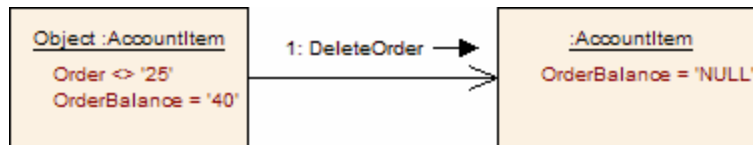
[Further Information](#)

At run-time, an object instance can have specific values for its attributes, or exist in a particular state. To

model the varying behavior of objects at run-time, utilize instance values and run-time states.

Typically there is interest in the run-time behavior of objects that already have a classifier set. You can select from the classifier's attribute list and apply specific values for your object instance. If the classifier has a child state machine, its states will propagate to a list, where the run-time state for the object can be defined. To do this, refer to the topics below.

The following example defines run-time values for the listed variables, which are attributes of the instances' classifier 'AccountItem'.



Further Information

- [Object](#)
- [Define an Instance Variable](#)
- [Remove a Defined Variable](#)
- [Define a Run-time State](#)

5.3.3.33.2.1 Define a Run-time Variable

[Further Information](#)

To add new instance variables to your object by using the *Define Run State* dialog, follow the steps below:

1. *Right-click* on the object to bring up the object context menu.
2. If Instance Variables are supported you can select the *Advanced Settings | Set Run State* option (or press *Ctrl+Shift+R*). The *Run State* dialog will appear.

3. Select the variable from the *Variable* drop down list -OR- type in the name of a new variable.
4. Set the *Operator*, the *Value* and optionally enter some *Notes*.
5. Press *OK* to save.

Further Information

- [Runtime State](#)
- [Remove a Defined Variable](#)

- [Define a Run-time State](#)

5.3.3.33.2 Remove a Defined Variable

[Further Information](#)

To delete a run-time variable:

1. Right click on an object to bring up the object context menu.
2. Select *Set Run State* to bring up the *Run State* dialog.
3. Select the variable to delete from the *Variable* drop down list.
4. Clear the *Value* field.
5. Press *OK*.

Further Information

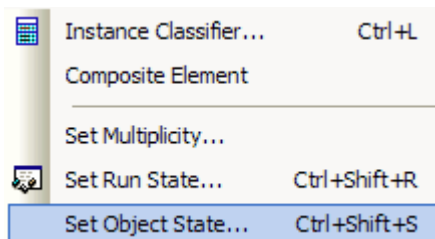
- [Runtime State](#)
- [Define a Run-time Variable](#)
- [Define a Run-time State](#)

5.3.3.33.3 Define a Run-time State

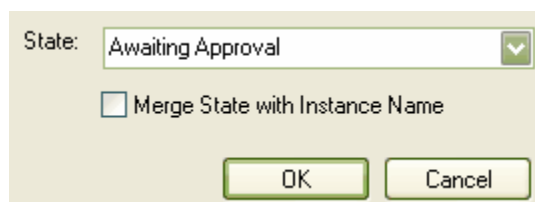
[Further Information](#)

To set the run-time state for a class instance, follow the steps below:

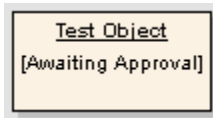
1. Right click on an object and select the *Advanced | Set Object State* menu item.



2. The *Set Object Runtime State* dialog is displayed. Insert the required state - eg. 'Awaiting Approval'. If the associated classifier has a child state machine, those states will propagate into this drop-down list.



- Press **OK** to apply the state. The object now shows the run-time state in square brackets below the object name (see below).

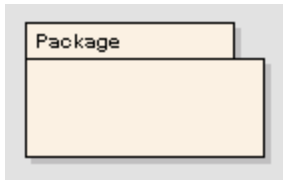


Further Information

- [Runtime State](#)
- [Define a Run-time Variable](#)
- [Remove a Defined Variable](#)

5.3.3.34 Package

[Common Usage](#) | [OMG UML Specification](#)



A *package* is a namespace as well as an element that can be contained in other package's namespaces. A package can own or merge with other packages, and its elements can be imported into a package's namespace. In addition to using packages in the Project Browser to organize your project contents, these packages can also be dragged onto a diagram workspace for structural or relational depictions, including package imports or merges.

Common Usage



- [Package Diagrams](#)
- [Class Diagrams](#)

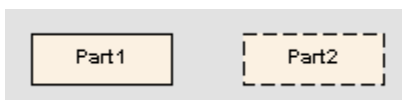
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 99) states:

"A package is a namespace for its members, and may contain other packages. Only packageable elements can be owned members of a package. By virtue of being a namespace, a package can import either individual members of other packages, or all the members of other packages. In addition a package can be merged with other packages."

5.3.3.35 Part

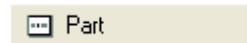
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Parts are run-time instances of classes or interfaces. Multiplicity can be specified for a part, using the notation [x{...y}], where x specifies the initial or set amount of instances when the composite structure is created, and y indicates the maximum amount of instances at any time. Parts are used to express composite structures, or modeling patterns that can be invoked by various objects to accomplish a specific purpose. When illustrating the composition of structures, parts can be embedded as properties of other parts. When embedded as [properties](#), parts can be bordered by a solid outline, indicating the surrounding part owns the part by composition. Alternatively, a dashed outline indicates that the property is referenced and used by the surrounding part, but not composed within it.

Common Usage

- [Composite Structure Diagram](#)



Further Information

- [Properties](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

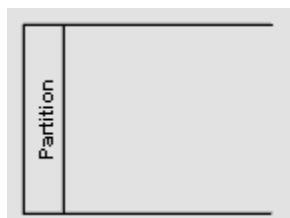
"An element representing a set of instances that are owned by a containing classifier instance or role of a classifier. (See role.) Parts may be joined by attached connectors and specify configurations of linked instances to be created within an instance of the containing classifier."

The OMG UML specification (*UML 2.0 Superstructure*, p. 14) also states:

A role is "the named set of features defined over a collection of entities participating in a particular context. Collaboration: The named set of behaviors possessed by a class or part participating in a particular context. Part: a subset of a particular class which exhibits a subset of features possessed by the class Associations: A synonym for association end often referring to a subset of classifier instances that are participating in the association."

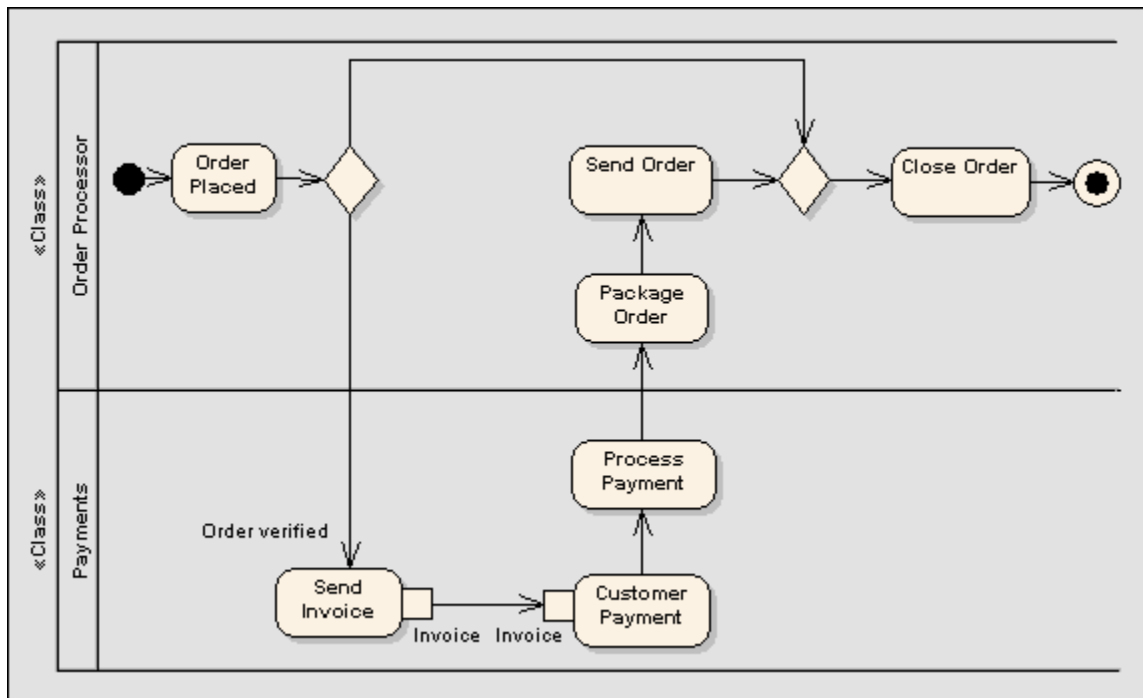

5.3.3.36 Partition

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Activity *partitions* are used to logically organize an Activity diagram. They do not affect the token flow of an Activity diagram, but help structure the view or parts of it.

This example depicts the partitioning between the classes Payments and Order Processor.

**Common Usage**
 Partition

- [Activity Diagram](#)

Further Information

- [Activities](#)
- [Activity Partition](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

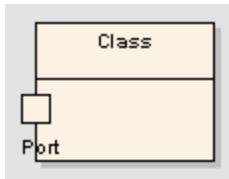
"A grouping of any set of model elements based on a set of criteria. 1. activity diagram: A grouping of activity nodes and edges. Partitions divide the nodes and edges to constrain and show a view of the contained nodes. Partitions can share contents. They often correspond to organizational units in a business model. They may be used to allocate characteristics or resources among the nodes of an activity."

The OMG UML specification (*UML 2.0 Superstructure*, p. 307) also states:

"An activity partition is a kind of activity group for identifying actions that have some characteristic in common."

5.3.3.37 Port

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Ports define the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted by using the expose interface toolbox element. Any connector to a port must provide the required interface, if defined. Ports can appear on either a contained part, a class, or on the boundary of a composite structure.

A Port is a "typed" structural feature or property of its containing classifier.

Common Usage



- [Class Diagrams](#)
- [Object Diagrams](#)
- [Composite Structure Diagrams](#)

Further Information

- [Adding a Port to an Element](#)
- [Managing Inherited and Redefined Ports](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 167) states:

"A port is a structural feature of a classifier that specifies a distinct interaction point between that classifier and its environment or between the (behavior of the) classifier and its internal parts. Ports are connected to properties of the classifier by connectors through which requests can be made to invoke the behavioral features of a classifier. A Port may specify the services a classifier provides (offers) to its environment as well as the services that a classifier expects (requires) of its environment."

5.3.3.37.1 Adding a Port to an Element

[Further Information](#)

You can add a new Port to an element by:

1. Clicking on the Port symbol in Composite section of the UML toolbox and then dragging to or re-clicking on the target host element. This creates an untyped, simple port on the boundary, near where the mouse cursor was pointing.
2. Use the context menu of a suitable Class, Part or Composite object to *Insert Embedded Element* - select Port and a new Port is added at the mouse cursor position.
3. Drag a suitable classifier from the *Project View* onto a Class or Part. EA will prompt to add a typed Port or Part at the mouse cursor position. The new Port will be typed by the original dragged classifier.
4. Use the *Embedded Elements* dialog to add a new Port to the currently selected element.

Further Information

- [Port](#)
- [Managing Inherited and Redefined Ports](#)

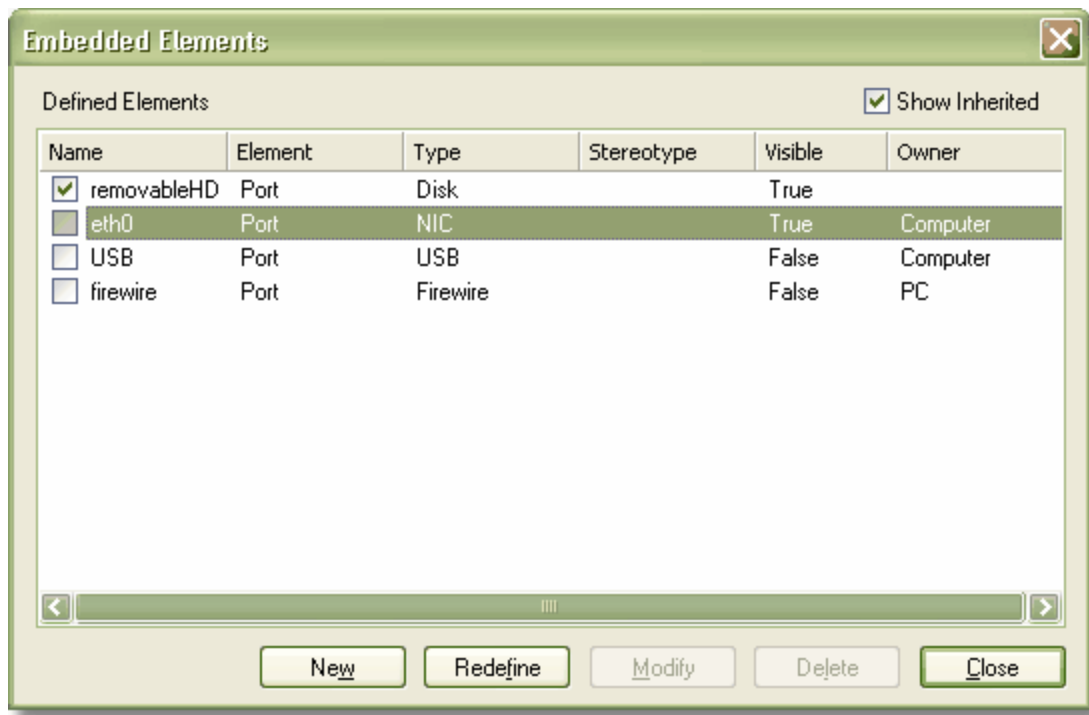
5.3.3.37.2 Managing Inherited and Redefined Ports**Further Information**

A Port is a **redefinable** and **re-useable** property of a composite classifier. So, like attributes, any class will inherit zero or more Ports from its parent and realized interfaces. If you have an inheritance hierarchy with Ports defined in the parent classes, when you open the *Embedded Elements* dialog, you will see the inherited Ports and their named owners.

It is possible to expose for design purposes, an inherited Port (ie. the child class is re-using the parent Port). In this case, EA will create a clone of the re-used Port and mark it as read-only in the child class. This is convenient for modeling Port interactions in child classes where the Ports are defined in the Parents.

It is also possible to redefine a Port in a child class, such that the name is the same, but that the child is a modifiable clone of the original. This is useful where a child class will place additional restrictions or behavior on the Port. The *Embedded Elements* dialog allows you to highlight an inherited Port and mark it as "redefined", this create a new Port on the Child class which is editable, but still logically related to the initial Port.

The *Embedded Elements* diagram below illustrates Port inheritance. The Port "removableHD" is owned by the Child class. The Ports "eth0" and "USB" are owned by the "Computer" class. The "firewire" Port has been added to PC. If any of the inherited Ports are made visible, they are considered re-use Ports and will appear on the child in read-only format. By using the Redefine function, the inherited Port can be copied down and made writeable.



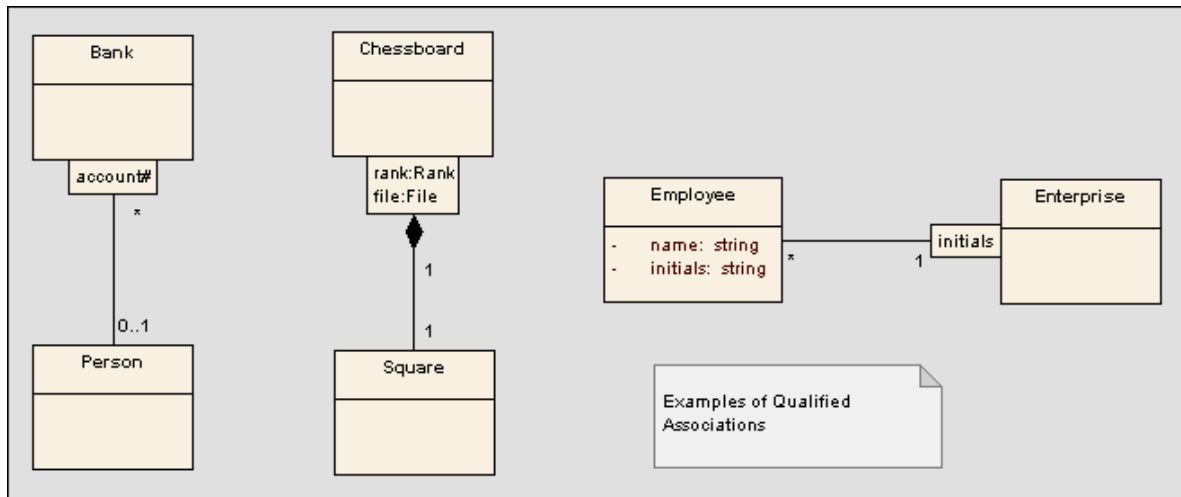
Further Information

- [Port](#)
- [Adding a Port to an Element](#)

5.3.3.38 Qualifiers[OMG UML Specification](#)

A *qualifier* is a property of an association which limits the nature of the relationship between two classifiers or objects.

Some examples of qualified associations are given in this diagram:



Qualifiers are set in the *Source Role* and *Target Role* tabs of the *Association Properties* dialog.

Note: Separate multiple qualifiers with a semi-colon - each qualifier will then appear on a separate line. For example in the diagram above, the qualifier "rank:Rank;file:File" has been rendered in two lines, with a line break at the ; character.

Note: You can enable/disable qualifier rectangles in the *Tools | Options | Diagram* page. If disabled, the old style text qualifiers are used. It is not recommended that you disable qualifiers as they are an integral part of the UML.

Note: You can enable or disable a mild shading on the qualifier rectangles in the *Tools | Options | Diagram* page.

| | |
|--|---|
| <input type="checkbox"/> Disable fully scoped object names | <input checked="" type="checkbox"/> Strict UML Syntax |
| <input type="checkbox"/> Allow change of Created Date | <input type="checkbox"/> Suppress Qualifier boxes |
| <input checked="" type="checkbox"/> Shade Qualifier boxes | <input type="checkbox"/> Suppress Link Constraints |
| <input checked="" type="checkbox"/> Zoom to best scale | |
| <input type="checkbox"/> Show Override Operation dialog on new connector | |

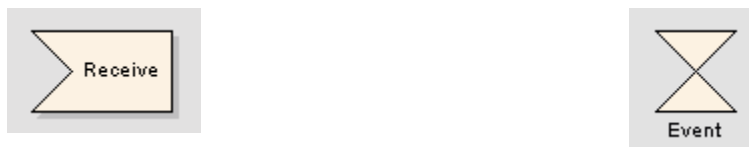
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 93) states:

"A qualifier declares a partition of the set of associated instances with respect to an instance at the qualified end (the qualified instance is at the end to which the qualifier is attached). A qualifier instance comprises one value for each qualifier attribute. Given a qualified object and a qualifier instance, the number of objects at the other end of the association is constrained by the declared multiplicity. In the common case in which the multiplicity is 0..1, the qualifier value is unique with respect to the qualified object, and designates at most one associated object. In the general case of multiplicity 0..*, the set of associated instances is partitioned into subsets, each selected by a given qualifier instance. In the case of multiplicity 1 or 0..1, the qualifier has both semantic and implementation consequences. In the case of multiplicity 0..*, it has no real semantic consequences but suggests an implementation that facilitates easy access of sets of associated instances linked by a given qualifier value."

5.3.3.39 Receive

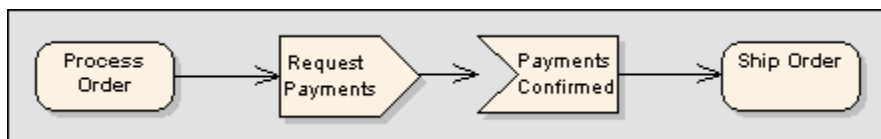
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A receive element is used to define the acceptance or receipt of a request. Movement from a receive element occurs only once receipt is fulfilled according to its specification. The receive element comes in two forms:

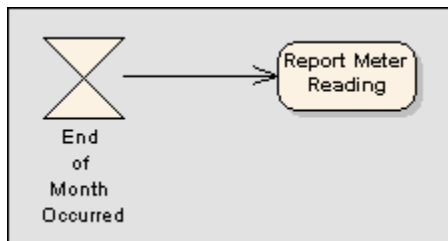
- Accept event action element
- Accept time event action element

The following example reflects a payment process on an order. Upon receiving the payment, the payment is confirmed and the flow continues to ship the order.



Refer to figure 158 (UML 2.0 Superstructure, p. 218).

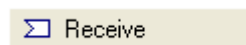
To depict an accept time event, use the standard receive element from the toolbox. Right-click this element, and select *Accept Time Event*. The following example shows the hourglass-shaped accept time event action:



Refer to figure 159 (UML 2.0 Superstructure, p. 219).

Common Usage

- [Activity Diagrams](#)



Further Information

- [Send](#)
- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 218) states:

"AcceptEventAction is an action that waits for the occurrence of an event meeting specified conditions."

5.3.3.40 Region

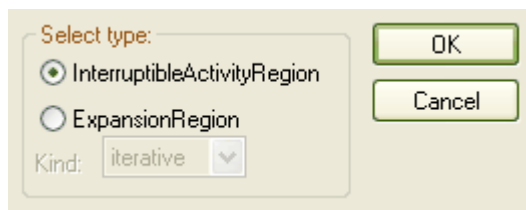
[Common Usage](#) | [Further Information](#)



There are two types of *regions* supported. These are as follows:

- [Expansion region](#)
- [Interruptible activity region](#)

After adding a region element to an activity diagram, the following prompt appears, from where the region type is selected.



Common Usage



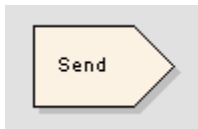
- [Activity Diagram](#)

Further Information

- [Expansion Region](#)
- [Interruptible Activity Region](#)

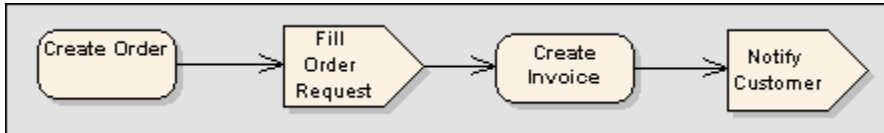
5.3.3.41 Send

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



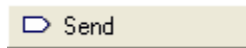
The *send* element is used to depict the action of sending a signal.

The following example shows an order being processed, where a signal is sent to fill the order processed, and, upon creation of that order, a notification is sent to the customer.



Refer to figure 158 (UML 2.0 Superstructure, p. 218).

Common Usage



- [Activity Diagrams](#)

Further Information

- [Receive](#)
- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 255) states:

"SendObjectAction is an action that transmits an object to the target object, where it may invoke behavior such as the firing of state machine transitions or the execution of an activity. The value of the object is available to the execution of invoked behaviors. The requestor continues execution immediately. Any reply message is ignored and is not transmitted to the requestor."


5.3.3.42 State

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *state* represents a situation where some invariant condition holds; this condition can be static, ie. waiting for an event, or dynamic, ie. performing a set of activities. State modeling is usually related to classes, and describes the allowable states a class or element may be in and the transitions that allow the element to move there. There are three types of states: simple states, composite states and submachine states.

Furthermore, there are pseudo-states, resembling some aspect of a state, but with a pre-defined implication. Pseudo-states are used to model complex transitional paths, and classify common state machine behavior.

Common Usage State

- [State Machine Diagram](#)

Further Information

- [Composite State](#)
- [Submachine State](#)
- [Pseudo-States](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 477) states:

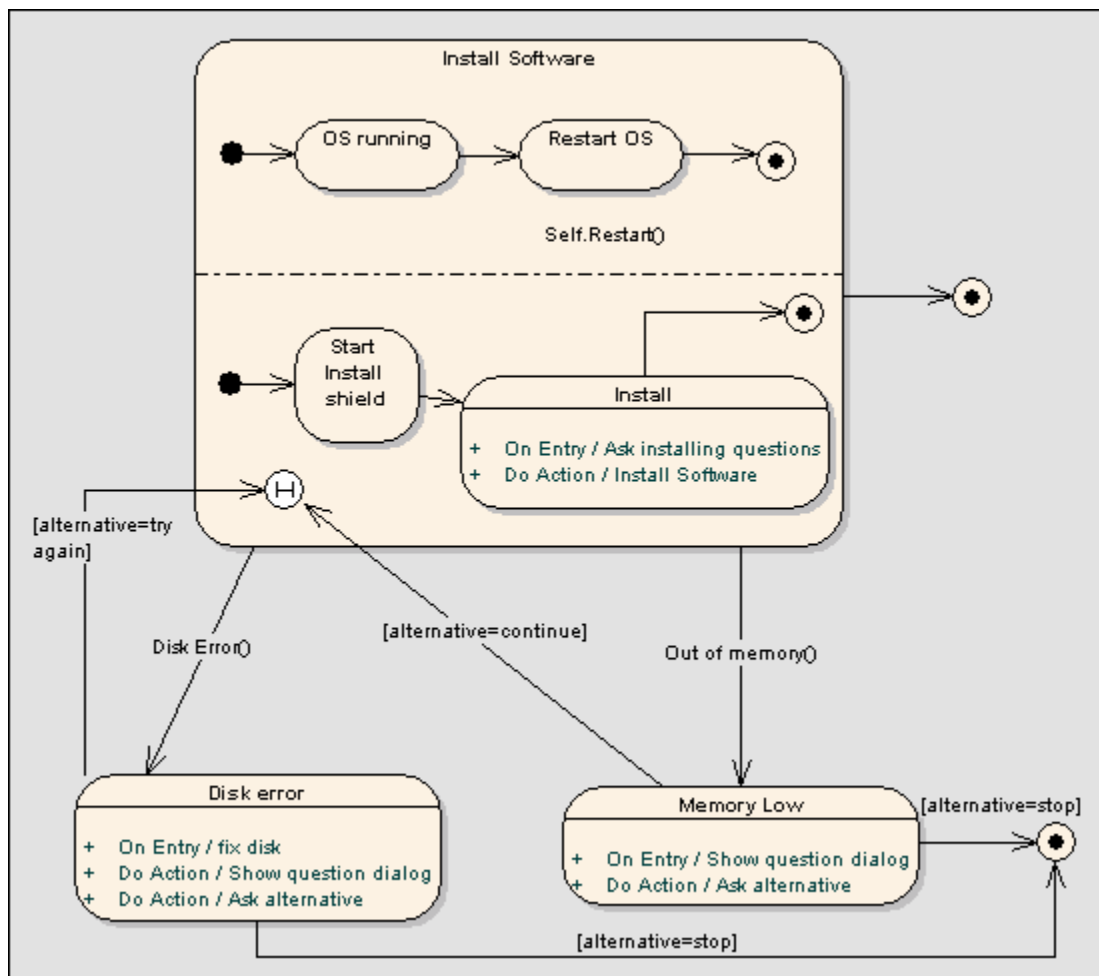
"A state models a situation during which some (usually implicit) invariant condition holds. The invariant may represent a static situation such as an object waiting for some external event to occur. However, it can also model dynamic conditions such as the process of performing some activity (i.e., the model element under consideration enters the state when the activity commences and leaves it as soon as the activity is completed)."

5.3.3.42.1 Composite State

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

Composite states are semantically equivalent to submachine states, but they cannot be reused as can submachine states. They are composed inline the state machine, by expanding a state, adding regions if applicable, and designing the appropriate composite state with internal states referred to as sub-states.

Composite states can be orthogonal, if regions are created. If a composite state is orthogonal, its entry denotes that a single sub-state is concurrently active in all regions. The hierarchical nesting of composite states, coupled with region usage, generates a situation of multiple states concurrently active; this situation is referred to as the active state configuration.



Common Usage

- [State Machine Diagram](#)

Further Information

- [Regions](#)
- [Submachine State](#)
- [State](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 478) states:

"A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions. A given state may only be decomposed in one of these two ways.

"Any state enclosed within a region of a composite state is called a substate of that composite state. It is called a direct substate when it is not contained by any other state; otherwise it is referred to as an indirect substate.

"Each region of a composite state may have an initial pseudostate and a final state. A transition to the enclosing state represents a transition to the initial pseudostate in each region. A newly-created object takes its topmost default transitions, originating from the topmost initial pseudostates of each region.

"A transition to a final state represents the completion of activity in the enclosing region. Completion of activity in all orthogonal regions represents completion of activity by the enclosing state and triggers a completion event on the enclosing state. Completion of the topmost regions of an object corresponds to its termination."

5.3.3.43 State Lifeline

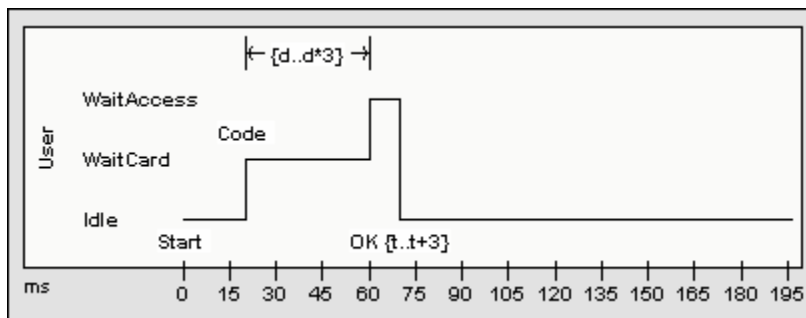
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *lifeline* is the path an object takes across a measure of time, as indicated by the x-axis.

A *state lifeline* follows discrete transitions between states, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations.

An example of a state lifeline is given in this diagram:



Refer to figure 350 (UML 2.0 Superstructure, p. 452).

A state life line consists of a set of transition points. Each transition point can be defined with the following properties:

| | |
|-----------------------|--|
| At time | Specifies the starting time for a change of state. |
| Transition to | Indicates the state to which the lifeline will change. |
| Event | Describes the occurring event. |
| Timing constraints | Refers to the time taken for a state to change within a lifeline, or the time taken to transmit a message (ie. $t..t+3$). |
| Timing observations | Provides information on the time of a state change or sent message. |
| Duration constraints | Pertains to a lifeline's period at a particular state. The constraint could be instigated by a change of state within a lifeline, or that lifeline's receipt of a message. |
| Duration observations | Indicates the interval of a lifeline at a particular state, begun from a change in state or message receipt. |

An example of this from the diagram above would be that the **OK** transition point has the properties:

| | |
|-----------------------|--------|
| At Time | 70ms |
| Transition to | 200ms |
| Event | OK |
| Timing constraints | t..t+3 |
| Timing observations | – |
| Duration constraints | – |
| Duration observations | – |

Common Usage
 State Lifeline

- [Timing Diagram](#)

Further Information

- [Value Lifeline](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 451) states:

"This is the state of the classifier or attribute, or some testable condition, such as an discrete enumerable value. See also 'StateInvariant (from BasicInteractions)' on page 433.

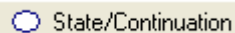
"It is also permissible to let the state-dimension be continuous as well as discrete. This is illustrative for scenarios where certain entities undergo continuous state changes, such as temperature or density."

5.3.3.44 State/Continuation

[Common Usage](#) | [Further Information](#)



The *state/continuation* symbol serves two different purposes for interaction diagrams, as state invariants and continuations.

Common Usage
 State/Continuation

- [Sequence Diagrams](#)

Further Information

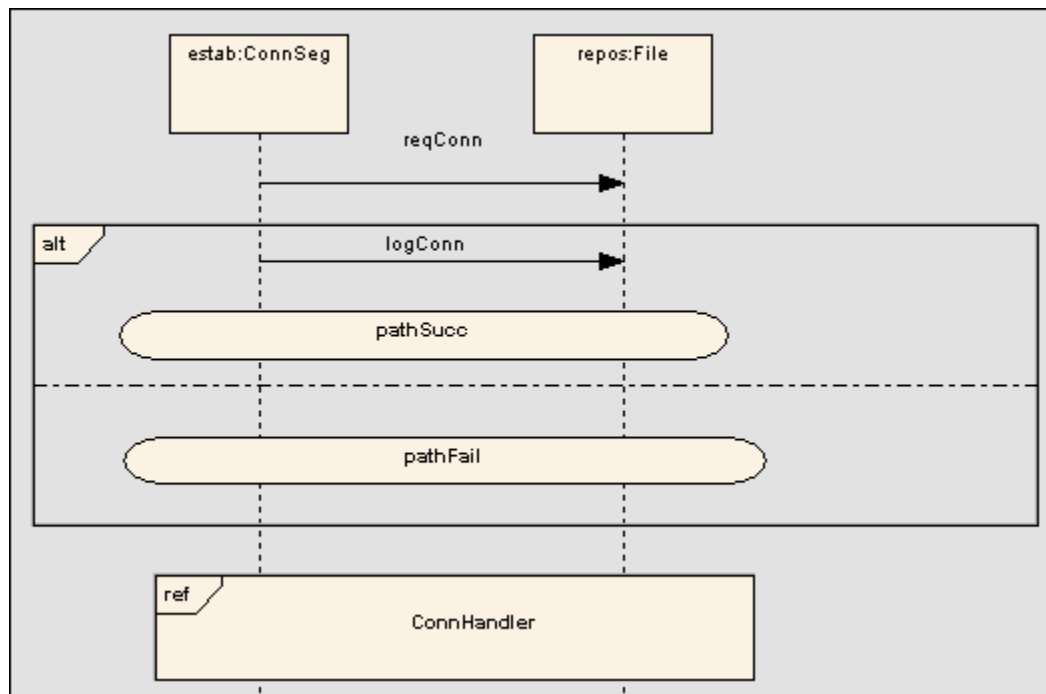
- [State Invariants](#)
- [Continuations](#)

5.3.3.44.1 Continuation

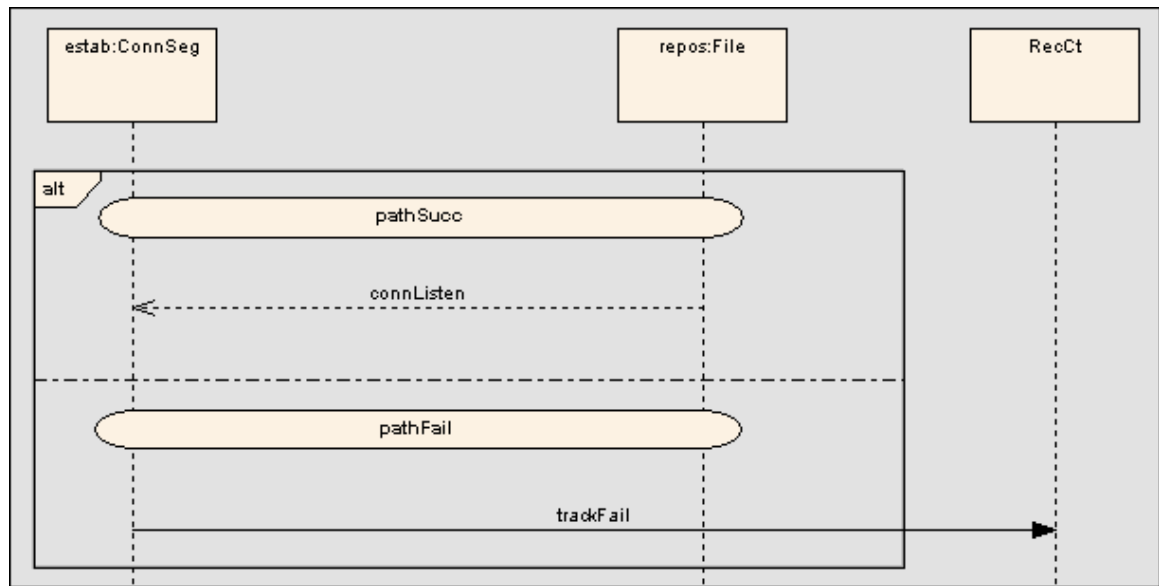
[Further Information](#) | [OMG UML Specification](#)

A *continuation* is:

- Used in seq and alt combined fragments, to indicate the branches of continuation an operand follows.
- To indicate a continuation, end an operand with a continuation, and indicate the continuation branch with matching continuation (same name) preceding the interaction fragment.
- For the following continuation example, an alt combined fragment has continuations, pathSucc and pathFail. These continuations are located within the interaction occurrence ConnHandler, which has subsequent events based on the continuation.



Below is the interaction referenced by the InteractionOccurrence.



Further Information

- [State/Continuation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 414) states:

"A Continuation is a syntactic way to define continuations of different branches of an Alternative CombinedFragment. Continuations is intuitively similar to labels representing intermediate points in a flow of control."

The OMG UML specification (*UML 2.0 Superstructure*, p. 415) states:

"Continuations have semantics only in connection with Alternative CombinedFragments and (weak) sequencing. If an InteractionOperand of an Alternative CombinedFragment ends in a Continuation with name (say) X, only InteractionFragments starting with the Continuation X (or no continuation at all) can be appended."

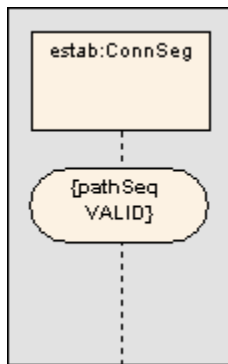
5.3.3.44.2 State Invariant

[Further Information](#) | [OMG UML Specification](#)

A state invariant is:

- A condition applied to a lifeline, which must be fulfilled for the lifeline to exist.

A state invariant is exemplified below.



When a State Invariant is moved near to a lifeline, it will snap to the center. If the sequence object is dragged left or right, the State Invariant will move with it.

Further Information

- [State/Continuation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 433) states:

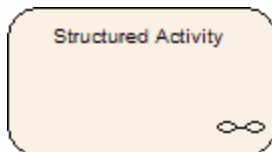
"A *StateInvariant* is a constraint on the state of a *Lifeline*. In this case we mean by "state" also the values of eventual attributes of the *Lifeline*.

"A *StateInvariant* is an *InteractionFragment* and it is placed on a *Lifeline*."

5.3.3.45 Structured Activity

Common Usage

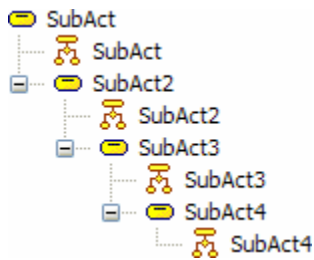

You can add a *structured activity* element to an Activity diagram. A *structured activity* element is a pointer to a child Activity diagram.



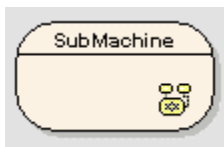
To create a *structured activity*, click on the *Structured Activity* element in the *Activity* group in the toolbox, then click on the diagram. *Structured activity* elements have a small symbol on the bottom right-hand corner.

To access the Activity diagram represented by the *structured activity* element, double click the *structured activity* element and the diagram will open.

When you create nested *structured activity* elements, they are shown as nested in the Project Browser - see the example below.

**Common Usage**
 Structured Activity

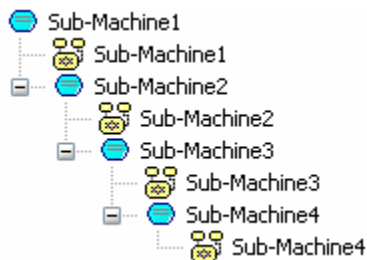

- [Activity Diagrams](#)

5.3.3.46 SubMachine[Common Usage](#)

You can add a *sub-machine* element to a State Machine diagram. A sub-machine element is a pointer to a child State Machine diagram.

To create a sub-machine, click on the sub-machine element in the State group in the toolbox, then click on the diagram. Sub-machine elements have a small yellow diagram on the bottom right-hand corner. To access the State Machine diagram represented by the sub-machine element, double click the sub-machine element and the diagram will open.

When you create nested sub-machine elements, they are shown as nested in the Project Browser - see the example below.

**Common Usage**
 SubMachineState

- [State Machine Diagram](#)

5.3.3.47 *Synch*

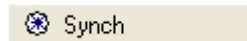
[Common Usage](#) | [OMG UML Specification](#)



A *synch* state is useful for indicating that concurrent paths of a state machine will be synchronized. After bringing the paths to a *synch* state, the emerging transition will indicate unison.

Common Usage

- [State Machine Diagram](#)



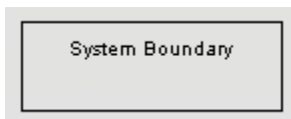
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 16) states:

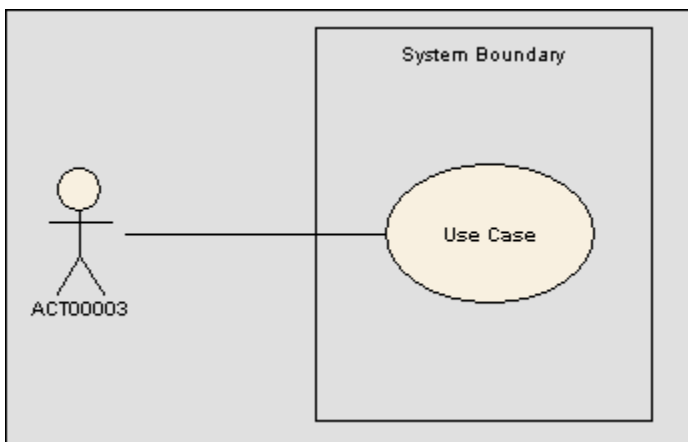
"A vertex in a state machine used for synchronizing the concurrent regions of a state machine."

5.3.3.48 *System Boundary*

[Common Usage](#) | [OMG UML Specification](#)

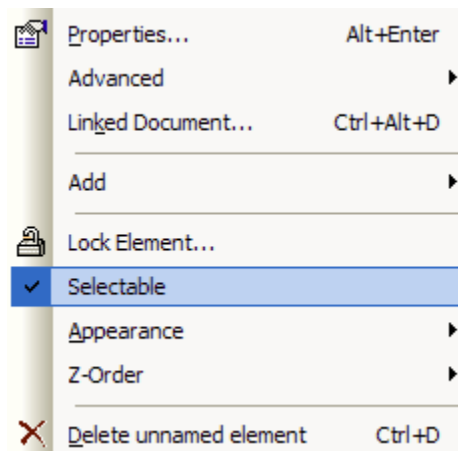


A *system boundary* element signifies a classifier, such as a class, component or sub-system, to which the enclosed use cases are applied. By depicting a boundary, its referenced classifier does not reflect ownership of the embodied use cases, but instead indicates usage.



The following properties of a boundary may be set: the name, the border style, and the number of horizontal or vertical swim lanes.

A boundary element may be marked as 'Selectable' or not. When not selectable, you can click within the boundary space without activating or selecting the boundary itself. This is useful when you have many elements within the boundary and selection of them is being made difficult by the boundary itself.

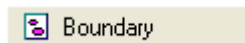


Note: A boundary may have an associated image which it will display instead of its default format ... use the *Select Alternate Image...* item to select an image.

Common

Usage

- [Use Case](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 520) states:

"If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier."

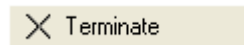
5.3.3.49 Terminate

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



The *terminate* pseudostate indicates that upon entry of its pseudostate, the state machine's execution will end.

Common Usage



- [State Machine Diagram](#)

Further Information

- [Pseudo-States](#)

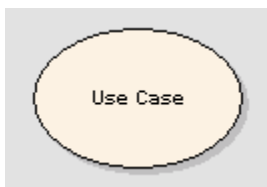
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"Entering a terminate pseudostate implies that the execution of this state machine by means of its context object is terminated."

5.3.3.50 Use Case

[Common Usage](#) | [OMG UML Specification](#)

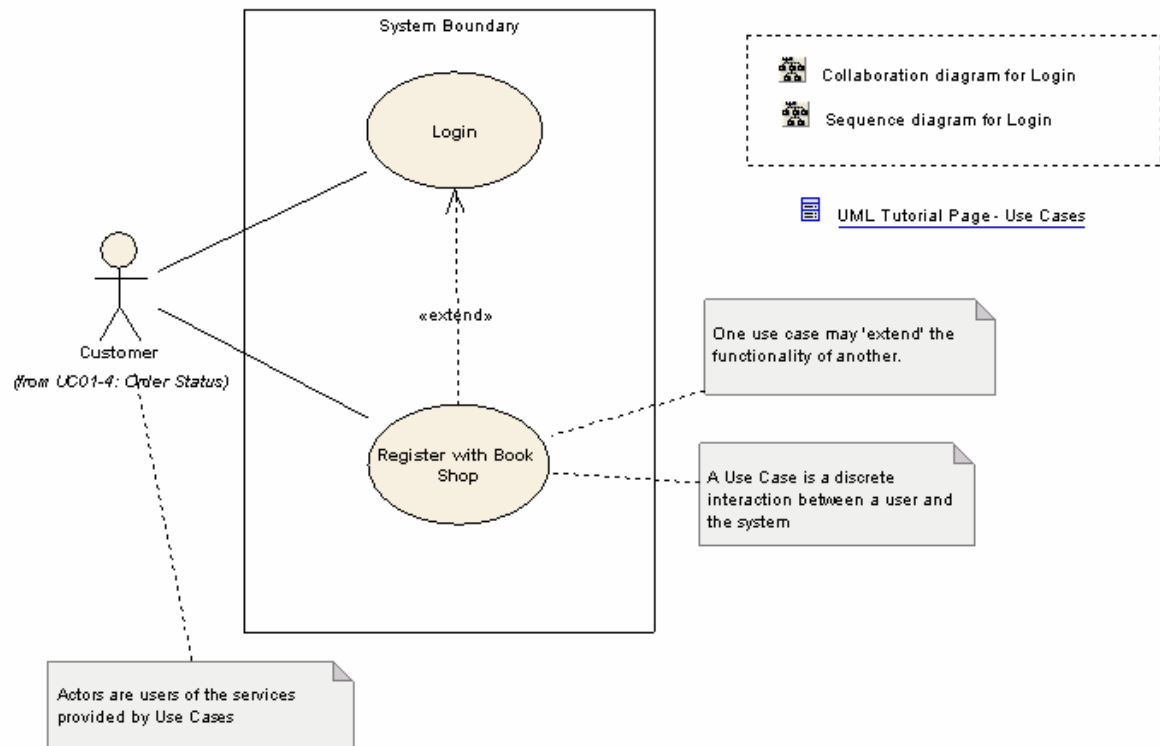


A *use case* is a UML modeling element that describes how a user of the proposed system will interact with the system to perform a discrete unit of work. It describes and signifies a single interaction over time that has meaning for the end user (person, machine or other system), and is required to leave the system in a complete state: either the interaction completed or was rolled back to the initial state.

- A use case typically has requirements and constraints that describe the essential features and rules under which it operates.
- A use case may have an associated Sequence diagram illustrating behavior over time - who does what and to whom, when.
- A use case typically has scenarios associated with it that describe the work flow over time that produces the end result. Alternate work flows (to capture exceptions, etc.) are also allowed.

Tip: Use a Use Case diagram and model to build up the functional requirements and implementation details of the system.

Below is an example Use Case model:



Common Usage

 Use Case

- [Use Case Diagram](#)

Further Information

- [Use Case Extension Points](#)
- [Using Rectangle Notation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

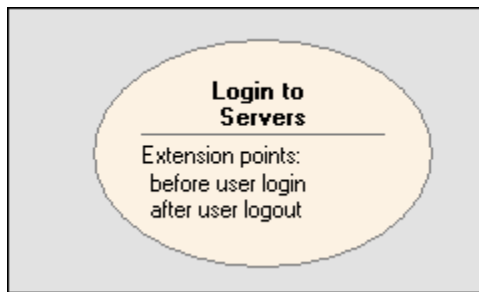
"The specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system."

5.3.3.50.1 Use Case Extension Points

[Further Information](#)

Use *extension points* to specify the point of an extended use case where an extending use case's behavior should be inserted. The specification text can be informal or precise to define the location of the extension point.

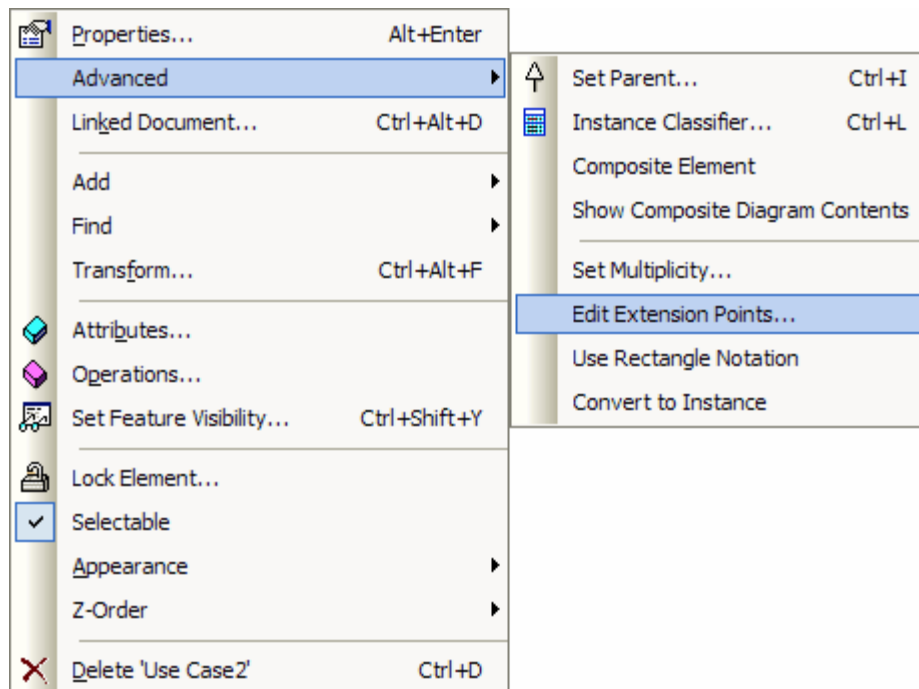
Note that conditions to apply that extending use case and the extension point to use should be attached as a note to the extend relationship.



Extension Points

To work with extension points, follow these steps:

1. Right click on the use case element to view the context menu.
2. Select *Advanced* | *Edit Extension Points...*



3. The *Extension Points dialog* will appear - listing defined points for that use case.
4. Right click in the list to access the create, edit, delete and move functions.

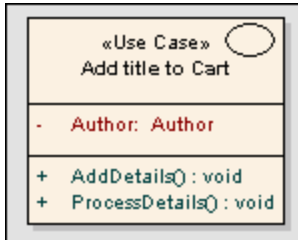
Further Information

- [Use Case](#)

5.3.3.50.2 Using Rectangle Notation

[Further Information](#)

You can display a use case using rectangle notation - this displays the use case in a rectangle, with an oval in the top right corner. Any attributes, operations, constraints, etc. belonging to the use case are shown, in the same style as a class.



To show a use case using rectangle notation, right click on the use case object on the diagram and select *Advanced Settings* | *Use Rectangle Notation*. This setting will only apply to the selected use case, and can be toggled on and off.

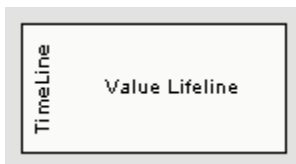
Tip: Actor elements can also be displayed using rectangle notation.

Further Information

- [Use Case](#)

5.3.3.51 Value Lifeline

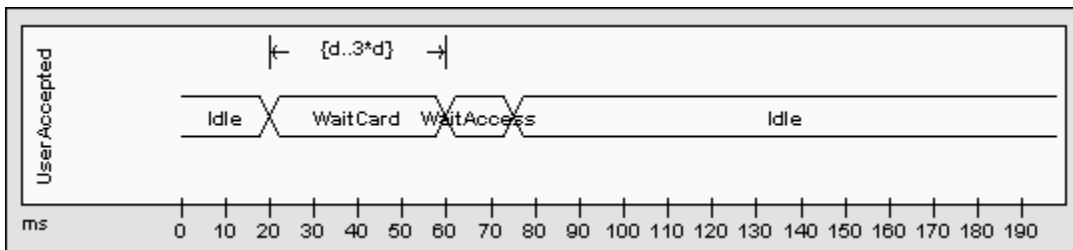
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *lifeline* is the path an object takes across a measure of time, indicated by the x-axis.

The *value lifeline* shows the lifeline's state across the diagram, within parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

An example of a value lifeline is shown in the below diagram:



Refer to figure 351 (UML 2.0 Superstructure, p. 453).

The value lifeline consists of a set of transition points. Each transition point can be defined with the following properties:

| | |
|-----------------------|--|
| At time | Specifies the starting time for a change of state. |
| Transition to | Indicates the state to which the lifeline will change. |
| Event | Describes the occurring event. |
| Timing constraints | Refers to the time taken for a state to change within a lifeline, or the time taken to transmit a message. |
| Timing observations | Provides information on the time of a state change or sent message. |
| Duration constraints | Pertains to a lifeline's period at a particular state. The constraint could be instigated by a change of state within a lifeline, or that lifeline's receipt of a message. |
| Duration observations | Indicates the interval of a lifeline at a particular state, begun from a change in state or message receipt. |

An example of this from the diagram above would be that the "OK" transition point has the properties:

| | |
|-----------------------|----------|
| At Time | 20ms |
| Transition to | 60ms |
| Event | WaitCard |
| Timing constraints | – |
| Timing observations | – |
| Duration constraints | d..3*d |
| Duration observations | – |

Common Usage Value Lifeline

- [Timing Diagram](#)

Further Information

- [State Lifeline](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 451) states:

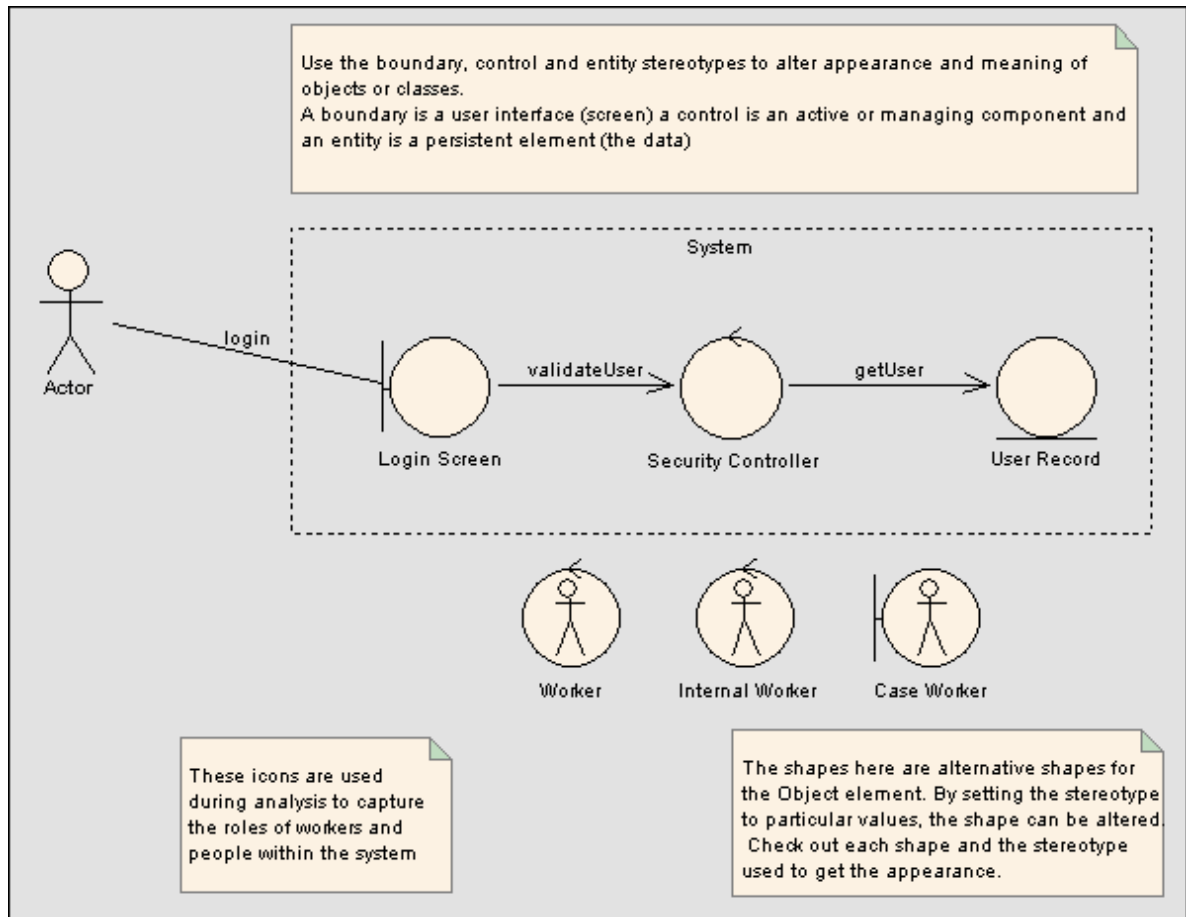
"Shows the value of the connectable element as a function of time. Value is explicitly denoted as text. Crossing reflects the event where the value changed."

5.3.4 Inbuilt and Extended Stereotypes

There are many other elements in UML which you can also work with in Enterprise Architect. This topic gives a brief introduction to some of these elements.

5.3.4.1 Analysis Stereotypes

EA has some built in stereotypes which you can assign to an element during analysis. The effect of these stereotypes is to display a different icon than the normal element icon, providing a visual key to the element purpose. The diagram below illustrates the main types of inbuilt icons for elements:

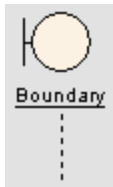


The stereotypes used are:

- [boundary](#) - for a system boundary (eg. a Login screen)
- [control](#) - to specify an element is a controller of some process (as in Model View Controller pattern)
- [entity](#) - the element is a persistent or data element
- [worker](#), [caseworker](#) and [internal worker](#) - denotes specific roles in the analysis based on RUP and Robustness analysis guidelines

5.3.4.2 Boundary

[Common Usage](#)

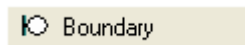


A *boundary* is a stereotyped class that models some system boundary - typically a user interface screen. It is used in the conceptual phase to capture users interacting with the system at a screen level (or some other boundary interface type). It is often used in Sequence and Robustness (Analysis) diagrams. It is the View in the Model-View-Controller pattern.

Tip: Use boundary elements in analysis to capture user interactions, screen flows and element interactions (or 'collaborations').

Common Usage

- [Sequence Diagram](#)
- [Analysis Diagram](#)
- [Robustness Diagram](#)



5.3.4.2.1 Create a Boundary

[Further Information](#)

Using the Toolbox

To create a boundary element, follow the steps below:

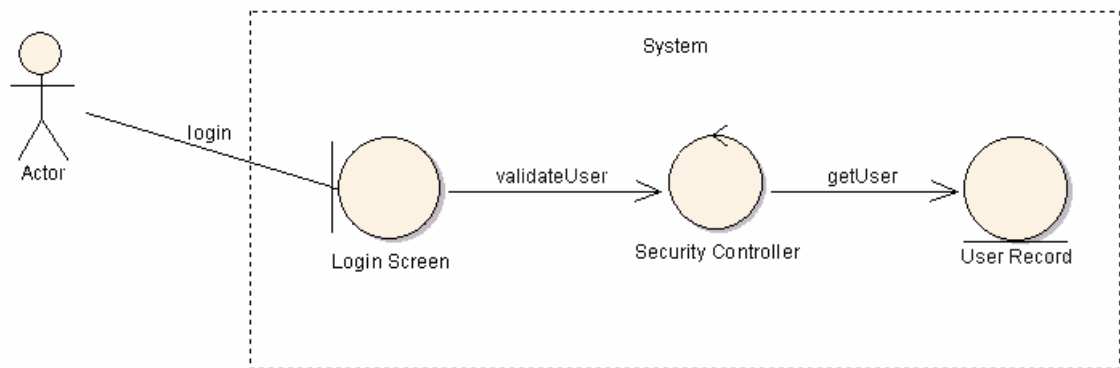
1. Open the *Communication* group from the *Toolbox*.
2. Drag the *Boundary* element onto the diagram

Using the Properties Dialogue

Alternatively, follow the steps below:

1. Insert a new class.
2. Open the *Properties* dialog.
3. Set the *Stereotype* to 'boundary'.
4. Save changes.

The illustration below shows an Actor interacting with a Boundary (in this case, a Login screen).

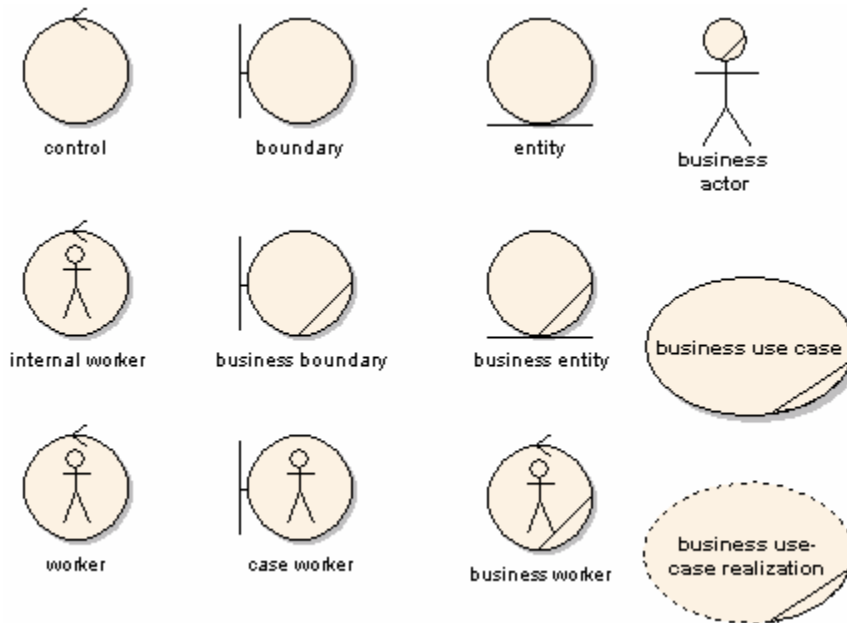


Further Information

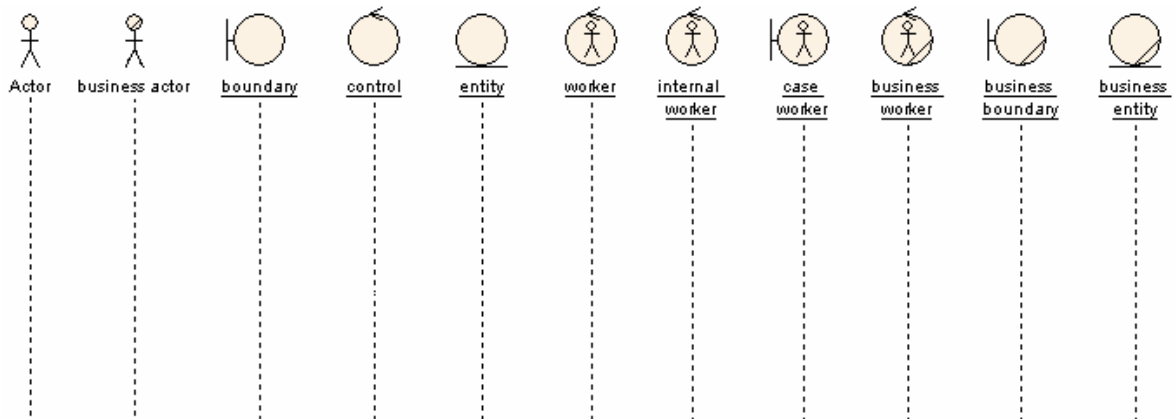
- [Boundary](#)

5.3.4.3 Business Modeling Stereotypes

This diagram shows the range of inbuilt stereotyped icons for business modeling. These include stereotypes for classes and objects - as well as one of actors and two for use cases. The name given to each graphical element in the diagram is that of the stereotype to apply.



This diagram shows the inbuilt stereotype icons for business modeling with sequence diagrams. The name given to each graphical element in the diagram is that of the stereotype to apply.



5.3.4.4 Composite Elements

Enterprise Architect supports *composite elements* for classes, objects, use cases etc. A composite element is a pointer to a child diagram.

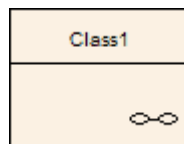
Creating a Composite Element

Composite elements can be set from the element context menu. Follow the instructions below:

1. Create the element you wish to set as a composite element.
2. Right click on the element in the diagram and select *Advanced Settings | Composite Element* from the context menu.

Note: If *Composite Element* does not appear in the element context menu, the option is not available for the type of element you have selected.

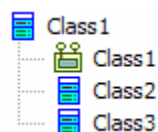
The element will appear as follows:



Note the small icon in the bottom right hand corner representing that this is now a composite element.

3. Double click the composite element to access the child diagram that it points to.

The composite element and its child diagram are represented in the Project Browser as follows:



Note that Class2 and Class3 are elements in the child diagram.

Alternate Notation

Composite Elements may show their contents instead of their usual notation. To enable this notation, *Right-Click* the element to open the context menu, then select *Advanced | Show Composite Diagrams Contents*.

The Automation Interface

Automation support is available for composite elements - Element has an Elements collection and a Diagrams collection.

5.3.4.5 Control

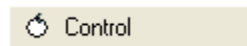
[Common Usage](#)



A *control* is a stereotyped class that represents a controlling entity or manager. A control organizes and schedules other activities and elements. It is the controller of the Model-View-Controller pattern.

Common Usage

- [Sequence Diagram](#)



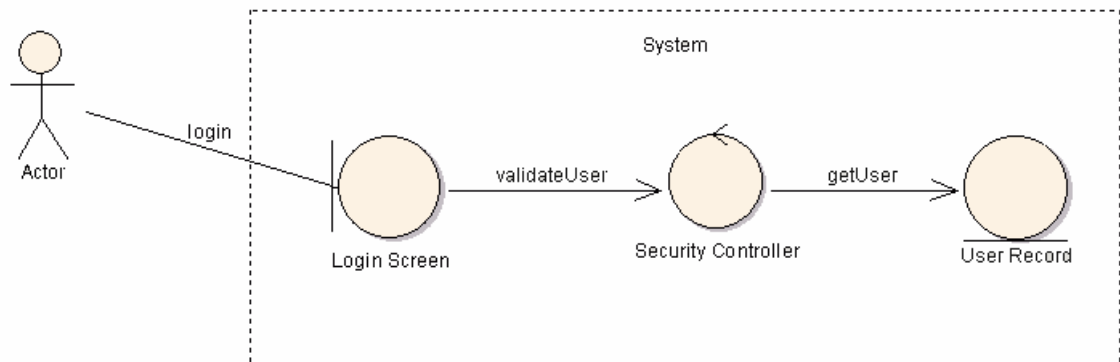
5.3.4.5.1 Create a Control Element

[Further Information](#)

To create a control element, follow the steps below:

1. Insert a new class.
2. Bring up the *Properties* dialog.
3. Set the *Stereotype* to 'control'.
4. Save changes.

The appearance will change to that illustrated in the diagram below (for the security controller element):



Note: The Model-View-Controller (MVC) pattern is a design pattern for building a wide range of applications that have a user interface, business or application logic and persistent data.

Further Information

- [Control](#)

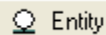
5.3.4.6 Entity

[Common Usage](#)



An *entity* is a store or persistence mechanism that captures the information or knowledge in a system. It is the Model in the Model-View-Controller pattern.

Common Usage



- [Sequence Diagram](#)

5.3.4.6.1 Create an Entity

[Further Information](#)

To create an entity, follow the steps below:

1. Insert a class from the UML toolbox.
2. Open the *Properties* dialog.
3. Set the element *Stereotype* to 'entity'.
4. Save changes.

Further Information

- [Entity](#)

5.3.4.7 Event

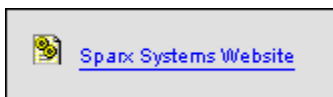
The UML includes two elements that are used to model events. The first element is the *receive event* and is depicted as a rectangle with a recessed 'V' on the left side. This element indicates that an event occurs in the system due to some external or internal stimulus. Typically this will invoke further activities and processing.



The second element is the *send event*. This element models the generation of a stimulus in the system and the passing of that stimulus to other elements within the system or external to the system.

Send and receive events can be added from the Analysis and Activity sections of the UML toolbox.

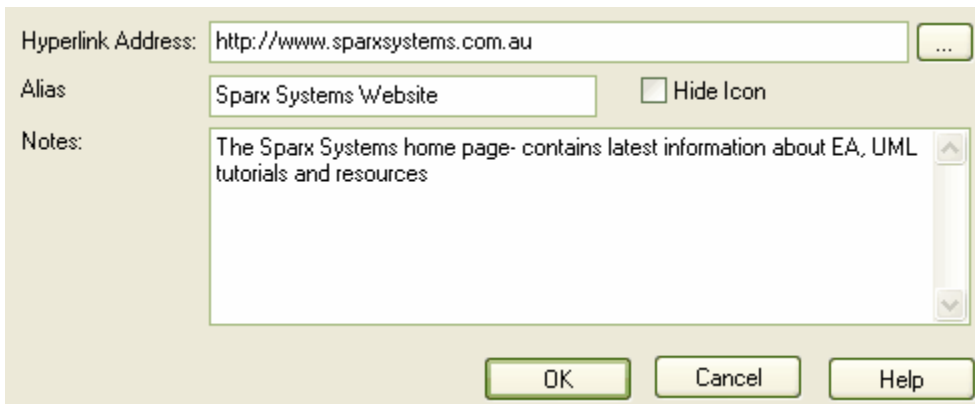
5.3.4.8 Hyperlinks



You can place a *hyperlink* element onto a diagram. This element is a type of text element - but one that may contain a pointer to a file or web address. When double clicked on, EA will attempt to execute the related file or address. You can connect diagrams to associated files, web pages and even other EA model files. You can add a hyperlink using the *Elements* toolbar

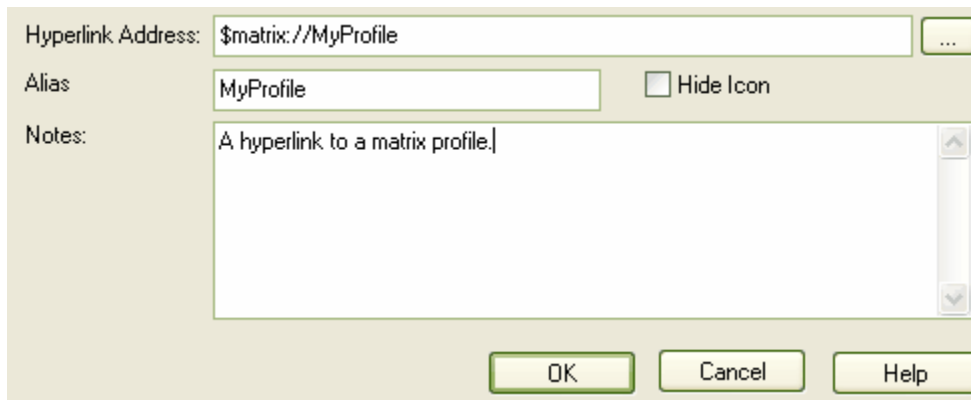


Configuring the hyperlink



Creating Hyperlinks to Matrix Profiles

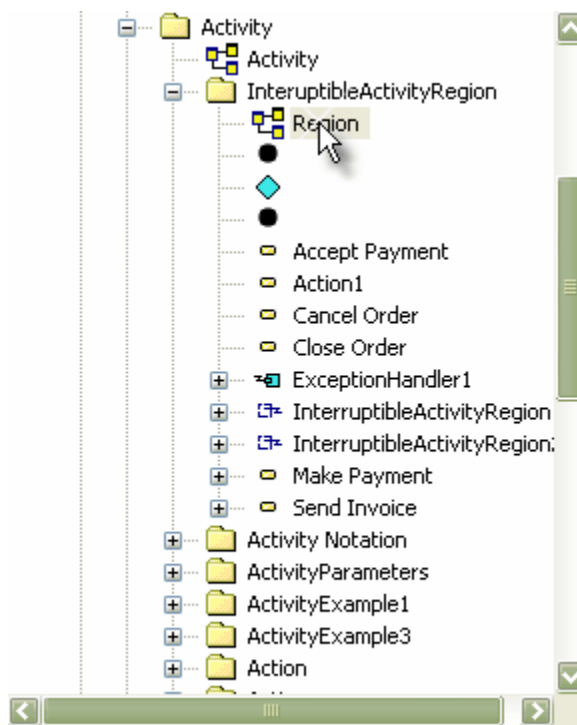
Hyperlinks may also be added with a Matrix Profile as its target from a hyperlink object in a diagram. To achieve this Use \$matrix:// as the target prefix followed by the name of the profile (e.g. \$matrix://MyProfile).



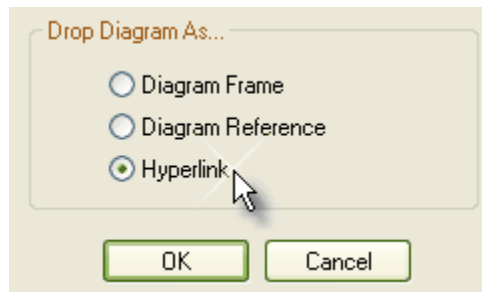
Creating Hyperlinks between diagrams

You can create hyperlinks between diagrams by following the steps detailed below:

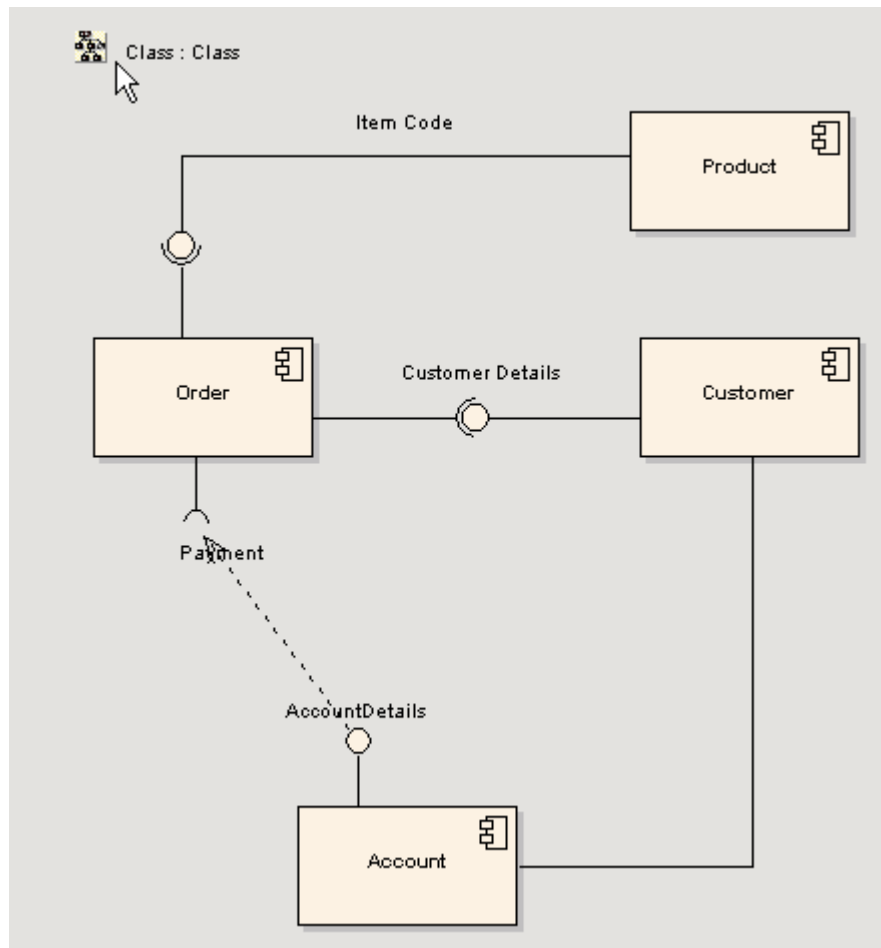
1. Open the diagram that you want to display a hyperlink to another diagram. From the project view select and drag the diagram you want to create a hyperlink.



2. Next, drag the diagram on to the current diagram and select the option *Hyperlink*.

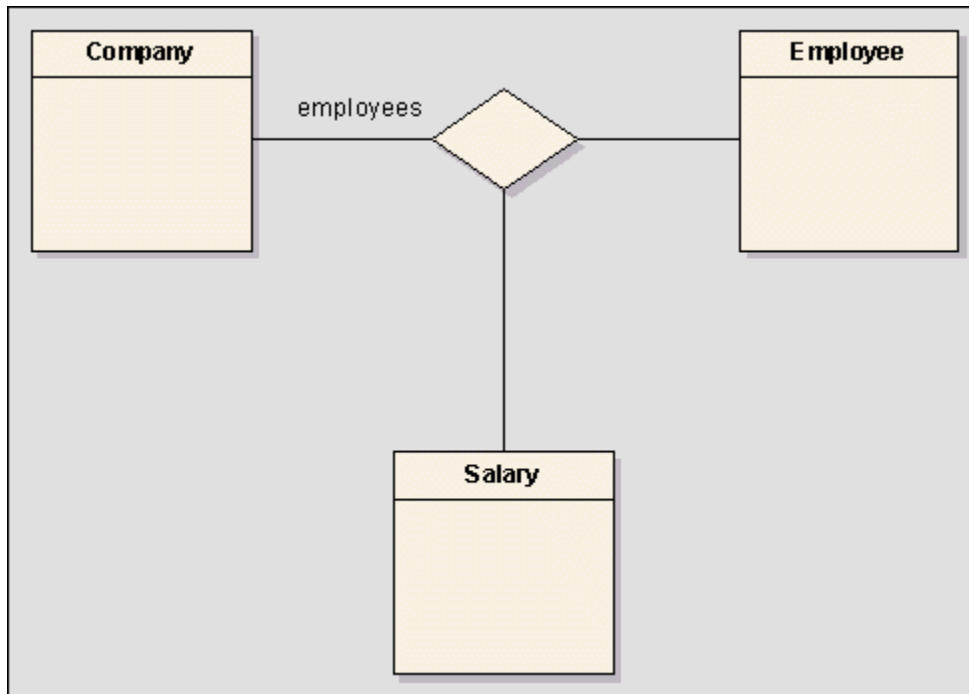


3. The final hyperlinked diagram should look like the diagram below:



Note: If the hyperlinks are appearing as sub activities go to the *Tools | Options | Diagram | Behavior* and uncheck the *Use Automatic SubActivities* check box.

5.3.4.9 N-Ary Association



An *n*-Ary association element is used to model complex relationships between three (or more) elements. It is not a commonly employed device, but can be used to good effect where there is a dependant relationship between more than two elements.

In the example above there is a relationship between a Company and an Employee and a Salary.

5.3.4.10 Other Elements

In addition to the standard UML elements, you can also add notes, boundaries, text, diagram properties and hyperlinks to your diagram.

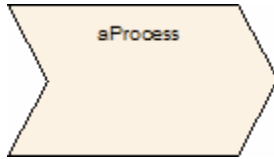
To add these, use the buttons in the [shortcut toolbar](#). The first inserts a system boundary, the second a note and the third a text element.



The dotted line is a note link, and can be used to link notes to particular elements.

These elements are simple comments and delimiters and have no structural role in the model. They are very useful in explaining and grouping other elements.

5.3.4.11 Process

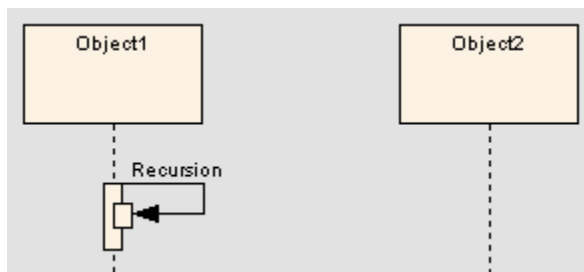


A *process* is a stereotyped activity element which expresses the concept of a business process. Typically this involves inputs, outputs, work flows, goals and connections with other processes.

Business processes typically range across many parts of the organization and span one or more systems.

5.3.4.12 Recursion

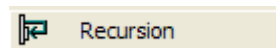
[Common Usage](#) | [Further Information](#)



A *recursion* is a type of message used in sequence diagrams to indicate a recursive function.

Common Usage

- [Sequence Diagram](#)



Further Information

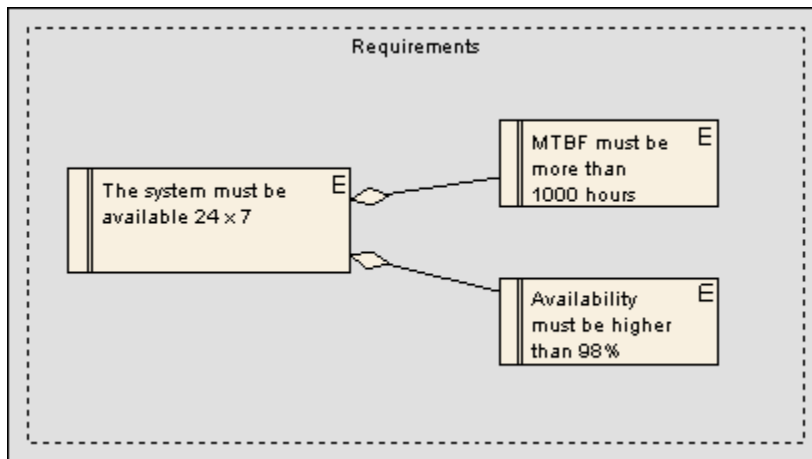
- [Message \(Sequence Diagram\)](#)

5.3.4.13 Requirements

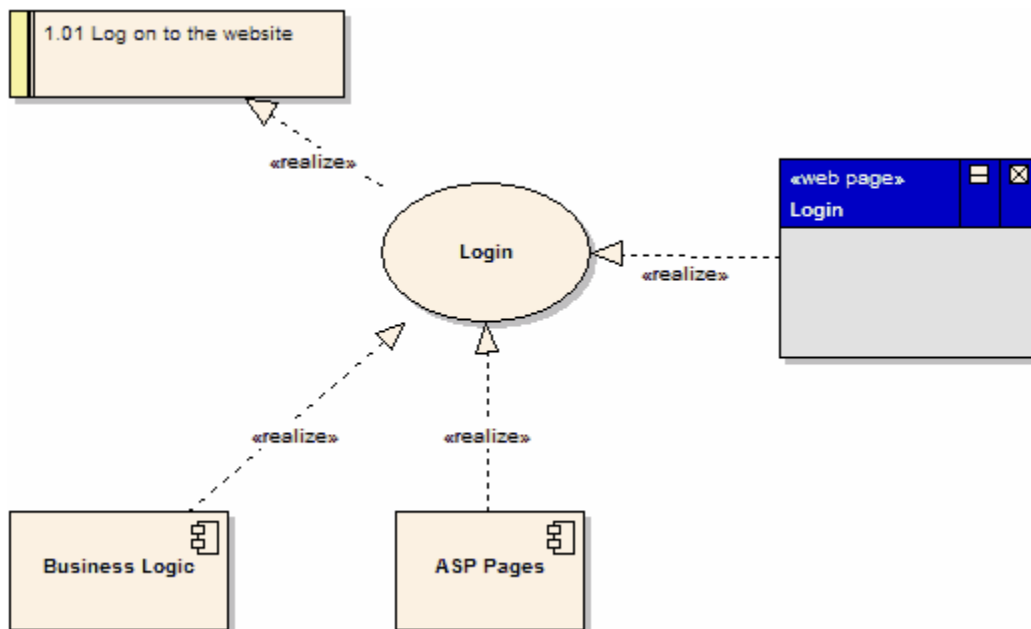
As an analysis step, often it is desirable to capture simple system *requirements*. These will eventually be realized by use cases.

In the initial requirement gathering phase, cataloging requirements may be achieved using the custom requirement extension.

Requirements may also be aggregated to create a hierarchy.



The diagram below illustrates how this might be done. A requirement that a user can log into a website is implemented (dependency link) by the 'Login' use case, which in turn is implemented by the Business Logic, ASP pages and Login Web Page. Using this approach, you can model quite detailed and complex dependencies and implementation relationships with ease.



5.3.4.14 Screen

A *screen* is a stereotyped class representing a User Interface screen. It is for prototyping and screen flow purposes only. By using UML features such as requirements, constraints and scenarios against user interface elements, you can build up a solid and detailed understanding of user interface behavior without having to use code. This becomes an excellent means of establishing the precise behavior the system will have from a user perspective, and in conjunction with the use case model, defines exactly how a user will get work done.

Web Pages may also be prototyped and specified rigorously using EA's custom interface extensions.

The example diagram below illustrates some features of EA's screen modeling extensions that support web

page prototyping. By adding requirements, rules, scenarios and notes to each element, a detailed model is built up of the form or web page, without having to resort to GUI builders or HTML.

Note: EA displays a selection of stereotyped UI Controls with special icons - for example a control stereotyped as a <<List>> will display with a vertical scroll bar.

Stereotypes and their Effects

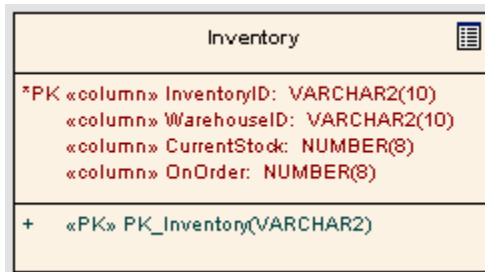
The following details the stereotypes you can add and the effect they produce:

- Vertical Scroll Bar for: list, treelist, report, listview
- Simple rectangle for: textbox, editbox, text, edit, input, date, time
- Shaded rectangle for: form, panel, dialog, dialogue
- Dark rectangle for: button, push button, action
- Combo-box type for: combobox, combo, dropdown, drop list
- Check box for: check, checkbox, check box, tick
- Radio button for: radio, group, option

The screenshot shows a 'Payment Screen' window with a blue title bar. The main content area is light gray and contains several elements:

- A scrollable text area labeled 'Order Details' with a vertical scrollbar on the right.
- A text box labeled 'OrderTotal' positioned below the 'Order Details' area.
- Another scrollable text area labeled 'Ship Details' with a vertical scrollbar on the right.
- A section containing three text boxes labeled 'CreditCard#', 'ExpiryDate', and 'NameOnCard' stacked vertically.
- Two buttons labeled 'Submit' and 'Cancel' stacked vertically to the right of the text boxes.
- Two buttons labeled 'Back' and 'View Cart' at the bottom of the window.

5.3.4.15 Table



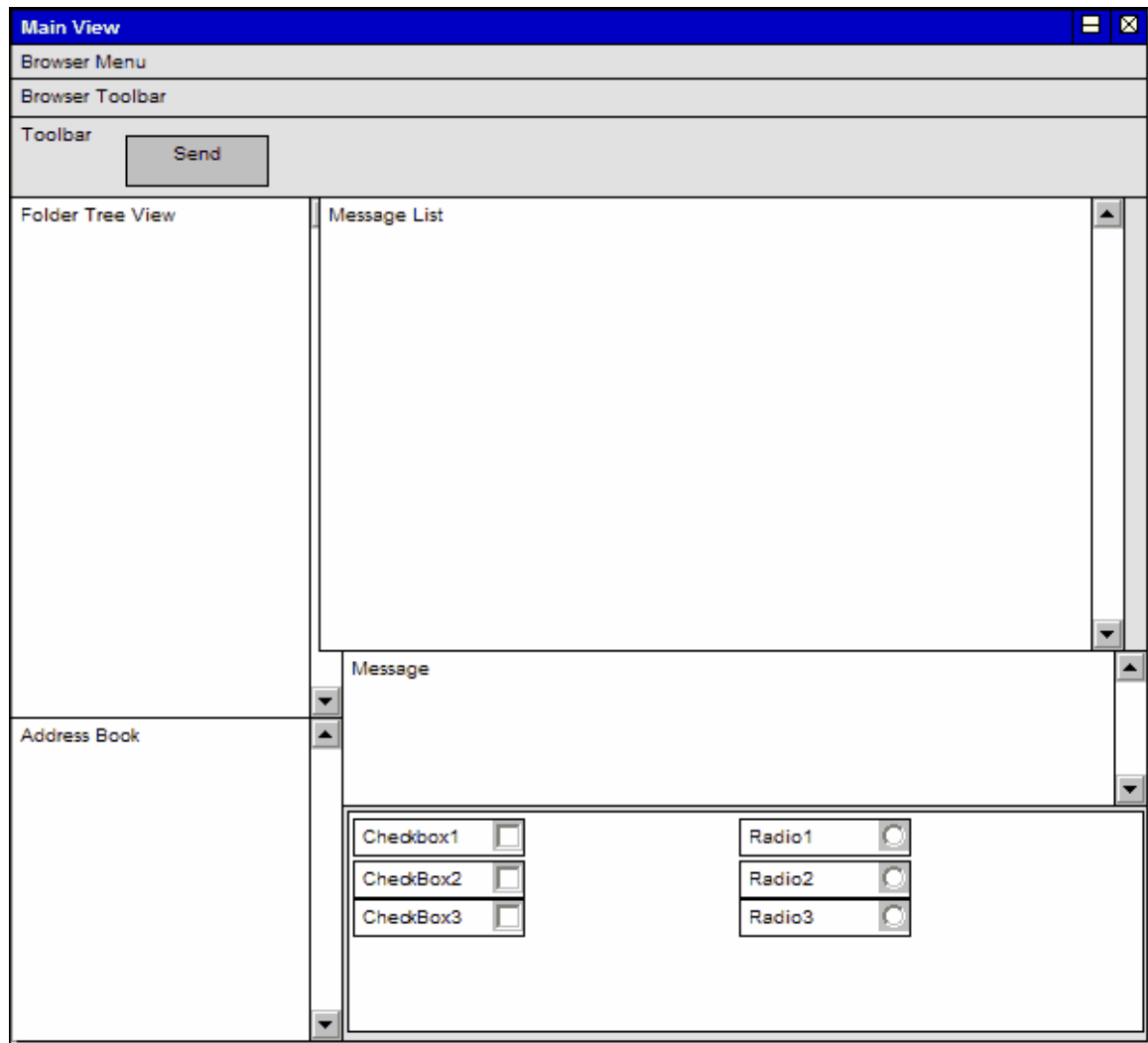
A *table* is a stereotyped class. It is drawn with a small table icon in the upper right corner. A table element has a special properties dialog with settings for database type and the ability to set column information and data related operations such as triggers and indexes. When setting up a table, make sure you set the default database type for that table - otherwise you will not have any data types to choose from when creating columns.

Note: PK and FK tags indicate primary and foreign key respectively.

5.3.4.16 UI Element

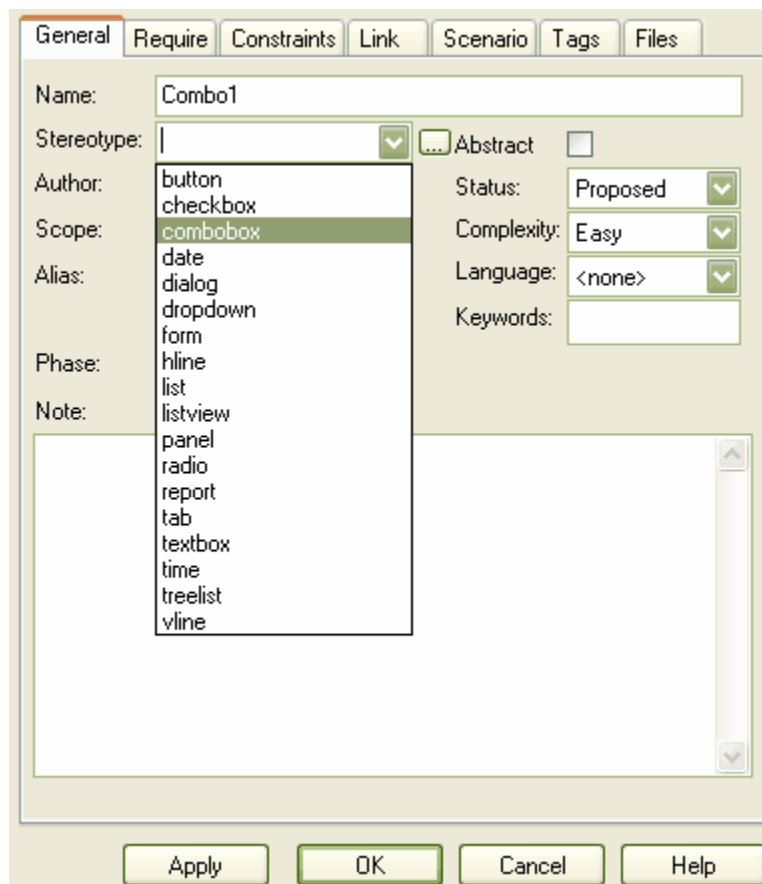
A *UI element* is a stereotyped class representing any user interface control element (eg. edit box). It is for capturing screen layouts and requirements. User interface controls may be drawn in a variety of ways depending on the element stereotype:

- Vertical Scroll Bar for: list, treelist, report, listview
- Simple rectangle for: textbox, editbox, text, edit, input, date, time
- Shaded rectangle for: form, panel, dialog, dialogue
- Dark rectangle for: button, push button, action
- Combo-box type for: combobox, combo, dropdown, drop list
- Check box for: check, checkbox, check box, tick
- Radio button for: radio, group, option



Set the UI Element stereotype to one of the words in the above list to change its representation use the following instructions:

1. Create a Custom diagram and then from the *UML toolbox* select the *Custom* tools
2. Drag an UI Control from the UML toolbox onto the diagram.
3. Give the UI Control element the appropriate stereotype from the *Stereotype* dropdown list in the GUIElement : UI Control dialog.



4. Give the element an appropriate name and then press the *Apply* button.

5.3.4.17 Web Stereotypes

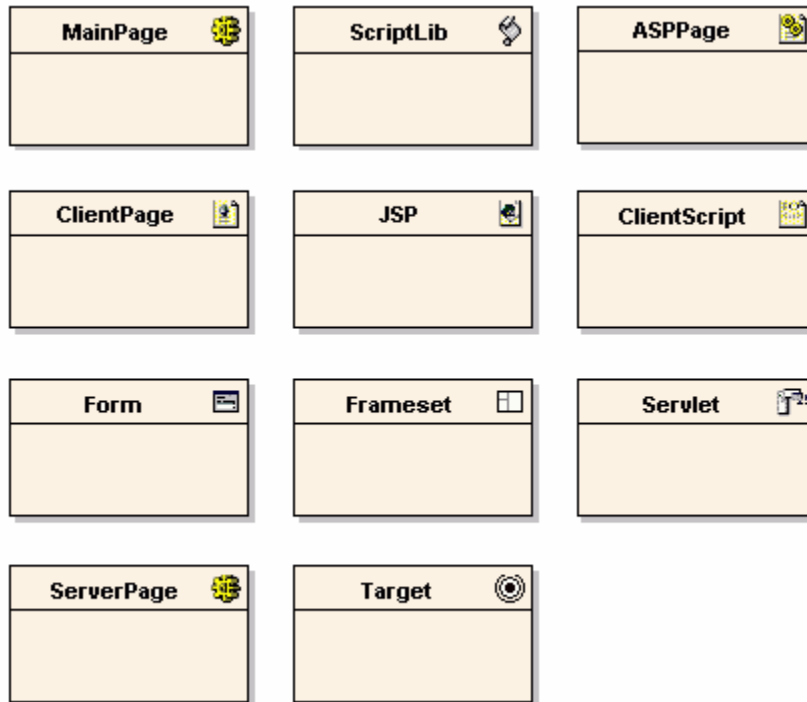
Enterprise Architect supports a number of stereotypes for web page modeling. These supported types will display with a graphical icon instead of the usual <<stereotype>> format. These stereotypes are only supported for elements of type 'Class'. The image below indicates the various graphical icons and their associated stereotypes.

Set a Web Icon

To set a web icon, follow the steps below:

1. Create a new class and place in a diagram.
2. Bring up the *Class properties* dialog.
3. From the drop down *Stereotype* combo, select the required stereotype (will be similar to examples below).
4. Press *OK*.

If you have selected a supported stereotype, the class will display as in one of the examples below:



5.3.4.18 Worker



Some additional stereotyped classes are available for performing business process modeling. These stereotypes are used to model the workers within and external to the system.

The additional stereotypes are





- *Case worker* (class with stereotype 'case worker')
- *Internal worker* (class with stereotype 'internal worker')
- *Worker* (class with stereotype 'worker')

5.4 UML Connections









What is a Connection?

A connection is a *logical or functional relationship between model elements*. There are several different connection types, each having a particular purpose and syntax. Enterprise Architect supports all of the UML connections as well as some custom ones of its own. Together with the [UML Elements](#), these form the basis of UML models. For more insight into using these connectors, consult the appropriate topic by clicking on the table below.



Behavioral Diagram Connectors**Activity Diagrams**

| | |
|---|----------------|
|  | Control Flow |
|  | Object Flow |
|  | Dependency |
|  | Interrupt Flow |


Use Case Diagrams

| | |
|---|------------|
|  | Use |
|  | Associate |
|  | Generalize |
|  | Include |
|  | Extend |
|  | Realize |
|  | Dependency |
|  | Trace |

State Diagrams

| | |
|---|-------------|
|  | Transition |
|  | Object Flow |





Timing Diagrams

| | |
|---|---------|
|  | Message |
|---|---------|





Sequence Diagrams

| | |
|---|--------------|
|  | Message |
|  | Self-Message |
|  | Recursion |
|  | Call |








Communication Diagrams

| | |
|---|------------|
|  | Associate |
|  | Dependency |
|  | Realize |
|  | Nesting |














Interaction Overview Diagrams

| | |
|---|----------------|
|  | Control Flow |
|  | Object Flow |
|  | Dependency |
|  | Interrupt Flow |








Structural Diagram Connectors**Composite Structure Diagrams**

| | |
|---|------------------|
|  | Expose Interface |
|  | Connector |
|  | Assembly |
|  | Delegate |
|  | Role Binding |
|  | Represents |
|  | Occurrence |








Package, Class and Object Diagrams

| | |
|---|-------------------|
|  | Association |
|  | Associate |
|  | Generalize |
|  | Compose |
|  | Aggregate |
|  | Association Class |
|  | Assembly |
|  | Dependency |
|  | Realize |
|  | Trace |
|  | Nesting |
|  | Pkg Merge |
|  | Pkg Import |



Component Diagrams

| | |
|---|------------|
|  | Assembly |
|  | Delegate |
|  | Associate |
|  | Realize |
|  | Dependency |
|  | Trace |
|  | Generalize |





Inbuilt and Extended Connectors**Analysis Diagrams**

| | |
|---|------------------|
|  | Information Flow |
|  | Object Flow |
|  | Associate |
|  | Realize |
|  | Dependency |
|  | Trace |
|  | Representation |

Use Case Diagrams

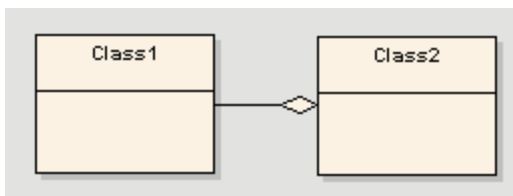
| | |
|---|----------|
|  | Invokes |
|  | Precedes |

Deployment Diagrams

| | |
|---|--------------------|
|  | Associate |
|  | Communication Path |
|  | Association Class |
|  | Generalize |
|  | Realize |
|  | Deployment |
|  | Manifest |
|  | Dependency |
|  | Trace |
|  | Object Flow |
|  | Nesting |

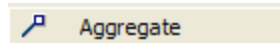
5.4.1 Aggregate

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *aggregation* relationship shows that an element *contains* or is *composed of* other elements. Used in class models to show how more complex elements are built from a collection of simpler elements (eg. a car from wheels, tyres, motor, etc.). A stronger form of aggregation, known as *composite aggregation*, is used to indicate ownership of the whole over its parts. After drawing an aggregation association, its form can be changed; consult the instructions under *Further Information*.

Common Usage



Aggregate

- [Class Diagram](#)
- [Package Diagram](#)
- [Object Diagram](#)

Further Information

- [Change the Form of an Aggregation Link](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 5) states:

"A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part."

The OMG UML specification (*UML 2.0 Superstructure*, p. 82) states:

"Composite aggregation is a strong form of aggregation that requires a part instance be included in at most one composite at a time. If a composite is deleted, all of its parts are normally deleted with it."

5.4.1.1 Change Aggregation Link Form

[Further Information](#)

In Enterprise Architect, the default aggregation relationship is the weak form of the relationship, represented by a hollow diamond. To change the form of an aggregation link from weak to strong, follow the steps below.

1. Right click on an aggregation link to bring up the context menu.
2. Select *Set Aggregation to Composite*.
3. The diamond will now be shown as filled.

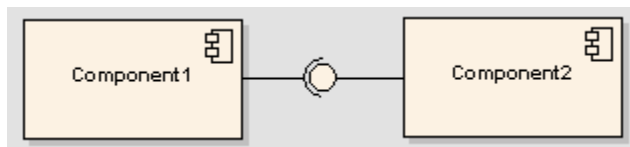
Note: *If the link is already a Strong (Composition) link, the context menu option will change to *Set Aggregation to Shared*.*

Further Information

- [Aggregate](#)

5.4.2 Assembly

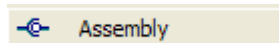
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



As shown above, the *assembly* connector bridges a component's required interface (Component1) with the provided interface of another component (Component2).

Common Usage

- [Component Diagram](#)



Further Information

- [Interface](#)
- [Expose Interface](#)

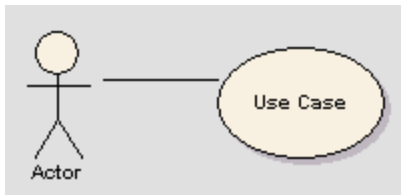
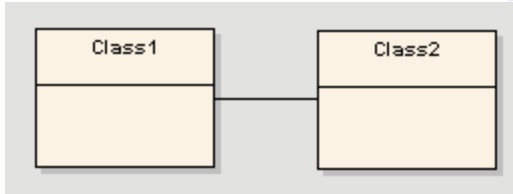
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 143) states:

"An assembly connector is a connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."

5.4.3 Associate

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

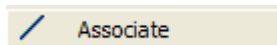


An *association* implies two model elements have a relationship - usually implemented as an instance variable in one class. This connector may include named roles at each end, multiplicity, direction and constraints. Association is the general relationship type between elements. For more than two elements, a [diagonal representation](#) toolbox element can be used as well.

When code is generated for class diagrams, associations become instance variables in the target class.

Common Usage

- [Class Diagram](#)
- [Package Diagram](#)
- [Object Diagram](#)
- [Communication Diagram](#)
- [Deployment Diagram](#)



Further Information

- [Diagonal Representation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 81) states:

"An association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type. When a property is owned by an association it represents a non-navigable end of the association. In this case the property does not appear in the namespace of any of the associated classifiers. When a property at an end of an association is owned by one of the associated classifiers it represents a navigable end of the association. In this case the property is also an attribute of the associated classifier. Only binary associations may have navigable ends."

5.4.3.1 Diagonal Representation

[Common Usage](#) | [Further Information](#)



Diagonal representation can be used to graphically ease ternary or multiply associated elements.

Common Usage

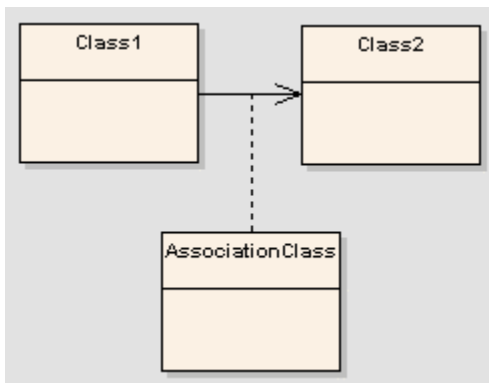
- [Class Diagram](#)

Further Information

- [Associate](#)

5.4.4 Association Class

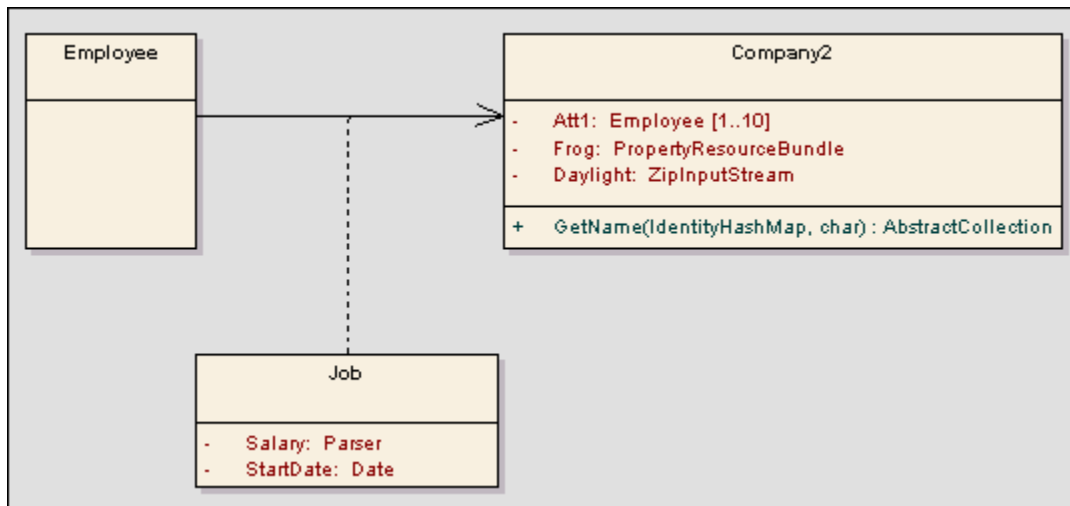
[Common Usage](#) | [OMG UML Specification](#)



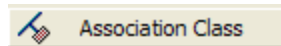
An *association class* connection is a UML construct that allows an association connector to have attributes and operations (features). This results in a hybrid relation with the characteristics of a linkage and a class. When an association class link is added, EA creates a class as well. This is automatically linked to the association. When you delete the association, the class is deleted also. If you hide the association, the class is hidden also. An association class is used to model particular types of connections in UML (see the [UML Specification](#) for more details).

To add an association class - in the UML toolbox select Structure, find the **Association Class** icon and click once. Click on the source object in the diagram and hold the mouse down while you drag the line to the target element - release the mouse button. EA will draw the link and add the class. You will be prompted to add the class name - note that the name of the class and the link are the same.

The following diagram illustrates an association class between model elements. Note the dotted line from the class to the association. This line is not movable or able to be deleted.



Common Usage



- [Deployment Diagram](#)
- [Class Diagram](#)

Further Information

- [Link a New Class to an Existing Association](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 118) states:

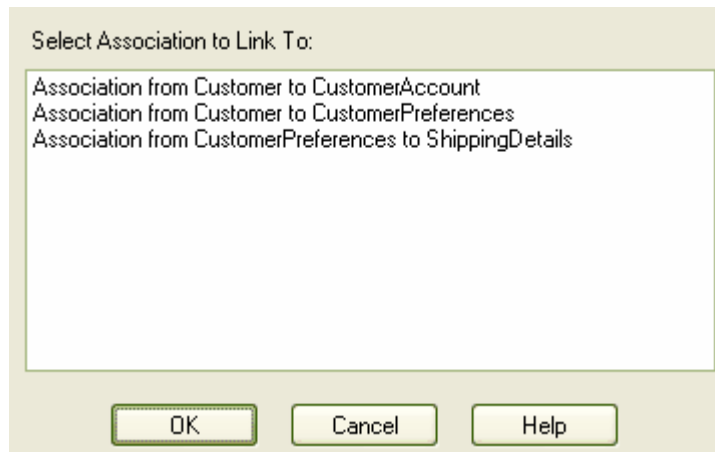
"A model element that has both association and class properties. An AssociationClass can be seen as an association that also has class properties, or as a class that also has association properties. It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not to any of the classifiers."

5.4.4.1 Link a New Class to an Existing Association

[Further Information](#)

To link a new class to an existing association, follow the steps below:

1. Create a class in the diagram workspace containing the association to link.
2. Right click on the new class for the context menu.
3. Select *Link Class to Association*.



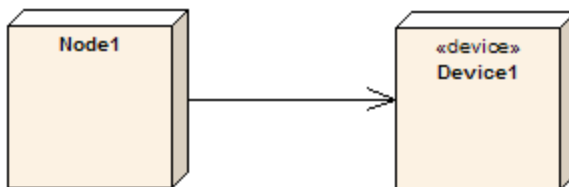
3. In the *Link class to association* dialog, check the link to connect to.
4. Press *OK*.

Further Information

- [Association Class](#)

5.4.5 Communication Path

[Common Usage](#) | [OMG UML Specification](#)



A communication path defines the path through which two DeploymentTargets are able to exchange signals and messages. Communication path is a specialization of Association. A DeploymentTarget is the target for a deployed Artifact and may be a Node, Property or InstanceSpecification.

Common Usage

- Deployment Diagram

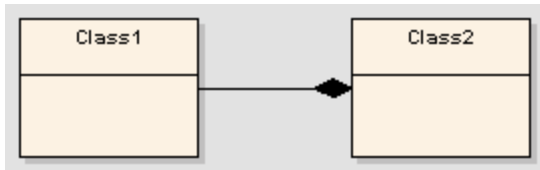
OMG UML Specification

The OMG UML specification ([UML 2.1 Superstructure](#), section 10.3.2) states:

"A communication path is an association between two DeploymentTargets, through which they are able to exchange signals and messages."

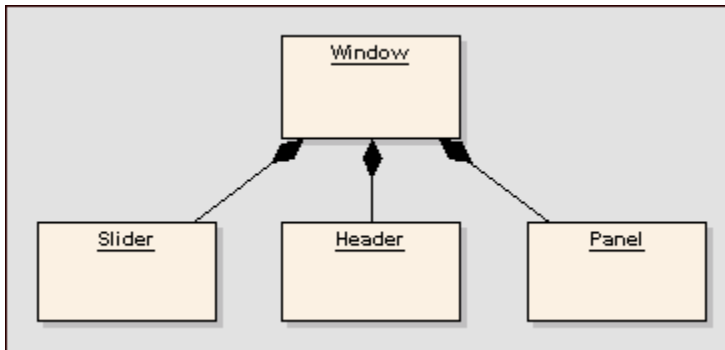
5.4.6 Compose

[Common Usage](#) | [OMG UML Specification](#)



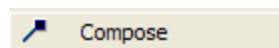
A *composite aggregation* is used to depict an element which is made up of smaller components. A component - or part instance - can be included in a maximum of one composition at a time. If a composition is deleted, usually all of its parts are deleted with it; however a part can be individually removed from a composition without having to delete the entire composition. Compositions are transitive, asymmetric relationships and can be recursive.

See example below.



Common Usage

- [Class Diagram](#)
- [Package Diagram](#)



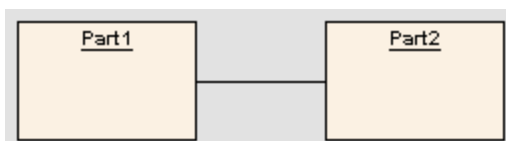
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure, p. 7*) states:

"A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive."

5.4.7 Connector

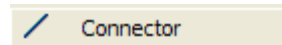
[Common Usage](#) | [OMG UML Specification](#)



Connectors illustrate communication links between parts to fulfill the structure's purpose. Each connector end is distinct, controlling the communication pertaining to its connecting element. These elements can define constraints specifying this behavior. Connectors can have multiplicity.

Common Usage

- [Composite Structure Diagram](#)



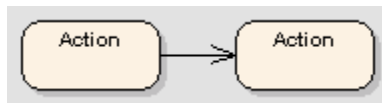
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 163) states:

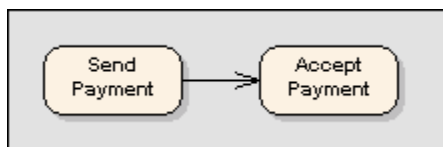
"Specifies a link that enables communication between two or more instances. This link may be an instance of an association, or it may represent the possibility of the instances being able to communicate because their identities are known by virtue of being passed in as parameters, held in variables, created during the execution of a behavior, or because the communicating instances are the same instance. The link may be realized by something as simple as a pointer or by something as complex as a network connection. In contrast to associations, which specify links between any instance of the associated classifiers, connectors specify links between instances playing the connected parts only."

5.4.8 Control Flow

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



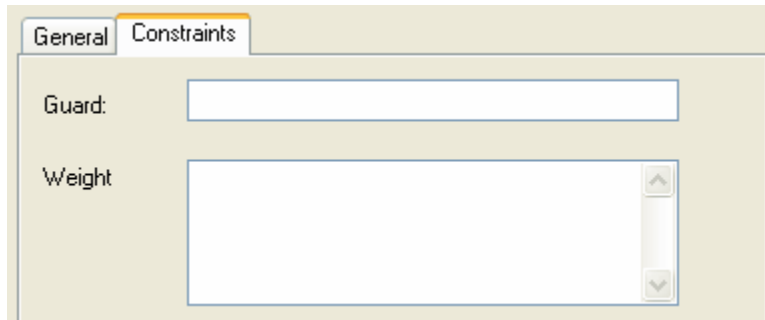
The *control flow* is a connector linking two nodes in an Activity diagram. Control flow connectors bridge the flow between activity nodes, by directing the flow to the target node once the source node's activity is completed.



Control Flows and Object Flows may define a Guard and a Weight condition.

A **Guard** defines a condition that must be true before control will pass along that activity edge. A practical example of this is where two or more activity edges (control flows) exit from a Decision element. Each flow should have a Guard condition which is exclusive of each other and defines which edge will be taken under what conditions. The ControlFlow properties dialog allows you to set-up Guard conditions on Control Flows and on Object Flows.

A **Weight** defines the number of tokens that may flow along a Control or Object Flow connection when that edge is traversed. Weight may also be defined on the ControlFlow and ObjectFlow properties dialog.

**Common Usage**

- [Activity Diagrams](#)

Further Information

- [Actions](#)
- [Activity Diagrams](#)

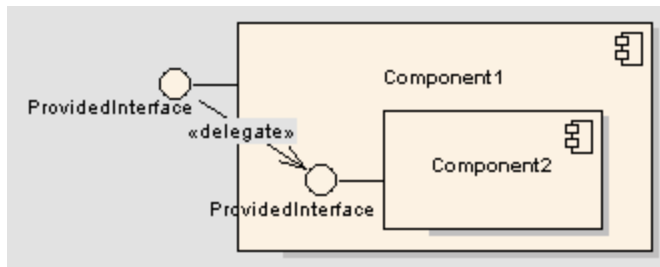
The OMG UML specification states:

The OMG UML specification (*UML 2.0 Superstructure*, p. 315) states:

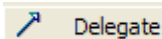
"A control flow is an edge starts an activity node after the previous one is finished."

5.4.9 Delegate

[Common Usage](#) | [OMG UML Specification](#)



A *delegate* connector defines the internal assembly of a component's external ports and interfaces. Using a delegate connector wires the internal workings of the system to the outside world, by a delegation of the external interfaces' connections.

Common Usage

- [Component Diagram](#)

OMG UML Specification

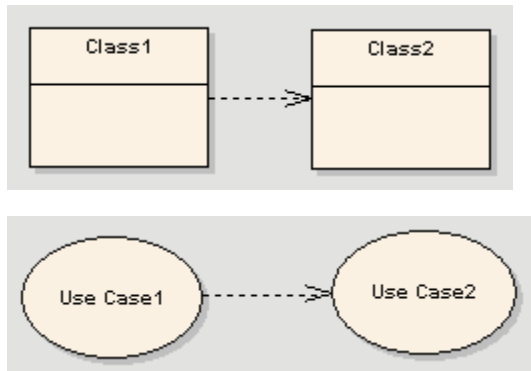
The OMG UML specification (*UML 2.0 Superstructure*, p. 143) states:

"A *delegation connector* is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component's parts. It represents the forwarding of signals (operation requests and events) : a signal that arrives at a port that has a delegation connector to

a part or to another port will be passed on to that target for handling."

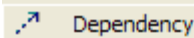
5.4.10 Dependency

[Common Usage](#) | [OMG UML Specification](#)



Dependency relationships are used to model a wide range of dependent relationships between model elements - and even between models themselves. The Dependencies package as defined in UML 2.0 has many derivatives, such as Realization, Instantiation and Usage. Once you create a dependency you can further refine its meaning by applying a specialized stereotype.

Common Usage



Dependency

- [Use Case Diagram](#)
- [Structural Diagrams](#)
- [Activity Diagram](#)

Further Information

- [Applying a Stereotype](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 108) states:

"A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s)."

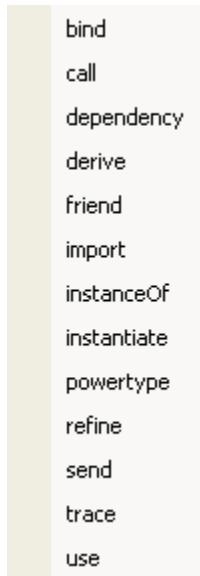
5.4.10.1 Applying a Stereotype

[Further Information](#)

To apply a stereotype to a dependency relationship, follow the steps below:

1. Select the dependency relationship to change.

2. Right click on the link, and from the context menu, select *Dependency Stereotypes*.
3. From the sub-menu select the required Stereotype:

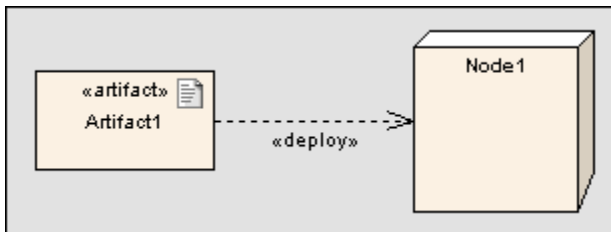


Further Information

- [Dependency](#)

5.4.11 Deployment

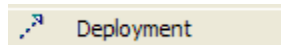
[Common Usage](#) | [OMG UML Specification](#)



A *deployment* is a type of dependency relationship that indicates the deployment of an artifact onto a node or executable target. A deployment can be made at type and instance levels. At the type level, a deployment would be made for every instance of the node. Deployment can also be specified for an instance of a node, so that a node's instances can have varied deployed artifacts. With composite structures modeled with nodes defined as parts, parts can also serve as targets of a deployment relationship.

Common Usage

- [Deployment Diagram](#)



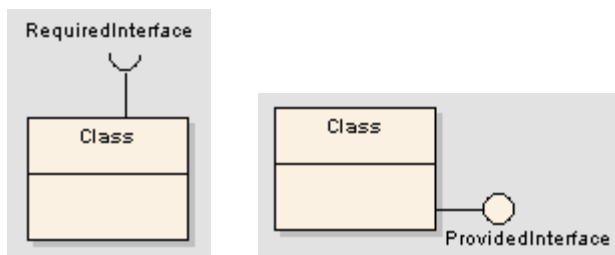
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 187) states:

"A deployment is the allocation of an artifact or artifact instance to a deployment target."

5.4.12 Expose Interface

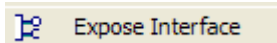
[Common Usage](#) | [Further Information](#)



The *expose interface* toolbox element is a graphical way to depict the required and supplied interfaces of a component, class, or part.

Common Usage

- [Component Diagram](#)
- [Class Diagram](#)
- [Composite Structure Diagram](#)

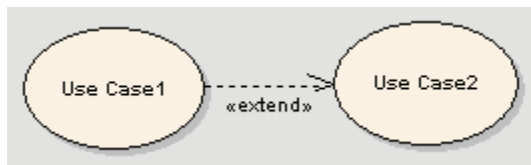


Further Information

- [Interface](#)
- [Assembly](#)

5.4.13 Extend

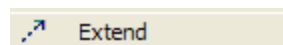
[Common Usage](#) | [OMG UML Specification](#)



An *extend* connection is used to indicate an element extends the behavior of another. Extensions are used in use case models to indicate one use case (optionally) extends the behavior of another. An extending use case often expresses alternate flows.

Common Usage

- [Use Case Diagram](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 515) states:

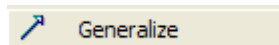
"This relationship specifies that the behavior of a use case may be extended by the behavior of another (usually supplementary) use case. The extension takes place at one or more specific extension points defined in the extended use case. Note, however, that the extended use case is defined independently of the extending use case and is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending use case defines a set of modular behavior increments that augment an execution of the extended use case under specific conditions. Note that the same extending use case can extend more than one use case. Furthermore, an extending use case may itself be extended."

5.4.14 Generalize

[Common Usage](#) | [OMG UML Specification](#)



A *generalization* is used to indicate inheritance. Drawn from the specific classifier to a general classifier, the generalize implication is that the source inherits the target's characteristics.

Common Usage

- [Class Diagram](#)
- [Component Diagram](#)
- [Object Diagram](#)
- [Package Diagram](#)
- [Use Case Diagram](#)

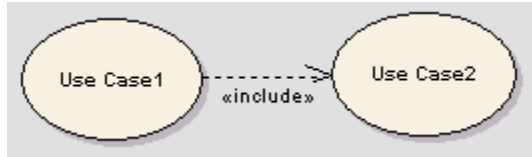
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 66) states:

"A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier."

5.4.15 Include

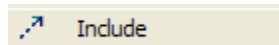
[Common Usage](#) | [OMG UML Specification](#)



An *include* connection indicates that the source element includes the functionality of the target element. Include connections are used in use case models to reflect that one use case includes the behavior of another. Use an include relationship to avoid having the same subset of behavior in many use cases; this is similar to [delegation](#) used in class models.

Common Usage

- [Use Case Diagram](#)



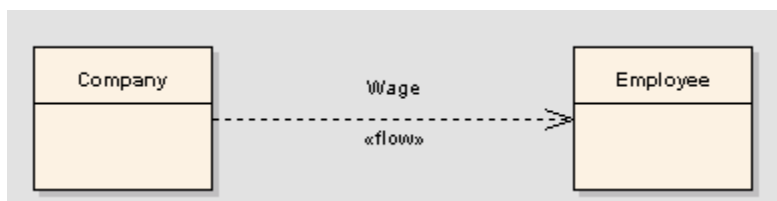
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 518) states:

"Include is a directed relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The including use case may only depend on the result (value) of the included use case. This value is obtained as a result of the execution of the included use case."

5.4.16 Information Flow

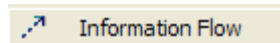
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



The information flow is a connector linking two nodes in an Activity diagram. The information flow represents information item(s) or classifier(s) flowing between two elements.

Common Usage

- [Activity Diagram](#)



Further Information

- [Information Item](#)

OMG UML Specification

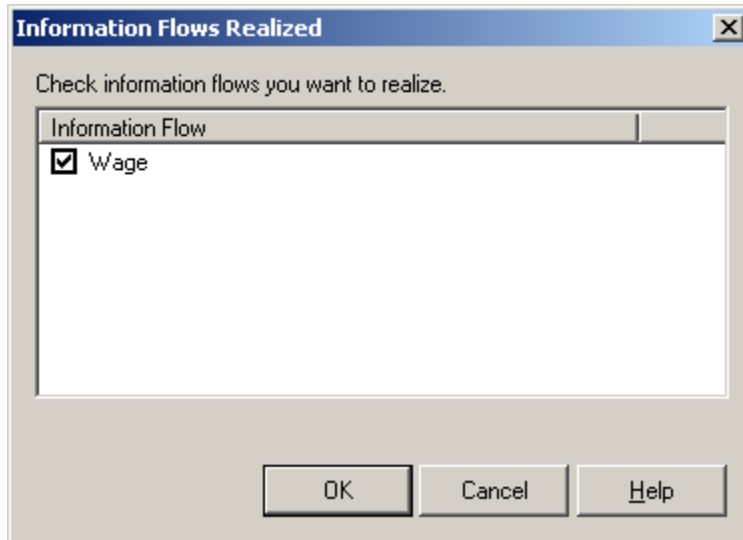
The OMG UML specification (*UML 2.1 Superstructure*, p. 634) states:

"An InformationFlow specifies that one or more information items circulates from its sources to its targets."

The OMG UML specification ([UML 2.1 Superstructure](#), p. 635) states:

“An information flow is an abstraction of the communication of an information item from its sources to its targets. It is used to abstract the communication of information between entities of a system. Sources or targets of an information flow designate sets of objects that can send or receive the conveyed information item.”

5.4.16.1 Realizing an Information Flow



This dialog displays all flows that are able to be realized on this connector. To realize an information flow on this connector, check the corresponding checkbox.

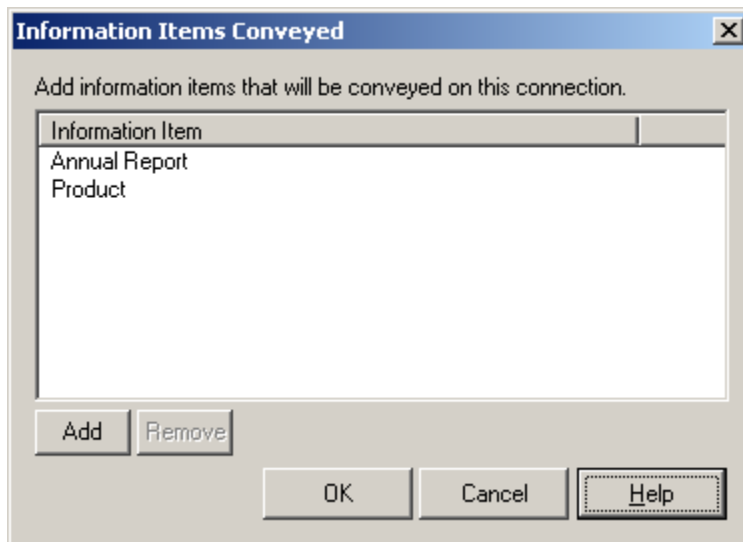
See Also

- [Information Flow](#)
- [UML Connections](#)

5.4.16.2 Convey Information on a Flow

Once you have an information flow between two elements, you can specify which information items or classifiers are conveyed will be conveyed on this flow.

To specify these information items or classifiers, **Right-Click** on the connection and select **Information Item Conveyed**. This dialog allows you to specify what information items or classifiers will be conveyed on this flow.



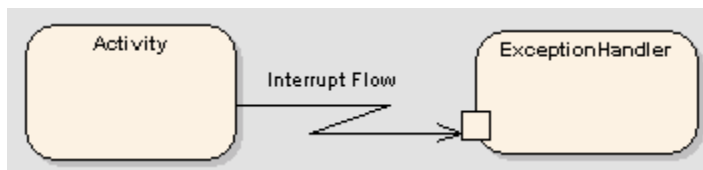
| | |
|--------|---|
| Add | Add an Information or Classifier Element. |
| Remove | Remove selected item. |

See Also

- [Information Flow](#)
- [Information Item](#)

5.4.17 Interrupt Flow

[Common Usage](#) | [OMG UML Specification](#)



The *interrupt flow* is a toolbox element used to define the two UML concepts of connectors for [Exception Handler](#) and [Interruptible Activity Region](#). An interrupt flow is also known as an activity edge.

Common Usage Interrupt Flow

- [Activity Diagram](#)

Further Information

- [Activity Diagram](#)
- [Exception handler](#)
- [Interruptible Activity Region](#)

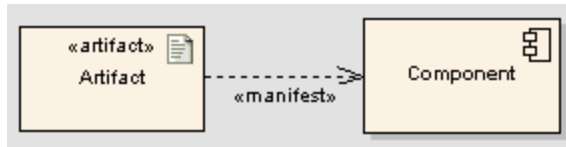
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 293) states:

"An activity edge is an abstract class for directed connections between two activity nodes."

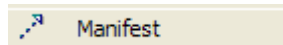
5.4.18 Manifest

[Common Usage](#) | [OMG UML Specification](#)



A *manifest* relationship indicates that the artifact source embodies the target model element. [Stereotypes](#) can be added to Enterprise Architect to classify the type of manifestation of the model element.

Common Usage



- [Component Diagram](#)
- [Deployment Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 195) states:

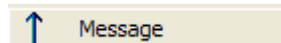
"An artifact embodies or manifests a number of model elements. The artifact owns the manifestations, each representing the utilization of a packageable element. Specific profiles are expected to stereotype the manifestation relationship to indicate particular forms of manifestation, e.g. <<tool generated>> and <<custom code>> might be two manifestations for different classes embodied in an artifact."

5.4.19 Message

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

Messages indicate a flow of information or transition of control between elements. Messages can be used by all interaction diagrams except the Interaction Overview diagram, to reflect system behavior. If between classes or classifier instances, the associated list of operations will be available to specify the event.

Common Usage



- [Sequence Diagram](#)
- [Communication Diagram](#)
- [Timing Diagram](#)

Further Information

- [Messages for Timing Diagrams](#)
- [Messages for Sequence Diagrams](#)
- [Messages for Communication Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 428) states:

"A Message defines a particular communication between Lifelines of an Interaction."

"A Message is a NamedElement that defines one specific kind of communication in an Interaction. A communication can be e.g. raising a signal, invoking an Operation, creating or destroying an Instance. The Message specifies not only the kind of communication given by the dispatching ExecutionOccurrence, but also the sender and the receiver.

"A Message associates normally two EventOccurrences - one sending EventOccurrence and one receiving EventOccurrence."

Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.4.19.1 Message (Communication Diagram)

[Common Usage](#) | [Further Information](#)

A message in a Communication diagram is equivalent in meaning to a message in a Sequence diagram. It implies that one object uses the services of another object, or sends a message to that object. Communication messages in Enterprise Architect are always associated with an association link between object instances. Always create the association link first - then add messages to the link.

Messages may be dragged into a suitable position by clicking and dragging on the message text.

Communication messages should be ordered to reflect the sequencing of the diagram. The numbering scheme should reflect the nesting of each event. A sample sequencing scheme could be 1, 2, 2.1, 2.2, 3. This would indicate events 2.1 and 2.2 occur within an operation initiated by event 2.

If the target object is a class or has its instance classifier set, the drop-down list of possible message names will include the exposed operations for the base type.

Common Usage

- [Communication Diagrams](#)

Further Information

- [Create a Communication Message](#)
- [Ordering Communication Messages](#)

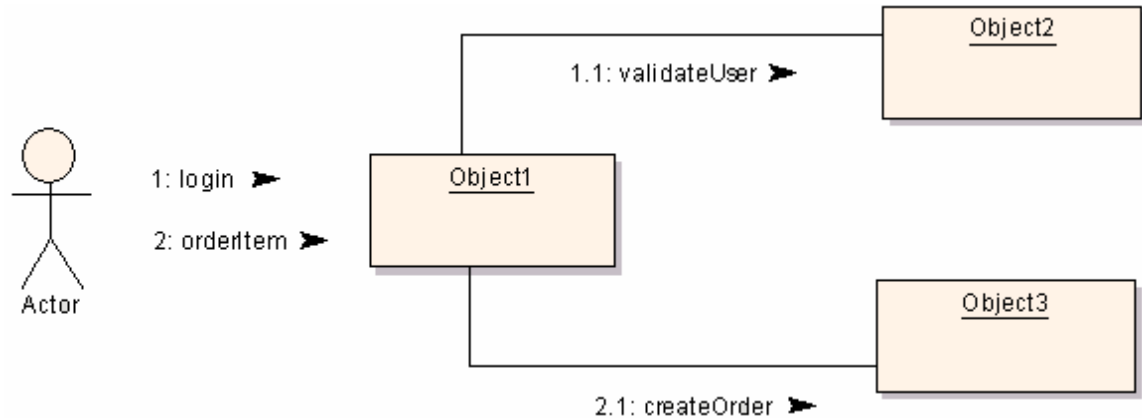
5.4.19.1.1 Create a Communication Message

[Further Information](#)

To create a communication message, follow the steps below:

1. Open a diagram (must be one of: Communication, Analysis, Interaction Overview, Object, Activity or State Machine).
2. Add the required objects.
3. Add an association relationship between all objects that will communicate.
4. Right click on an association to view the context menu.
5. Select the option to add a message from one object to another.

6. When the *Message* dialog appears, fill in a *Name* and any other required details.
7. Press *OK*. The message is added - linked to the association and object instances.
8. Move to the required position.



Further Information

- [Message \(Communication\)](#)

5.4.19.1.2 Re-Order Messages

[Further Information](#)

When constructing your Communication diagram, it is frequently necessary to re-order the sequence of messages and to create or delete message 'groups'. There are two dialogs that help you perform these tasks - the *Message Properties* dialog and the *Sequence Communications* dialog.

Organizing Message Groups

If you have several messages that are in the form 1.2, 1.2, 1.3, 1.4 etc., but would like to start a new numbering group on the third message eg. 1.1, 1.2, 2.1, 2.2, 2.3 etc., indicate message 3 is a 'Start Group' message by using the following instructions.

1. View the communication message context menu by right clicking on a message.
2. Select *Communication Properties...* to open the *Message properties* dialog.
3. To make the selected message the start of a new group, check the box called *Start New Group*.
4. Press *OK* to save changes.

The image shows a dialog box for configuring a UML message. It is divided into four sections:

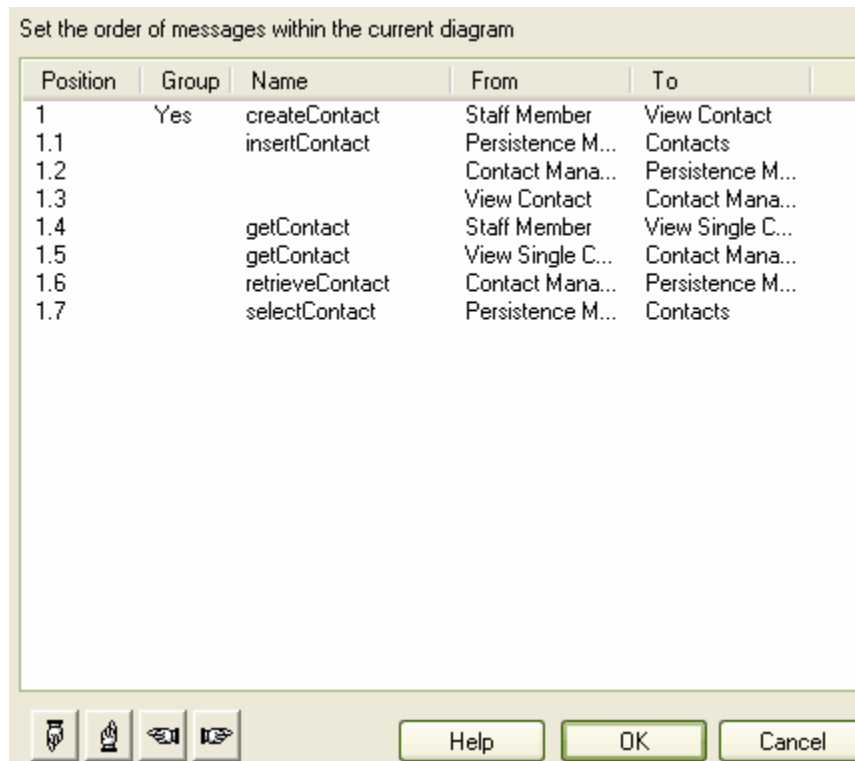
- Signature:** Contains a 'Message:' dropdown menu, a 'Parameters:' text box, a 'Return:' dropdown menu (set to 'void'), an 'Is Return' checkbox (unchecked), and a checked 'Show Inherited Methods' checkbox. An 'Operations' button is located to the right of the 'Message:' dropdown.
- Sequence Expression:** Contains a 'Condition:' text box, a 'Constraint:' text box, and an 'Is Iteration' checkbox (unchecked).
- Control Flow Type:** Contains a 'Synch:' dropdown menu (set to 'Synchronous'), a 'Lifecycle:' dropdown menu, and a 'Kind:' dropdown menu (set to 'Call').
- Notes:** A large text area for entering notes, with scroll bars on the right side.

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

Sequencing Messages

To re-order messages, follow the steps below:

1. Select *Diagram | Sequence Communications* from the main menu to open the *Communication Messages* sequencing dialog.
2. Select the message you want to move, and move up or down the sequence using the hand icons. Repeat until the sequence matches your requirements.
3. Press *OK* to apply changes.



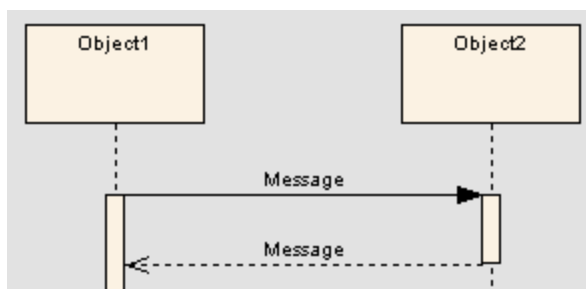
Further Information

- [Message \(Communication\)](#)

Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.4.19.2 Message (Sequence Diagram)

[Common Usage](#) | [Further Information](#)



Sequence diagrams depict work flow or activity over time using messages passed from element to element. These messages correspond in the software model to class operations and behavior. They are semantically similar to the messages passed between elements in a Communication diagram.

Double click on a sequence message to view the *Message properties* dialog. If the destination class has defined operations, these will appear in the message name drop down list to be selected.

You can also set:

- Conditions: indicates what must be true for message to be sent
- Return value
- Synchronization
- Frequency
- Creation (Lifecycle) - set to New for element creation, Delete for element destruction

Note: It is possible to depict returns from a self message. Simply create a second self message at the end of execution and set *Is Return* in the message properties.

Note: To represent an asynchronous message, *Right Click* on message and bring up the *Message Properties* dialog. From the *Control Flow Type*, select *Asynchronous* from the *Synch: Combo box*.

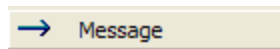
Co-Region Notation

Co-Region notation can be used as a short hand for parallel combined fragments. To access the co-region submenu, *Right Click* on a connector in a sequence diagram and select the *Co-Region* submenu. There are four options available:

- Start at head
- End at head
- Start at tail
- End at tail

Common Usage

- [Sequence Diagram](#)

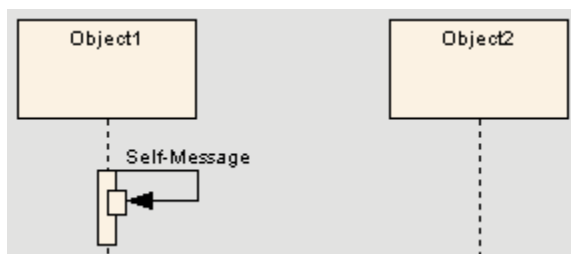


Further Information

- [Self-Message](#)
- [Call](#)
- [Changing the Timing Details](#)
- [Examples](#)

5.4.19.2.1 Self-Message

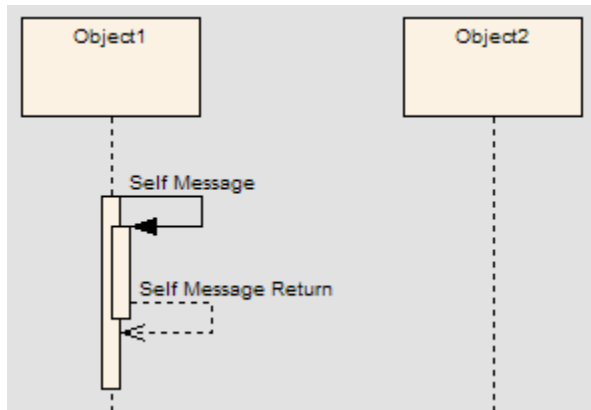
[Common Usage](#) | [Further Information](#)



A *self-message* reflects a new process or method invoked within the calling lifeline's operation. It is a specification of a message.

Self-Message as Return

It is possible to depict a return from a self message call.



To create a self message return:

1. Create a second self message at the end of execution
2. Open the *Message Properties* dialog (*Right Click | Message Properties*)
3. Check the *Is Return* option
4. [Raise the Activation level](#) of the return

Common Usage



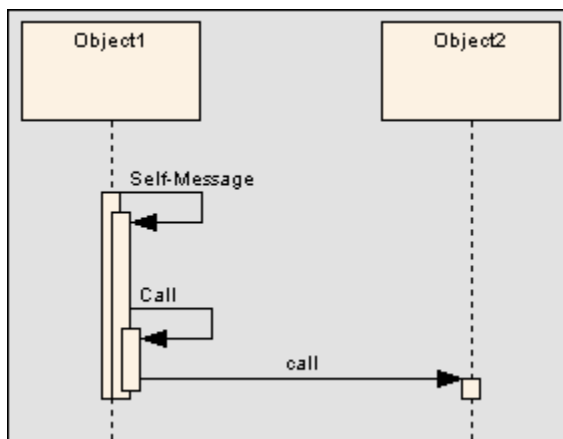
- [Sequence Diagram](#)

Further Information

- [Message](#)

5.4.19.2.2 Call

[Common Usage](#) | [Further Information](#)



A *call* is a type of message element that extends the level of activation from the previous message. All self-messages create a new activation level, but this focus of control usually ends with the next message (unless

[activation levels](#) are manually adjusted). Self-message calls, as depicted above by the first call, indicate a nested invocation; new activation levels are added with each call. Unlike a regular message between elements, a call between elements continues the existing activation in the source element, implying that the call was initiated within the previous message's activation scope.

Common Usage



- [Sequence Diagram](#)

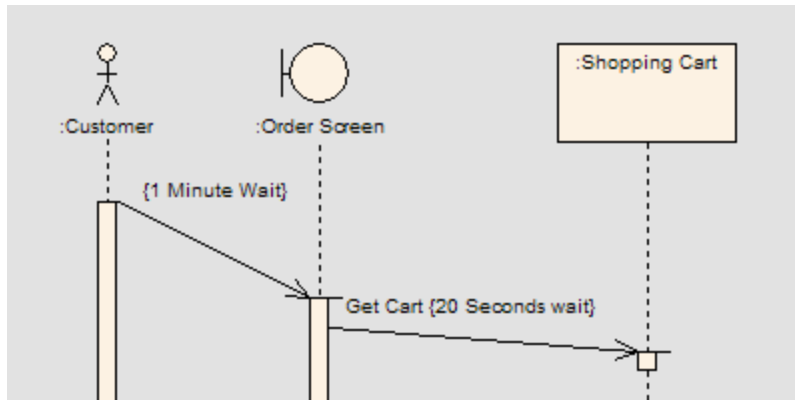
Further Information

- [Message \(Sequence Diagram\)](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)

5.4.19.2.3 Changing the Timing Details

It is possible to change the timing details of a message by right clicking on the message and selecting **Timing Details**.

By filling in the Duration Constraint field, the message angle may now be adjusted. Click on the head of the message connector and drag the connector up or down to change the angle. You can't extend the angle beyond the life line of the connecting sequence object or create an angle less than 5 degrees.

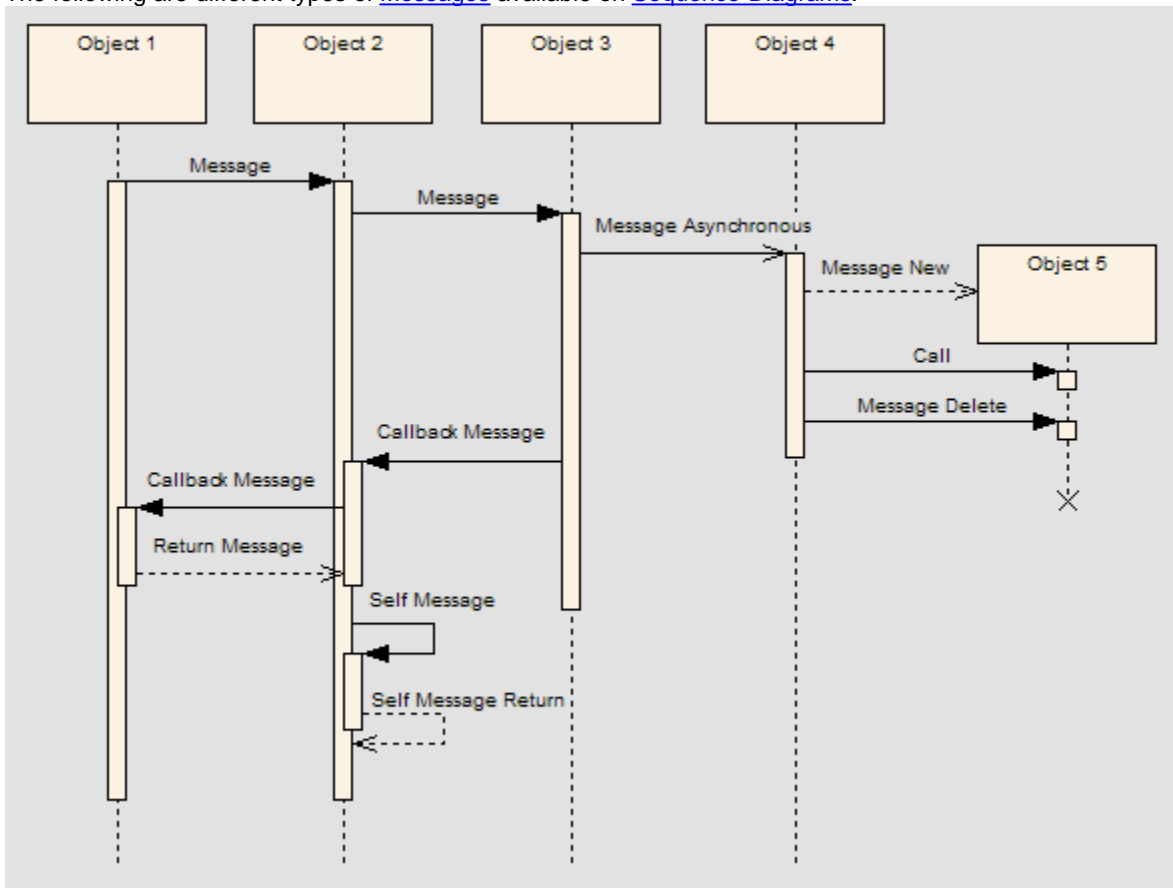


Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Diagram](#)
- [Changing the Top Margin](#)
- [Changing the Timing Details](#)

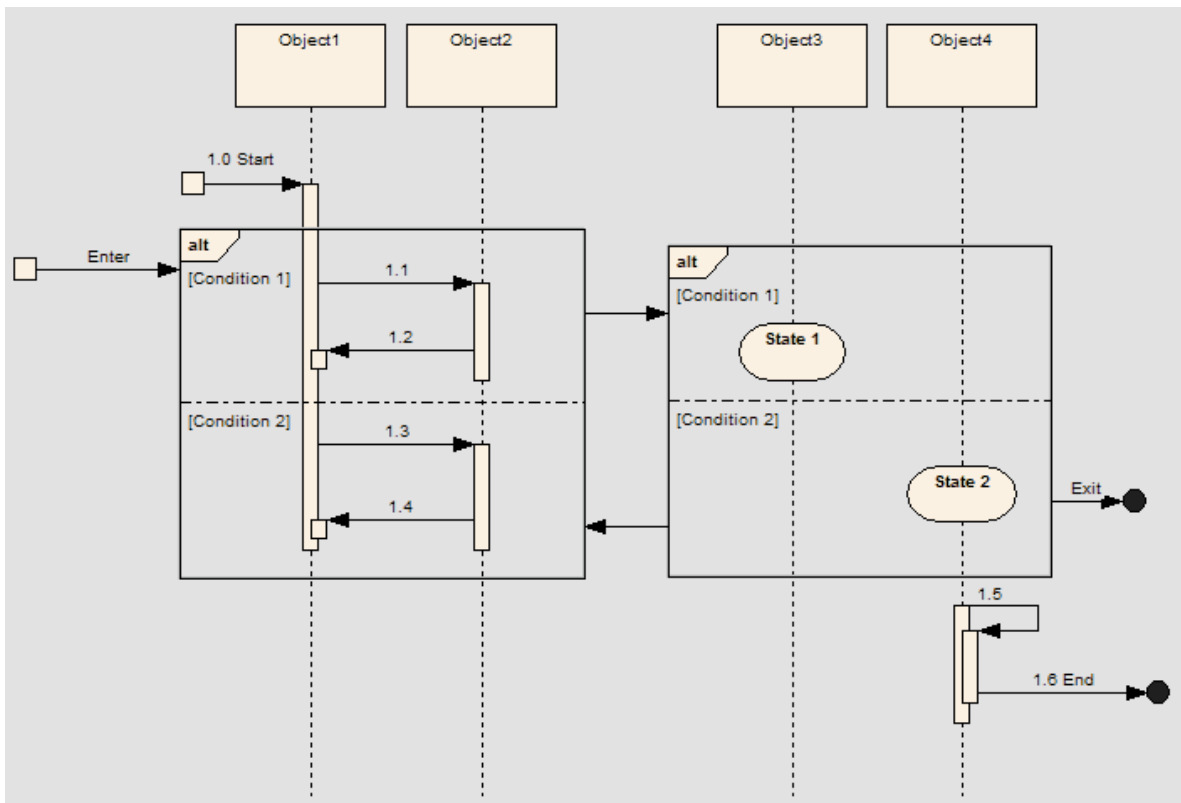
5.4.19.2.4 Message Examples

The following are different types of [Messages](#) available on [Sequence Diagrams](#).



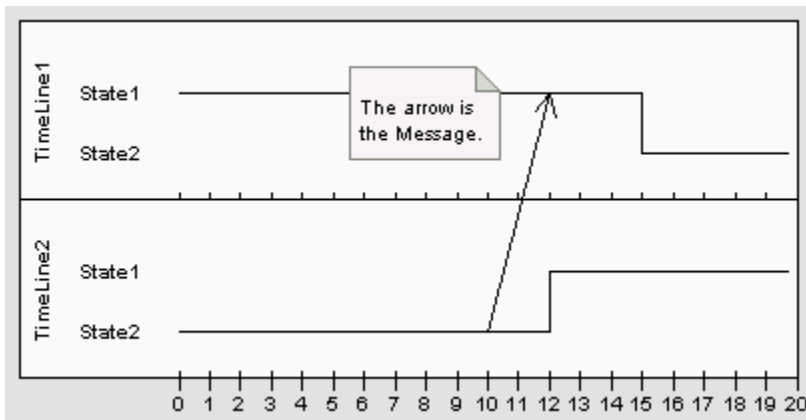
Other Sequence Messages

The following are examples of Messages that are not part of the sequence described by the diagram.

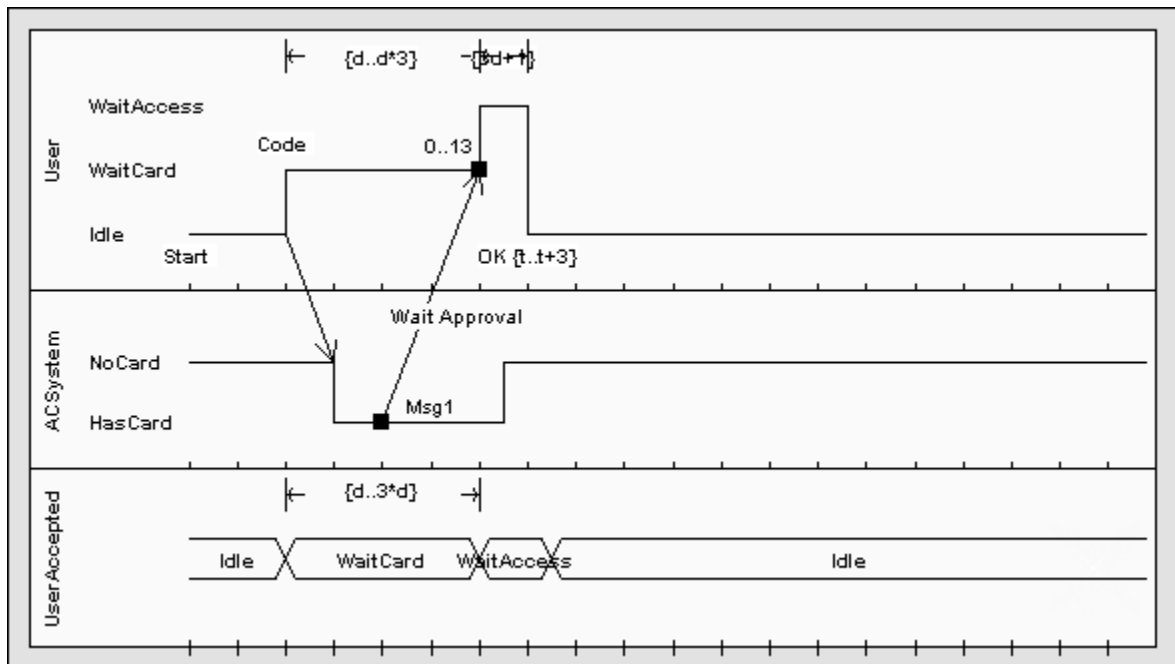


5.4.19.3 Message (Timing Diagram)

[Common Usage](#) | [Further Information](#)

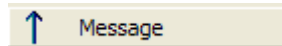


Messages are the communication links between lifelines. In the case of a timeline, a message is a connection between two timeline objects.



Refer to figure 352 (UML 2.0 Superstructure, p. 454) and figure 351 (UML 2.0 Superstructure, p. 453).

Common Usage



- [Timing Diagram](#)

Further Information

- [Timing Diagram](#)

5.4.19.3.1 Create a Timing Message

Further Information

To create a message in a Timing diagram, at least two lifeline objects (state or value), each with existing transition points, must first be created. To create a message between lifelines, follow the steps below:

1. Select a message from the *Timing* section of the UML toolbox.
2. Click on one of the lifelines in the Timeline diagram.
3. Drag the mouse onto the lifeline where the Message is to be connected.

The following dialog box will become visible:

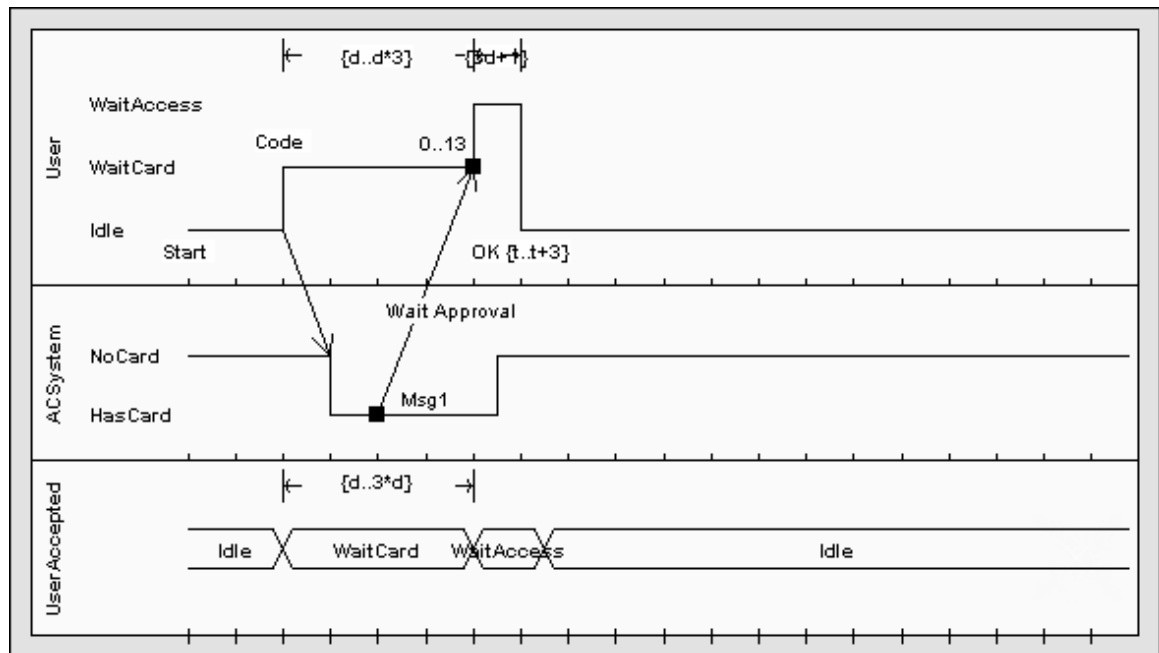
The dialog will consist of a set of transition points. Each transition point can be defined with the following properties:

| | |
|-------|--|
| Start | Defines the lifeline where the message originates. |
| End | Defines the lifeline where the message terminates. |

These are set by default when a message is created by dragging the cursor between two lifelines.

| | |
|----------------------|--|
| Start Time | Specifies the start time for a message. |
| End Time | Specifies the end time for a message. |
| Event | Describes the occurring event. |
| Time Constraint | Refers to the time taken to transmit a message. |
| Time Observation | Provides information on the time of a sent message. |
| Duration Constraint | Pertains to a lifeline's period at a particular state. The constraint could be instigated by that lifeline's receipt of a message. |
| Duration Observation | Indicates the interval of a lifeline at a particular state, begun from a message receipt. |

The following diagram shows the message configured by the above dialog snapshot.



Refer to figure 352 (UML 2.0 Superstructure, p. 454) and figure 351 (UML 2.0 Superstructure, p. 453).

Further Information

- [Message \(Timing Diagram\)](#)

5.4.19.4 Message Endpoint

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *message endpoint* element defines the termination of a state or value lifeline in a Timing diagram.

Common Usage Message Endpoint

- [Timing Diagram](#)

Further Information

- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 434) states:

"A *Stop* is an *EventOccurrence* that defines the termination of the instance specified by the *Lifeline* on which the *Stop* occurs."

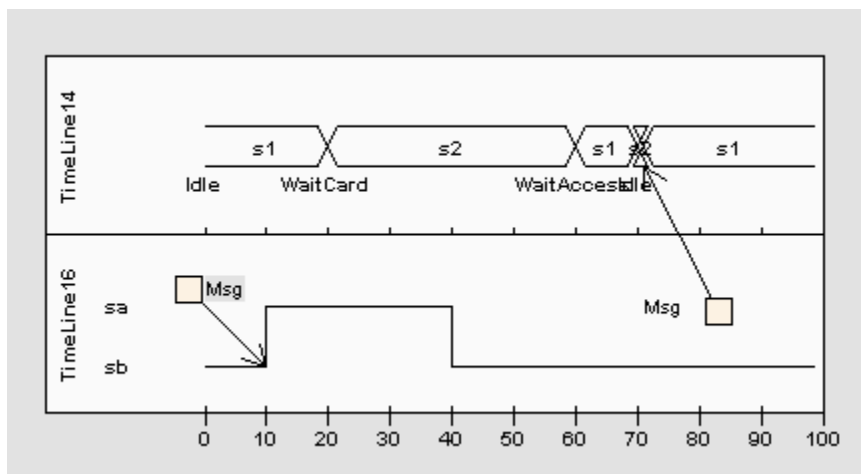
5.4.19.5 Message Label

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *message label* is an alternate way to denote messages between lifelines, which is useful for 'uncluttering' timing diagrams strewn with messages. To indicate a message between lifelines, draw a connector from the source lifeline into a message label. Next, draw a connector from another message label to the target lifeline. Note that the label names must match to reflect that the message occurs between the two message labels.

The below diagram illustrates how message labels are used to construct a message between lifelines.



Common Usage

- [Timing Diagram](#)

Further Information

- [Timing Diagram](#)

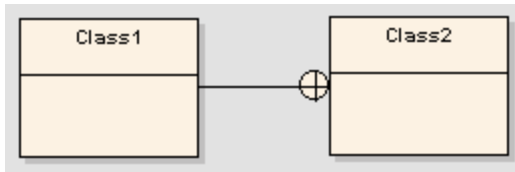
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 451) states:

"Labels are only notational shorthands used to prevent cluttering of the diagrams with a number of messages crisscrossing the diagram between Lifelines that are far apart. The labels denote that a Message may be disrupted by introducing labels with the same name."

5.4.20 Nesting

[Common Usage](#)



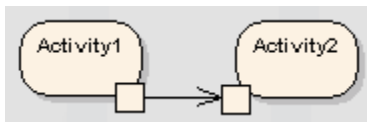
The *nesting* connector is an alternative graphical notation for expressing containment or nesting of elements within other elements. It is most appropriately used for displaying package nesting.

Common Usage Nesting

- [Package Diagram](#)

5.4.21 Object Flow

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Object flows are used in Activity diagrams and State Machines. When used in an Activity diagram, an object flow connects two elements, with specific data passing through it. Please refer to the [Further Information](#) section below to view [sample Activity diagrams](#) using object flows. With respect to State Machines, an object flow is a specification of a state flow or transition. It implies the passing of an object instance between elements at run-time.

You can insert an object flow from the State or Activity sections of the toolbox, or from the drop-down list of all relationships, located in the header toolbar. You can also modify a transition connection to an ObjectFlow by checking the ObjectFlow check box on the connection properties dialog.

See [ControlFlow](#) for information on setting up Guards and Weights on ObjectFlows.

Common Usage Object Flow

- [Activity Diagram](#)
- [Interaction Overview Diagram](#)

Further Information

- [Using Object Flows in Activity Diagrams](#)
- [Action Pin](#)
- [Object](#)

OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 12) states:

"A state in an activity diagram that represents the passing of an object from the output of actions in one state to the input of actions in another state."

The OMG UML specification (*UML 2.0 Superstructure*, p. 344) states:

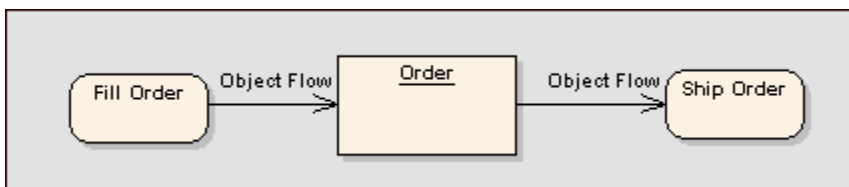
"An object flow is an activity edge that can have objects or data passing along it."

5.4.21.1 Using Object Flows in Activity Diagrams

[Further Information](#)

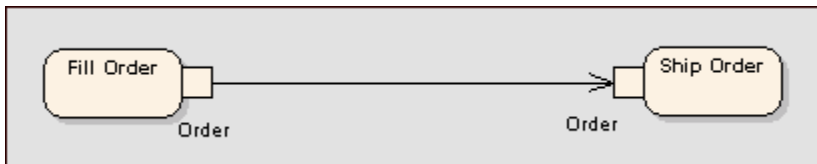
In Activity diagrams, there are several ways to define the flow of data between objects.

The below diagram depicts a simple object flow between two actions, Fill Order and Ship Order, both accessing order information.



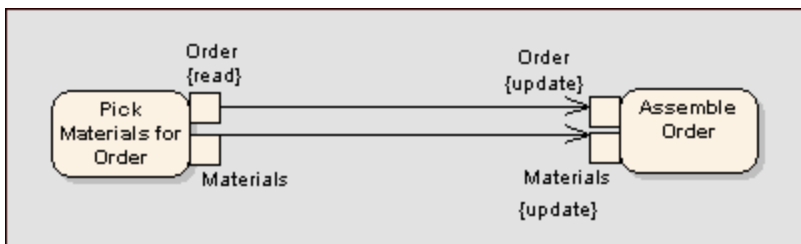
Refer to figure 271 (UML 2.0 Superstructure, p. 347).

This explicit portrayal of the data object, Order, connected to the activities by two object flows, can be refined by using the following format. Here, action pins are utilized to reflect the order.



Refer to figure 271 (UML 2.0 Superstructure, p. 347).

Below is an example of multiple object flows exchanging data between two actions.



Refer to figure 272 (UML 2.0 Superstructure, p. 347).

Selection and transformation behavior, together composing a sort of query, can specify the nature of the object flow's data access. Selection behavior determines which objects are affected by the connection. Transformation behavior might then further specify the value of an attribute pertaining to a selected object.

Selection and transformation behaviors can be defined by attaching a note to the object flow. To do this, right-

click on the object-flow and select *Attach Note or Constraint*. A dialog will indicate other flows in the diagram, to which you can select to attach the note, if the behavior applies to multiple flows. To comply with UML 2, preface behavior with the notation <<selection>> or <<transformation>>.



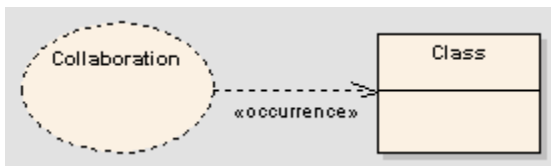
Refer to figure 268 (UML 2.0 Superstructure, p. 346).

Further Information

- [Object Flow](#)

5.4.22 Occurrence

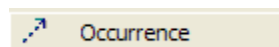
[Common Usage](#) | [OMG UML Specification](#)



An *occurrence* relationship indicates that a collaboration represents a classifier. An occurrence connector is drawn from the collaboration to the classifier.

Common Usage

- [Composite Structure Diagram](#)



Further Information

- [Collaboration](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 161) states:

"A dashed arrow with a stick arrowhead may also be used to show that a collaboration represents a classifier, optionally labelled with the keyword <<occurrence>>."

5.4.23 Package Import

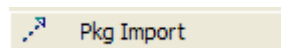
[Common Usage](#) | [OMG UML Specification](#)



A *package import* relationship is drawn from a source package to a package whose contents will be imported. Private members of a target package cannot be imported.

Common Usage

- [Package Diagram](#)



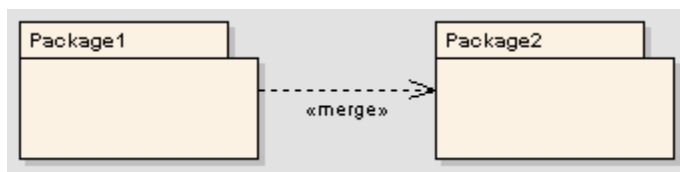
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 38) states:

"A package import is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace. Conceptually, a package import is equivalent to having an element import to each individual member of the imported namespace, unless there is already a separately-defined element import."

5.4.24 Package Merge

[Common Usage](#) | [OMG UML Specification](#)



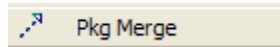
A *package merge* indicates a relationship between two packages whereby the contents of the target package are merged with those of the source package. Private contents of a target package are not merged. The applicability of a package merge addresses any situation of multiple packages containing identically-named elements, representing the same thing. A merging package will merge all matching elements across its merged packages, along with their relationships and behaviors. Note that a package merge essentially performs generalizations and redefinitions of all matching elements, but the merged packages and their independent element representations still exist and are not affected.

The package merge serves a graphical purpose in Enterprise Architect, but creates an ordered "Package" relationship applied to related packages (which can be seen under the *Link tab* in the package's *Properties* dialog). Such relationships can be reflected in XML exports or EA Automation Interface scripts for code generation or other MDA (Model Driven Architecture) interests.

Package merge relationships are useful to reflect situations where existing architectures contain functionalities involving like elements, which will be merged in a developing architecture. Merging doesn't affect the merged objects, and supports the common situation of product progression.

Common Usage

- [Package Diagram](#)

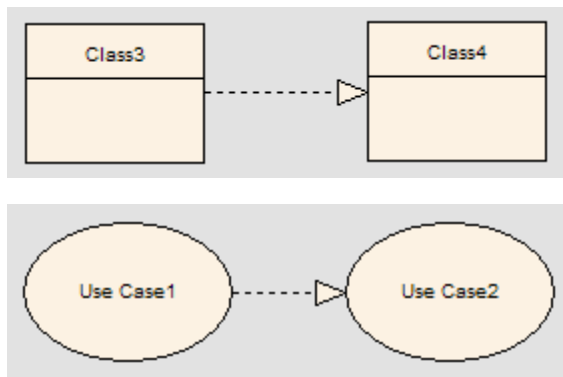
**OMG UML Specification**

The OMG UML specification (*UML 2.0 Superstructure*, p. 101) states:

"A package merge is a relationship between two packages, where the contents of the target package (the one pointed at) is merged with the contents of the source package through specialization and redefinition, where applicable. This is a mechanism that should be used when elements of the same name are intended to represent the same concept, regardless of the package in which they are defined. A merging package will take elements of the same kind with the same name from one or more packages and merge them together into a single element using generalization and redefinitions. It should be noted that a package merge can be viewed as a short-hand way of explicitly defining those generalizations and redefinitions. The merged packages are still available, and the elements in those packages can be separately qualified. From an XML point of view, it is either possible to exchange a model with all PackageMerges retained or a model where all PackageMerges have been transformed away (in which case package imports, generalizations, and redefinitions are used instead)."

5.4.25 Realize

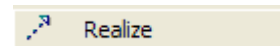
[Common Usage](#) | [OMG UML Specification](#)



The source object *implements* or *realizes* the destination. *Realize* is used to express traceability and completeness in the model - a business process or requirement is realized by one or more use cases which are in turn realized by some classes, which in turn are realized by a component, etc. Mapping requirements, classes, etc. across the design of your system, up through the levels of modeling abstraction, ensures the big picture of your system remembers and reflects all the little pictures and details that constrain and define it.

Common Usage

- [Use Case Diagram](#)
- [Component Diagram](#)

**OMG UML Specification**

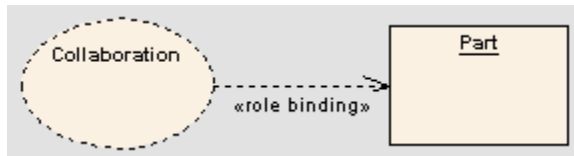
The OMG UML specification (*UML 2.0 Superstructure*, p. 110) states:

"A Realization signifies that the client set of elements are an implementation of the supplier set, which serves as the specification. The meaning of 'implementation' is not strictly defined, but rather implies a more refined or elaborate form in respect to a certain modeling context. It is possible to specify a mapping between the specification and implementation elements, although it is not necessarily

computable."

5.4.26 Role Binding

[Common Usage](#) | [OMG UML Specification](#)

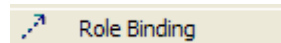


Role binding is the mapping between a collaboration occurrence's internal roles and the respective parts needed to implement a specific situation. The associated parts can have properties defined to enable the binding to occur, and the collaboration to take place.

A role binding connector is drawn between a collaboration and the classifier's fulfilling roles, with the collaboration's internal binding roles labeled on the classifier end of the connector.

Common Usage

- [Composite Structure Diagram](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 160) states:

"A mapping between features of the collaboration type and features of the classifier or operation. This mapping indicates which connectable element of the classifier or operation plays which role(s) in the collaboration. A connectable element may be bound to multiple roles in the same collaboration occurrence (that is, it may play multiple roles)."

5.4.27 Represents

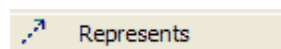
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



The *represents* connector indicates a collaboration is used in a classifier. The connector is drawn from the collaboration to its owning classifier.

Common Usage

- [Composite Structure Diagrams](#)



Further Information

- [Collaboration](#)

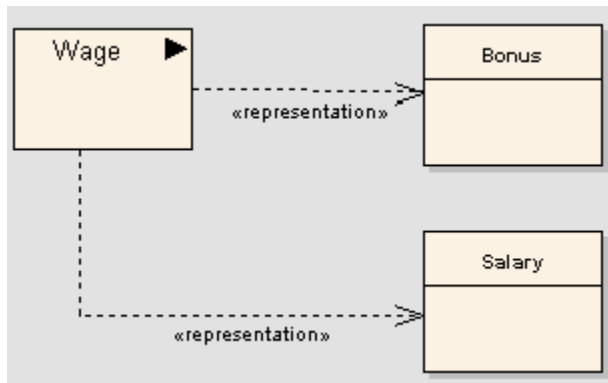
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 161) states:

"A dashed arrow with a stick arrowhead may be used to show that a collaboration is used in a classifier, optionally labelled with the keyword «represents»."

5.4.28 Representation

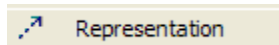
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



The representation relationship is a specialization of a dependency, linking InformationItem elements that represent the same idea across models. i.e. Bonus and Salary are both a Representation of the InformationItem Wage.

Common Usage

- [Activity Diagram](#)



Further Information

- [Information Item](#)
- [Information Flow](#)

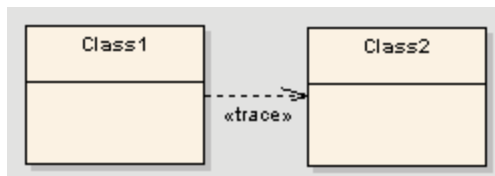
OMG UML Specification

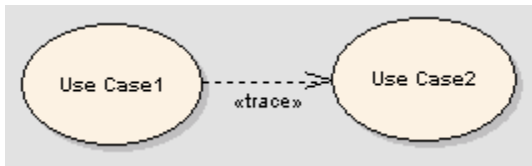
The OMG UML specification states:

"An information item is usually represented attached to an information flow, or to a relationship realizing an information flow."

5.4.29 Trace

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

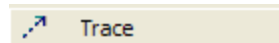




The *trace* relationship is a specialization of a dependency, linking model elements or sets of elements that represent the same idea across models. Traces are often used to track requirements and model changes. As changes can occur in both directions, the order of this dependency is usually ignored. The relationship's properties can specify the trace mapping, but the trace is usually bi-directional, informal and rarely computable.

Common Usage

- [Class Diagram](#)
- [Use Case Diagram](#)
- [Object Diagram](#)
- [Composite Structure Diagram](#)



Further Information

- [Dependency](#)

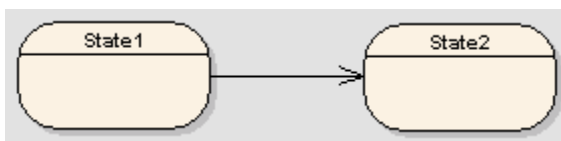
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

"A dependency that indicates a historical or process relationship between two elements that represent the same concept without specific rules for deriving one from the other."

5.4.30 Transition

[Common Usage](#) | [OMG UML Specification](#)



A *transition* defines the logical movement from one state to another. The transition can be controlled through the following properties:

General Constraints

Guard:

Effect is an Activity

Effect:

Trigger

Name:

Type:

Specification:

New Save Remove

Triggers

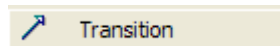
| Name | Type | Specification |
|------|------|---------------|
| | | |

| | |
|-----------------------|--|
| Guard | An expression that is evaluated after an event is dispatched, but before the corresponding transition is triggered. If the guard is true at that time, the transition is enabled; otherwise, it is disabled. |
| Effect is an Activity | Specifies that Effect is a Activity |
| Effect | Specifies an optional activity to be performed during the transition |
| Name | Name of Trigger |
| Type | Can be one of "Call", "Change", "Signal", "Time" |
| Specification | Specifies the event instigating the transition |
| New | Create a new trigger |
| Save | Save the current trigger |
| Remove | Remove the selected trigger from the list. |

Note: Fork and join segments can have neither triggers nor guards.

Common Usage

- [State Machine Diagram](#)



OMG UML Specification

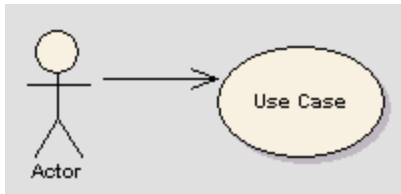
The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

"A relationship between two states indicating that an object in the first state will perform certain specified actions and enter the second state when a specified event occurs and specified conditions are satisfied."

On such a change of state, the transition is said to fire."

5.4.31 Use

[Common Usage](#) | [OMG UML Specification](#)



A *use* link indicates that one element requires another to perform some interaction. The Usage relationship does not specify how the target supplier is used, other than that the source client uses it in definition or implementation. A usage relationship is a sub-typed dependency relationship.

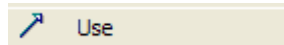
You are likely to use the use relationship mostly in Use Case diagrams to model how actors use system functionality (Use Cases), or to illustrate usage dependencies between classes or components.

Note: it is more usual (and correct UML) to have an [association connector](#) between an actor and a use case.

Note: To depict a usage dependency on a Class or Component diagram, draw a dependency connector. Right-click on the dependency, and select [Dependency Stereotypes | use](#).

Common Usage

- [Use Case Diagram](#)
- [Class Diagram](#)
- [Component Diagram](#)



OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 111) states:

"A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. In the metamodel, a Usage is a Dependency in which the client requires the presence of the supplier."

Part

6

6 Modeling with UML

What is UML?

UML is the *Unified Modeling Language*, a standard that defines the rules and notation for specifying software systems. The notation supplies a rich set of graphic elements for modeling object-oriented elements, and the rules say how those elements may be connected and used.

UML is not a prescriptive process for creating software systems - it does not supply a method or process, simply the language. You can therefore use UML in a variety of ways to specify and develop your software engineering project. Enterprise Architect supports many different kinds of UML elements (as well as some custom extensions). Together with the links and connectors between elements, these form the basis of the model.

In addition to the base UML elements, the modeling environment can be extended using UML Profiles. A Profile is a set of stereotyped and tagged elements that together solve some modeling problem or scenario. Examples are UML Profiles for modeling XML Schema or Business Process Modeling.

This section contains a list of the basic UML building blocks. Familiarity with these elements and their purpose will greatly assist your modeling tasks. The full meaning of each of the elements is beyond the scope of this user guide.

For more information, see the UML section at <http://www.omg.org> or any good book.

Modeling can be defined as the act of representing something, usually on a smaller scale or with reduced detail. Using EA, modeling can be more specifically described as the act of graphically representing a business process or software system. A model thus created can be used to emphasize a certain aspect of the system being represented and record, document and communicate its detail. A study of such a model can allow insight or understanding of the system.

Recommended Reading:

In addition to the UML Specification available from the OMG, two books which provide excellent introductions to UML are:

- *Schaum's Outlines: UML* by Bennet, Skelton and Lunn. Published by McGraw Hill. ISBN 0-07-709673-8
- *Developing Software with UML* by Bern Oestereich. Published by Addison Wesley. ISBN 0-201-36826-5

More Information

Modeling with UML:

- [Working with UML Diagrams](#)
- [Working with UML Elements](#)
- [Working with UML Connectors](#)
- [UML Stereotypes](#)
- [MDG Technologies](#)
- [UML Profiles](#)
- [UML Patterns](#)
- [Requirements Management](#)
- [Element Relationship Matrix](#)

See Also

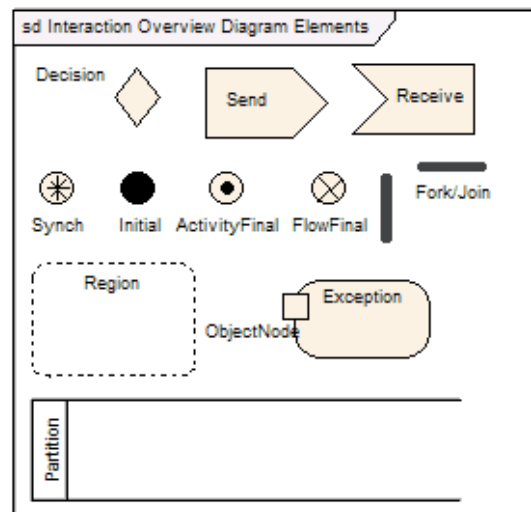
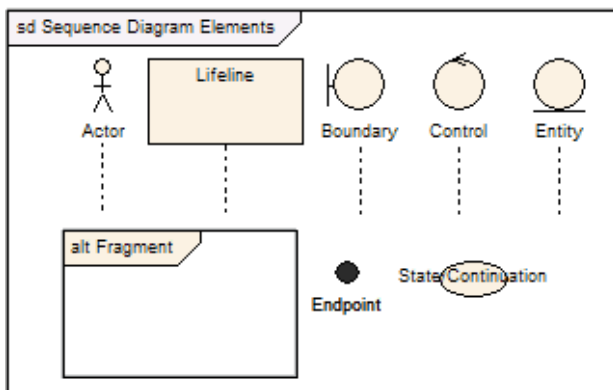
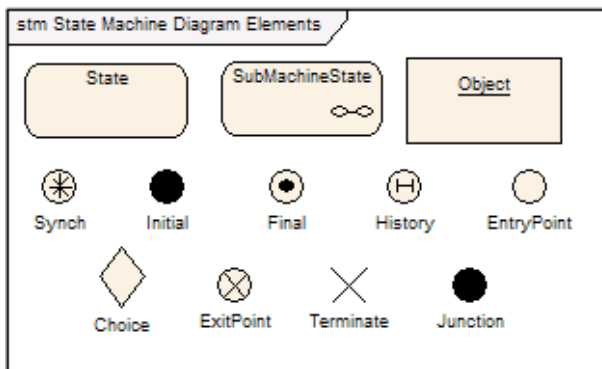
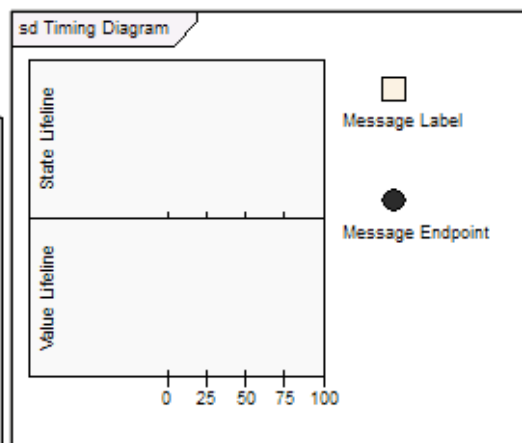
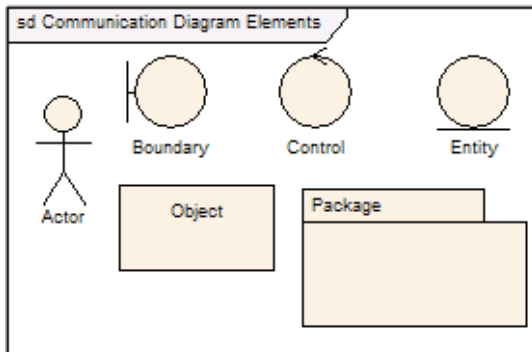
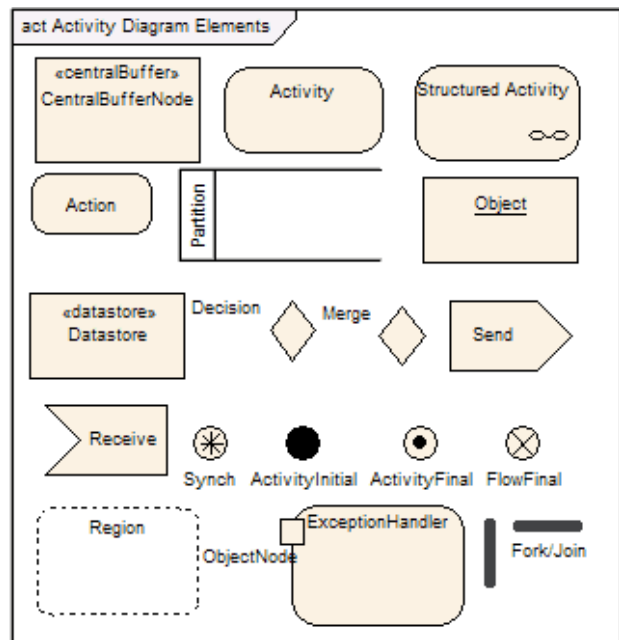
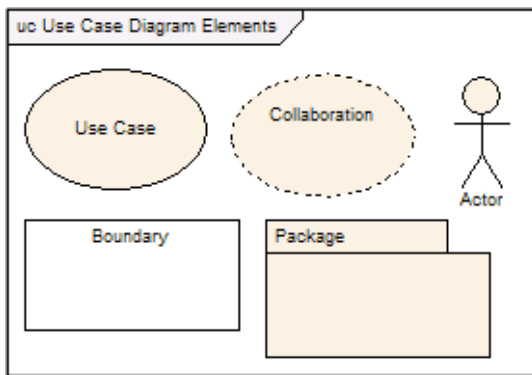
- [UML Diagrams](#)
- [UML Elements](#)
- [UML Connectors](#)

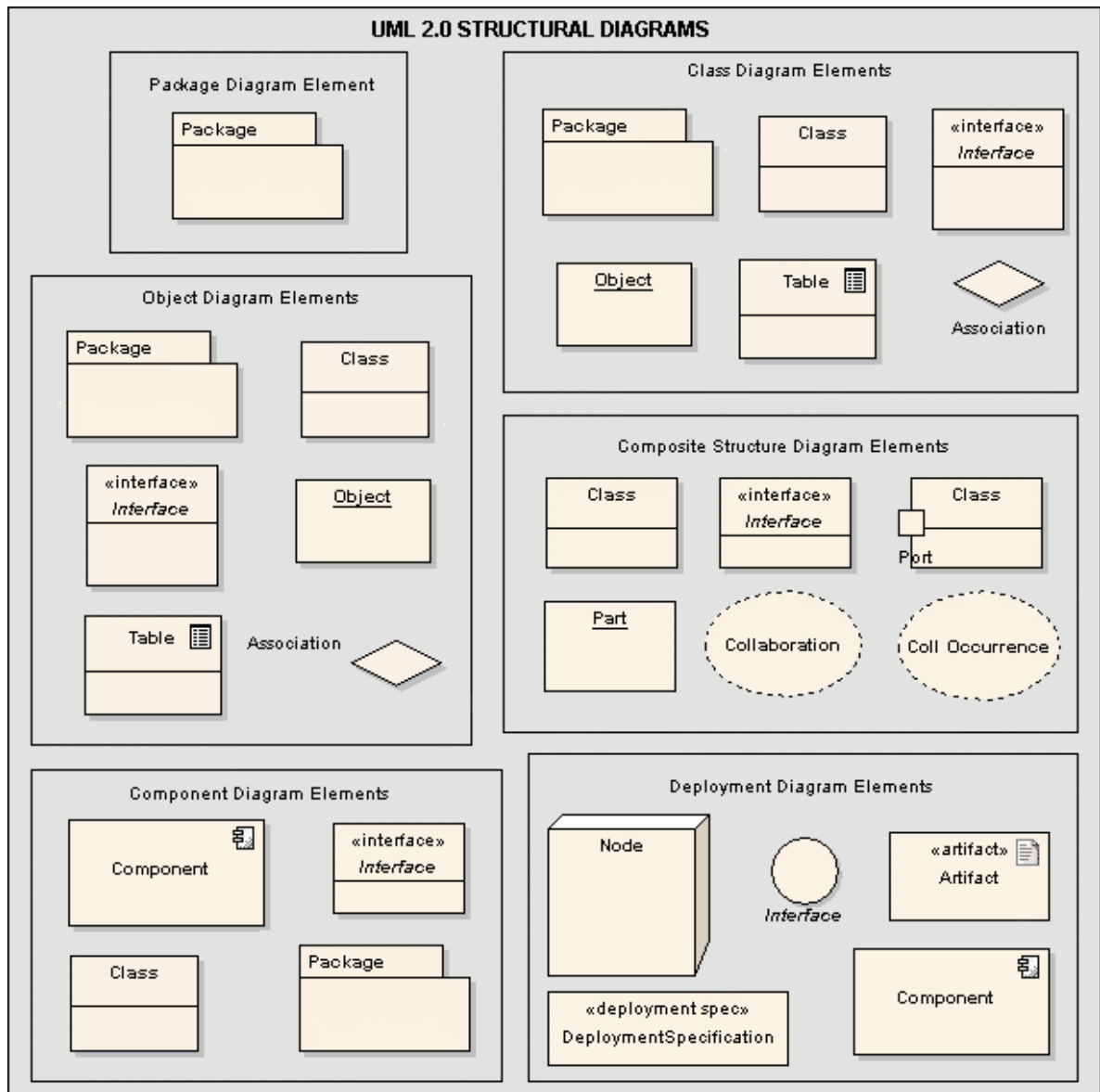
6.1 Working with UML Diagrams

UML Diagrams are collections of project elements laid out and inter-connected as required using Enterprise Architect supports several kinds of UML diagrams (as well as some custom extensions).

Together with the Enterprise Architect elements and connectors, these form the basis of the model. Diagrams are stored in packages and may have a parent object (optional). Diagrams may be moved from package to package.

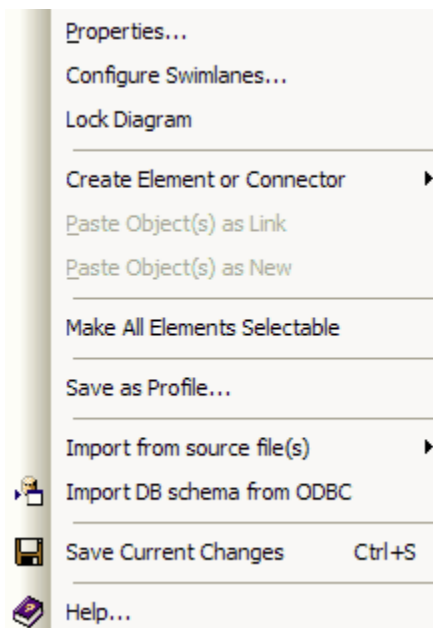
Basic UML diagram elements:





6.1.1 Diagram Context Menu

Right click on the diagram window when a valid diagram is selected to open the diagram context menu. Not all menu options shown below will appear on all diagram context menus.



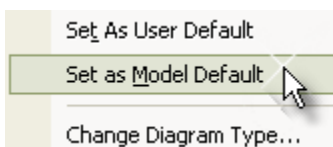
The diagram context menu allows you to:

- View the diagram properties dialog
- Protect a diagram from inadvertent changes (lock diagram)
- Insert various elements into a diagram (select from the following - Boundary, Note, Text, Diagram Notes, Hyperlink)
- Paste element(s) as a link or as new elements
- Make all the elements on the diagram selectable.
- Import source code (reverse engineer) - *not available in the Desktop edition*
- Import database tables from an ODBC data source - *not available in the Desktop edition*
- Save the current diagram
- View the EA Help files

6.1.2 Set the Default Diagram

A project may have a *default diagram*. If set, this diagram will load when EA first opens the model. It is often convenient to place hyperlinks to other diagrams and resources on the default diagram, thus creating a kind of Home Page for your model.

To set the currently active diagram as the model default, select *Set as Model Default* from the *Diagram* menu.



Tip: Once you have specified a default diagram, the 'Home' icon on the diagram toolbar will take you to that diagram.



6.1.3 Common Diagram Tasks

This section details some of the common tasks associated with managing diagrams:

- [New Diagrams](#)
- [Deleting Diagrams](#)
- [Diagram Properties](#)
- [Renaming Diagrams](#)
- [Copy Diagram Element](#)
- [Diagram Navigation Hotkeys](#)
- [Convert Linked Element to Local Copy](#)
- [Z Order Elements](#)
- [Copy Image to Disk](#)
- [Copy Diagram Image to Clipboard](#)
- [Change Diagram Type](#)
- [Open a Package](#)
- [Duplicate a Diagram](#)
- [Set Feature Visibility](#)
- [Insert Diagram Properties Note](#)
- [Autosize Elements](#)
- [Drop Elements from the Project Browser](#)
- [Pasting from the Project Browser](#)
- [Place Related Elements on Current Diagram](#)
- [Swimlanes](#)
- [Using the Image Manager](#)
- [Show Realized Interfaces for a Class](#)
- [Label Menu Section](#)

6.1.3.1 New Diagrams

To add a new diagram, follow these steps:

1. In the *Project Browser* window, select the appropriate package or element under which you wish to place the diagram.
2. Right click to open the context menu and select *Add | Add Diagram....*
3. Enter the name of the new diagram in the dialog provided and select the type of diagram you require.
4. Press *OK* to create your new diagram.

Alternatively, select *Add Diagram* from the *Project* menu.

Note: *the diagram type determines the default toolbar associated with the diagram and whether it can be set as a child of another element in the Project Browser (eg. a sequence diagram under a use case).*

See Also

- [UML Diagrams](#)

6.1.3.2 Deleting Diagrams

Warning: There is no Undo feature in Enterprise Architect for deleting diagrams, so be very sure that you wish to delete a diagram before you do.

Note: When you delete a diagram, you do not delete the elements on the diagram from the model.

Delete a Diagram

To delete a diagram from your model, follow the steps below:

1. In the Project Browser, right click on the diagram you wish to delete.
2. Select the *Delete '<diagram name>'* option.
3. Confirm you wish to delete the diagram by pressing *OK*.

6.1.3.3 Diagram Properties

You may set several properties of a diagram using the *Diagram Properties* dialog (shown below). Some influence the display and some are logical attributes that appear in the documentation.

There are several options for opening the *Diagram Properties* dialog for a given diagram:

- Select *Properties* from the *Diagram* menu to open the properties dialog for the currently active diagram.
- Right click on the required diagram in the Project Browser and select *Properties* from the context menu.
- Right click on the background of the open diagram and select *Diagram Properties* from the context menu.
- Double click in the background of the open diagram.

In the *Diagram Properties* dialog you may set various properties including name, author and version information, zoom factor, paper size and layout, diagram notes, and various appearance attributes. Once you have made any necessary changes, press *OK* to save and exit.

See the following topics for further information:

- [Set Appearance Options](#)
- [Document Options](#)
- [Scale Image to Page Size](#)
- [Set the Page Size](#)
- [Set Visible Class Members](#)

Name: My Diagram

Author: Stereotype:

Version: 1.0 Created: 23/06/2006

Zoom: 100 Modified: 26/06/2006 3:41:24 PM

Page Size: A4 Sheet, 210- by 297-millimeters

Landscape

Appearance Options

Use Stereotype Icons Show Page Border

Show Element Stereotypes Show Details on Diagram

Show Feature Stereotypes Show Relationships

Use Alias if Available Show Non-Navigable Ends

Show Additional Parents Show Sequence Notes

Show Visibility Indicators Show Collaboration Numbers

Show Table Owner

Highlight Foreign Objects

Show Element Property String

Show Feature Property String

Show Connector Property String

Visible Compartments

Attributes

Operations

Tags

Requirements

Constraints

Testing

Maintenance

Package Contents

Visible Class Members

Public

Protected

Private

Package

Property Methods

Show Parameter Detail:

Connector Notation:

Document Options

Exclude image from RTF Documents Document each contained element in RTF

Notes:

6.1.3.4 Renaming Diagrams

To rename a diagram, follow the steps below:

1. Open the *Diagram Properties* dialog by double clicking on the diagram background, or by selecting *Diagram|Properties* from the main menu at the top of the screen.
2. Enter the new *Name* for your diagram.
3. Press *OK* to save changes.

6.1.3.5 Copy Diagram Element

To copy a diagram element, follow the steps below:

1. Select the element(s) to copy.
2. For multiple elements, right click to open the context menu and select *Copy All Selected Objects to*

Clipboard. Alternatively, press *Ctrl+C*.

- For single elements, select the *Edit | Copy* command or alternatively press *Ctrl+C*.

Paste Diagram Element(s)

To paste diagram element(s), follow the steps below:

- Open the diagram to paste into.
- Right click on the diagram background to open the diagram context menu.
- Select whether to *Paste Object(s) as New* (completely new element) or *Paste Object(s) as Link* (reference to the existing element).

Note: *Diagram elements may be pasted as links or as new elements. Select the most appropriate action for your model.*

6.1.3.6 Diagram Navigation Hotkeys

The diagram hotkeys allow the user to quickly navigate and select elements within a diagram. The following table details the Hotkey combinations along with the functionality of the hotkeys.

| Hotkey Command | Description |
|--------------------------------|--|
| Arrow key, Element(s) selected | Move the selected element(s). |
| Arrow key, No element selected | Scroll around the diagram. |
| Esc key | Clears the current selection. |
| Tab key | Selects the first element in the diagram if none currently selected. |
| Shift + mouse click | Adds the clicked element to the current selection. |
| Ctrl + mouse click | Adds the clicked element to the current selection. |
| Ctrl + Shift + mouse move | Pans the diagram. |
| Alt + G | Selects the item in the Project View and gives it focus. |

6.1.3.7 Convert Linked Element to Local Copy

To convert a linked element to a local copy, follow the steps detailed below:

- Open the diagram with the linked element.
- Select the linked element and *right click* on the element to bring up its context menu.
- Click the *Convert Linked Element to Local Copy* menu option.
- The Element will be changed to a local copy and will be placed in the appropriate package

6.1.3.8 Z Order Elements

Z Order refers to an element's depth in the diagram hierarchy, and thus influences which elements appear in front of others and which appear behind.

Set the Z Order

To set the Z Order of an element, follow the steps below:

1. Select the element in the diagram view.
2. From the *Element* | *Z order* submenu, select the operation to perform (see below).
3. The element will be moved to the new position in the diagram hierarchy.

6.1.3.9 Copy Image to Disk

You can copy a diagram image to a disk file in the following formats:

- Windows bitmap (256 color bitmap)
- GIF image
- Windows Enhanced Metafile (standard metafile)
- Windows Placeable Metafile (older style metafile)
- PNG format
- JPG
- TGA

Copy a Diagram Image to File

To copy a diagram image to file, follow the steps below:

1. Open the diagram you wish to save.
2. From the *Diagram* menu, select *Save Image to File*. (Alternatively, press *Ctrl+T*).
3. When prompted, enter a name for the file and select an image format.
4. Press *OK*.

Note: EA will clip the image size to the smallest bounding rectangle that encompasses all diagram elements.

6.1.3.10 Copy Diagram Image to Clipboard

Diagram images may be copied onto the clipboard and pasted directly into MS Word or other applications.

Copy an Image to the Clipboard

To copy an image to the clipboard, follow the steps below:

1. Open the diagram you wish to copy.
2. From the *Diagram* menu, select *Save Image to Clipboard*. (Alternatively, press *Ctrl+B*).
3. Press *OK*.

The diagram has been copied to the clipboard and can now be pasted into compatible applications. You can set the clipboard format in the *Local Options* dialog (*Tools* | *Options*). Bitmap or Metafile format are supported.

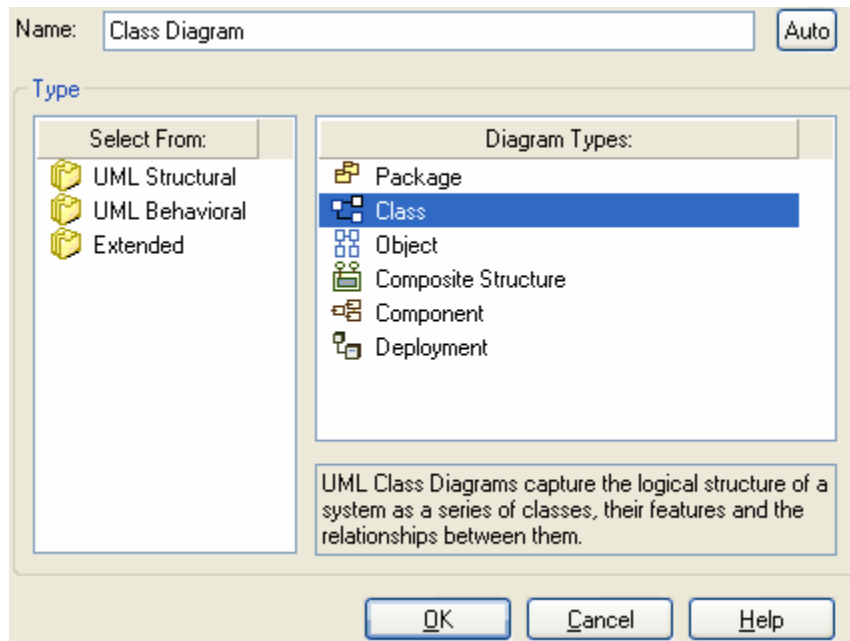
6.1.3.11 Change Diagram Type

You may change a diagram to another type if required. This is useful if you have either made a mistake in selecting the diagram type to begin with, or if the purpose and nature of a diagram changes during analysis.

Change a Diagram Type

To change a diagram type, follow the steps below:

1. Open the diagram you wish to change.
2. From the *Diagram* menu, select *Change Diagram Type...*
3. In the *Change Diagram Type* dialog, select the required diagram type.
4. Press *OK* to save changes.

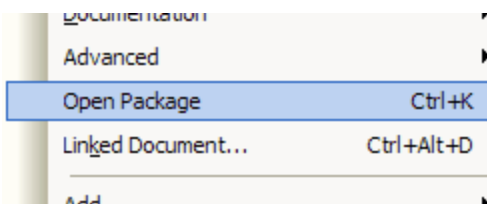


Note: Some diagram types will not transfer to others - for example you cannot change a Class Diagram into a Sequence diagram.

6.1.3.12 Open a Package**Open a Package**

To open a package, follow the steps below:

1. Open a diagram which shows the package you wish to open.
2. Right click on the package element to open the context menu.
3. Select *Open Package*. Alternatively, use the keyboard shortcut *Ctrl+K*.



Note: EA will find the default diagram and open it for you. If you haven't specified a default, this will be the first available diagram in the package (selected in alphabetical order).

6.1.3.13 Duplicate a Diagram

Enterprise Architect makes it easy to duplicate a complete diagram - either with links back to the original diagram elements, or with complete copies of all elements in the diagram.

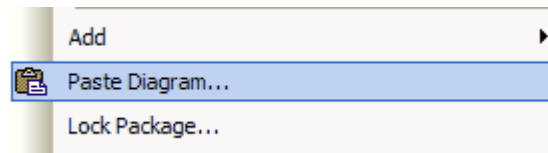
When you copy a diagram in 'shallow' mode, the elements in the new diagram are only *linked* to the originals - so if you change the properties of one, the other will reflect those changes. If you copy the diagram in 'deep' mode, then all elements are duplicated completely - so that changing an element on one does not affect the other.

Element position and size should be independent in both copy modes.

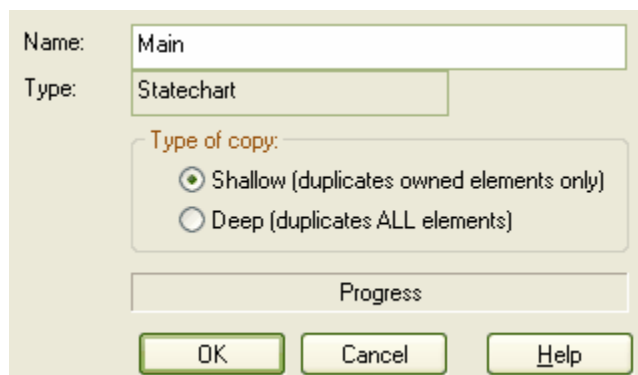
Duplicate a Diagram

To duplicate a diagram, follow the steps below:

1. In the Project Browser, select the diagram you wish to copy.
2. Select *Copy Diagram to Clipboard* from the right-click context menu.
3. Navigate to the package you wish to host the new diagram, and right click to open the context menu.
4. Select *Paste Diagram...*



5. In the *Copy diagram* dialog, enter a *Name* for the new diagram.
6. Check the *Type of copy* you require - either linked elements (shallow copy) or complete copies of the originals (deep copy).



7. Press *OK*.

EA will automatically create the new diagram, link or create new elements and arrange as in the original diagram. All links are also copied between diagram elements where appropriate.

6.1.3.14 Set Feature Visibility

Enterprise Architect allows you to set the visibility of attributes and operations on a per class basis / per diagram basis or the visibility of attributes and operations on a package diagram. For example you can hide all protected attributes, or all private operations or any other combination of attributes and operations. The visibility you set will apply only to the current diagram - so a class may appear in one diagram with all elements displayed - and in another its elements may be hidden.

It is possible to show inherited attributes, operations, requirements, constraints and tagged values for elements that support those features. When EA displays inherited features, it creates a merged list from all generalized parents and from all realized interfaces. If a child class redefines something found in a parent, the parent feature is omitted from the Merge List.

Tip: To show features for element types that do not have visible compartments, such as use cases and actors, right-click the diagram object and click "Advanced Settings | Use Rectangle Notation".

Customize Feature Visibility

To customize feature visibility, follow the steps below:

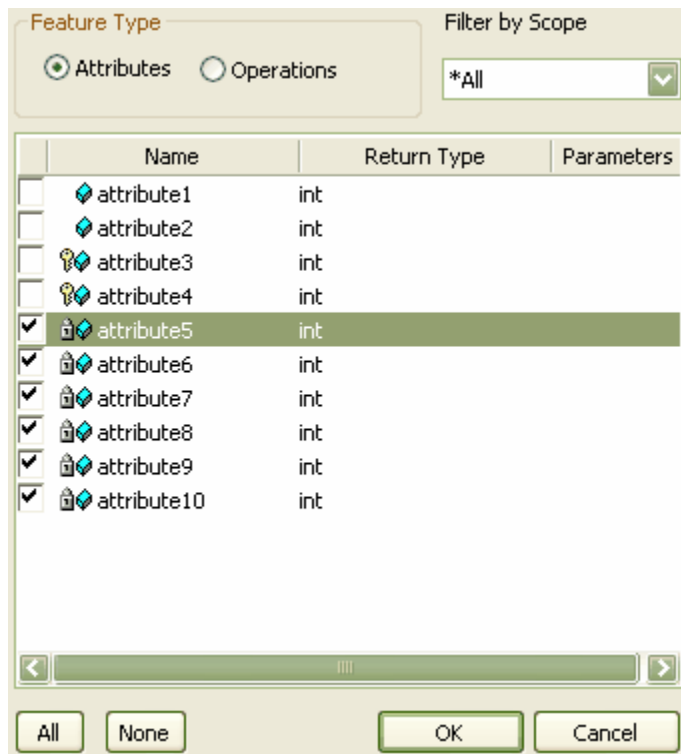
1. From the *Element* menu, select *Set feature visibility*. Ctrl+Shift +Y is a convenient keyboard shortcut -OR- use right click on an element in a diagram to bring up its context menu. Then mouse over the *Element Feature* menu item and select *Specify Feature Visibility*.
2. In the *Set feature visibility* dialog, check the features you want visible and clear those you do not. Also set whether EA should display inherited features as well as directly owned ones.

3. Press **OK** to save changes.

EA will redraw the diagram with the appropriate level of feature visibility.

Suppress Features in Diagram

Enterprise Architect allows you to set the visibility of attributes and operations - where shown - for a class on a per diagram basis. You can hide attributes and operations by scope, or by clicking the custom button you are able to hide attributes and operations individually. The visibility you set will apply only to the current diagram - so a class may appear in one diagram with all elements displayed - and in another its elements may be hidden.

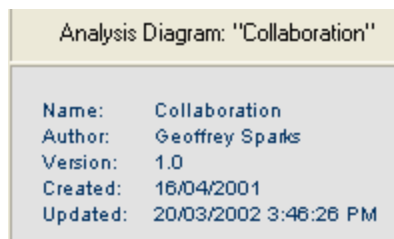
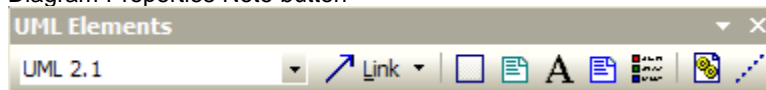


6.1.3.15 Insert Diagram Properties Note

Properties of a diagram may be displayed on screen within a custom text box. You can move this text box around and change its appearance. You cannot change what the text box says.

The Diagram Properties Note button is located on the *UML Elements* toolbar.

Diagram Properties Note button



6.1.3.16 Autosize Elements

You can autosize a group of elements in a diagram to a best fit.

Autosize a Group of Elements

To autosize a group of elements, follow the steps below:

1. Select all the elements you wish to resize (**Ctrl+A** selects all).
2. Use the right click context menu option **Autosize all selected**. Alternatively, press **Alt+Z**.

EA will resize elements where necessary.

Note: Not all elements will resize - some like Events and Use Cases remain the same

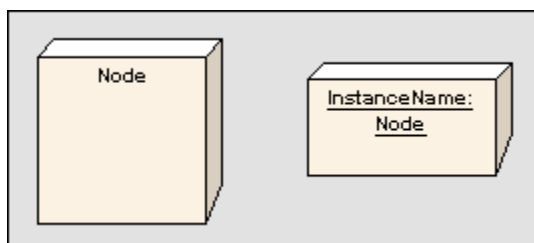
Note: The minimum size EA will set is the default size for the element - usually around 100x100 - but different for some elements.

6.1.3.17 Drop Elements from the Project Browser

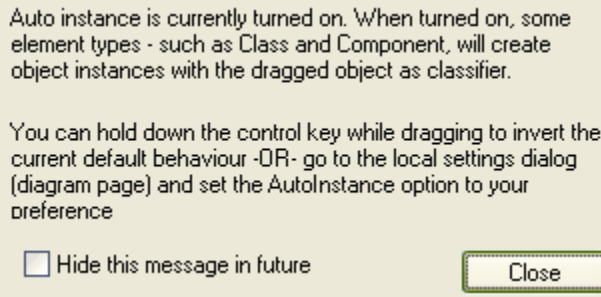
When you drag an element from the Project Browser onto a diagram, for some elements there are two possible paste options. Elements that are classifiers and support instances of themselves at runtime may be dropped as either a link to the classifier itself, or as a new instance of the classifier.

The example below shows a linked element on the left (a Node) and an instance of the Node on the right. Note that the Node instance is drawn like a simple element with the `:<ElementName>` displayed. If you name your instance it will display `<InstanceName>:<ElementName>`

If you are working on instance diagrams - such as a collaboration diagram, you will often want to quickly drag and drop classes and elements from the browser onto a diagram as an instance of the classifier. If you are working in class diagrams, you may prefer to drop links to the classifier itself. There are a couple of settings available to simplify this process.

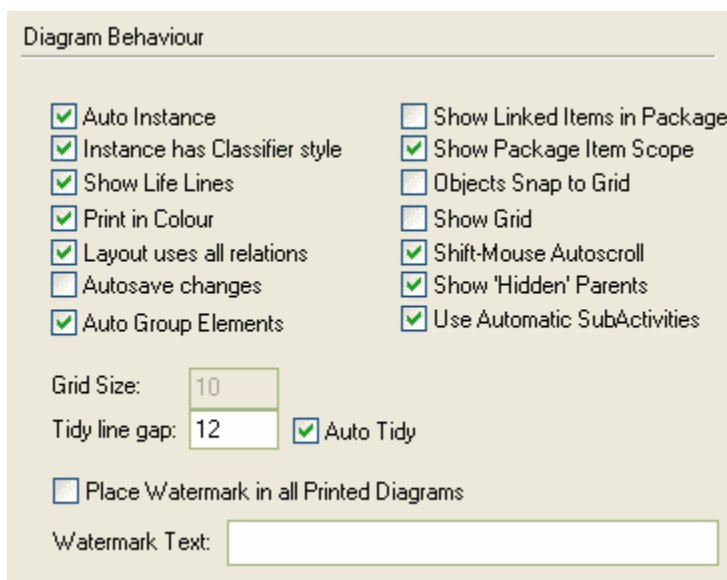


Note that EA will display a warning about this behavior (see below), indicating what is happening. Check the **Hide this message in future** option to prevent seeing this dialog.



There are two important things to remember here:

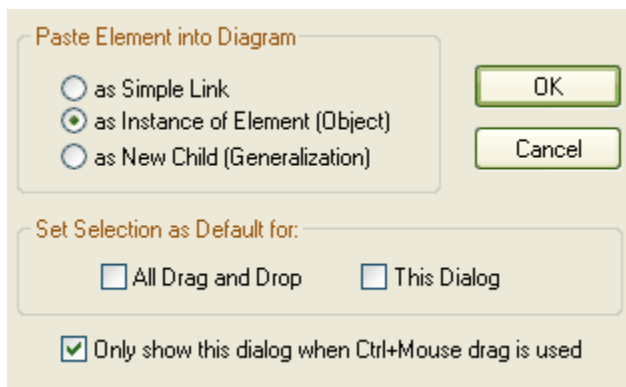
- Hold down the Ctrl key while dragging and dropping an element into a diagram (see [Pasting from the Project Browser](#)).
- Hold down either the Ctrl key or the Shift key to select multiple elements into a diagram (See [Pasting Multiple Elements from the Project Browser](#)).
- To change the default behavior, go to the Local Options dialog (Tools | Options), and select Diagram | Behavior. To enable or disable Auto Instance, check or clear the Auto Instance option (shown below). Remember - the Ctrl key will invert the current default.



6.1.3.18 Pasting from the Project Browser

You can paste an element from the Project Browser into the current diagram.

When you drag an element from the Project Browser onto the current diagram with the **Ctrl** key held down, EA will prompt you to select the type of paste action to carry out.



Three options are available:

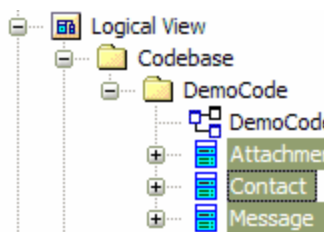
1. Paste the element as a link. In this case the element will appear in the current diagram as a simple reference back to the original source element. Changes to the element in the diagram will affect all other links to this element.
2. Paste as an instance of the element. If the element can have a classifier (eg. an Object, Sequence instance, Node instance etc.) you can drop the element as an instance of the source element, with the classifier pre-set to the original source. This is useful when creating multiple instances of a class in a sequence diagram, or in a Collaboration diagram.
3. Create as a child of the source element. This will automatically create a new class - which you will be prompted to name - and create a Generalization link back to the source. This is very useful when you have a class library or framework from which you inherit new forms (eg. a Hashtable may be pasted as "MyHashtable" and automatically become a child of the original Hashtable). Used with the Override parents operations and features, this is a quick way to create new structures based on frameworks like the Java SDK and the .NET SDK.

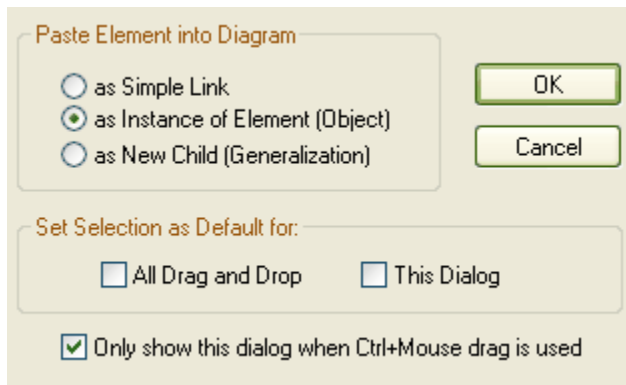
Note: In order to make use of the *Only show this dialog when Ctrl + Mouse drag is used* the *Auto Instance* check box must be selected in the *Tools|Options|Diagram|Behavior* dialog window.

6.1.3.18.1 Pasting Multiple Items from the Project Browser

You can paste multiple elements from the Project Browser into the current diagram.

To select multiple elements, use the mouse to select items from the Project Browser and either hold down the *Ctrl* key to add a single item to the selection of multiple elements one at a time, or hold down the *Shift* key to perform a selection of all of the elements between the first selected item in the Project Browser tree and the last item selected. You can then drag the selected elements from the Project Browser onto the current diagram with the *Ctrl* key held down, EA will prompt you to select the type of paste action to carry out.





Three options are available:

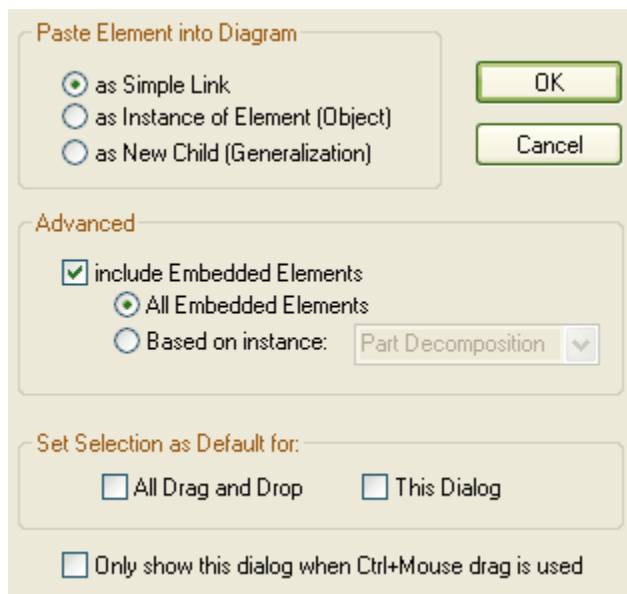
1. Paste the element as a link. In this case the element will appear in the current diagram as a simple reference back to the original source element. Changes to the element in the diagram will affect all other links to this element.
2. Paste as an instance of the element. If the element can have a classifier (eg. an Object, Sequence instance, Node instance etc.) you can drop the element as an instance of the source element, with the classifier pre-set to the original source. This is useful when creating multiple instances of a class in a sequence diagram, or in a Collaboration diagram.
3. Create as a child of the source element. This will automatically create a new class - which you will be prompted to name - and create a Generalization link back to the source. This is very useful when you have a class library or framework from which you inherit new forms (eg. a Hashtable may be pasted as "MyHashtable" and automatically become a child of the original Hashtable). Used with the Override parents operations and features, this is a quick way to create new structures based on frameworks like the Java SDK and the .NET SDK.

Note: In order to make use of the *Only show this dialog when Ctrl + Mouse drag is used* the *Auto Instance* check box must be selected in the *Local Options | Diagram Behavior* dialog window. To access this items go to the *Tools* menu and select *Options*.

6.1.3.18.2 Pasting Composite Elements

Several additional options are available to the user when pasting composite elements from one diagram to another.

When you drag a composite element from the Project Browser onto the current diagram with the *Ctrl* key held down, EA will prompt you to select the type of paste action to carry out with the composite element.



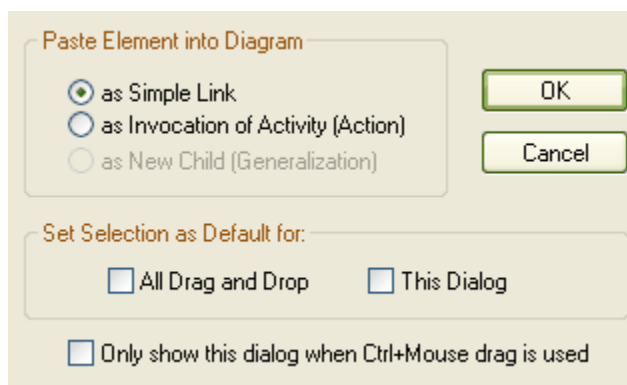
Two Advanced options are available for pasting composite elements, this requires that the *include Embedded Elements* checkbox be selected:

1. The *All Embedded* elements option, this will paste all of the composite elements embedded elements.
2. The *Based on instance* option, this will paste only the elements contained in a specific in an instance of the composite element. Select the appropriate instance from the drop down menu.

6.1.3.18.3 Pasting Activities

You can paste an activity from the Project Browser into the current diagram.

When you drag an activity from the Project Browser onto the current diagram with the *Ctrl* key held down, EA will prompt you to select the type of paste action to carry out.



Two options are available:

1. Paste the activity as a link. In this case the activity will appear in the current diagram as a simple reference back to the original source activity. Changes to the activity in the diagram will affect all other

links to this activity.

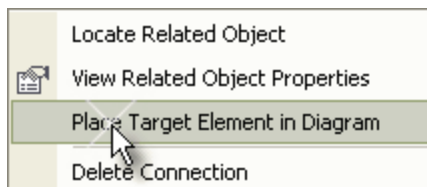
2. Paste as an invocation of the activity.

Note: In order to make use of the *Only show this dialog when Ctrl + Mouse drag is used* the *Auto Instance* check box must be selected in the *Local Options | Diagram Behavior* dialog window. To access this items go to the *Tools* menu and select *Options*.

6.1.3.19 Place Related Elements on Current Diagram

If you wish to find and place related elements on the current diagram, use the Relationships tab on the Properties window.

Right click on any link in the list to open the context menu



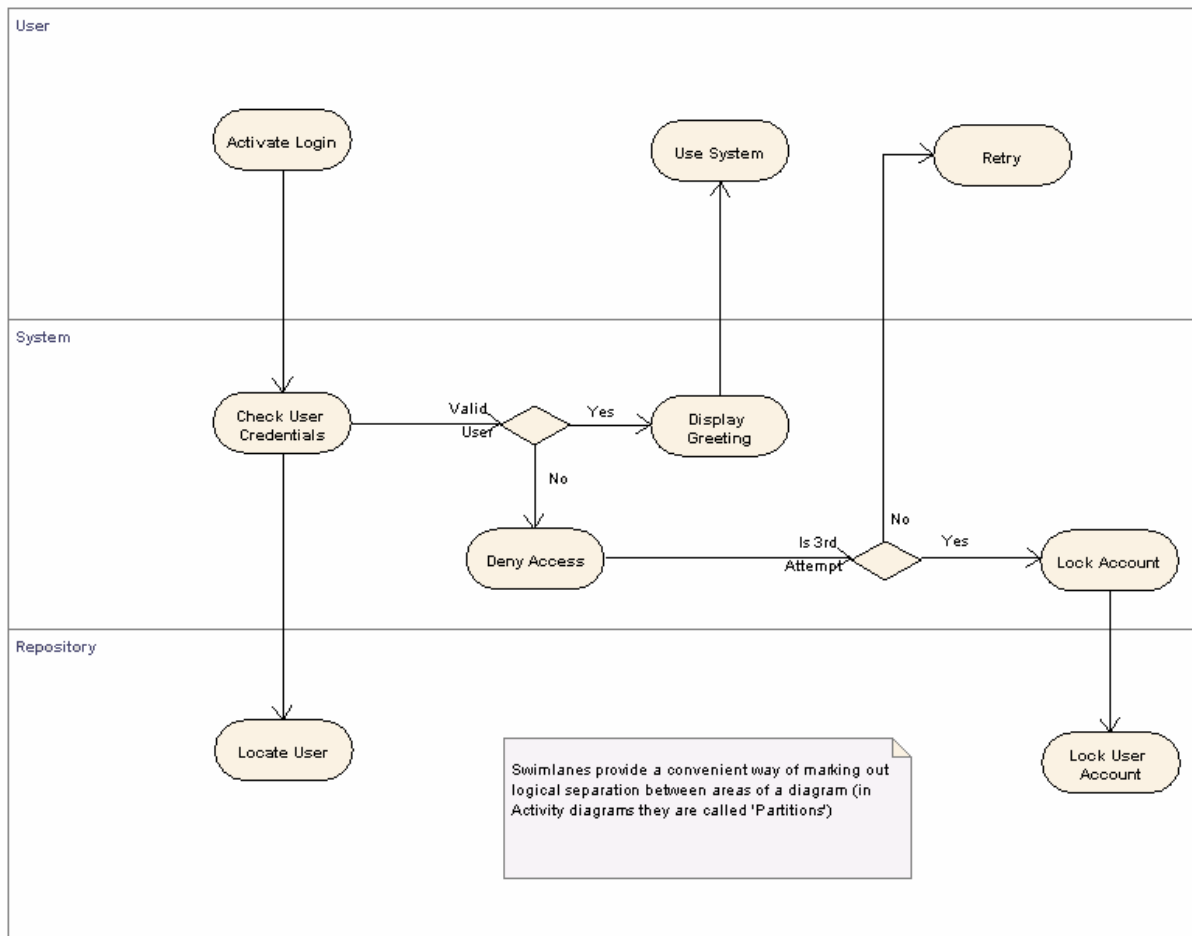
If an element is not present in the current diagram, the context menu will contain the *Place Target Element in Diagram* option. This is useful when you are building up a picture of what an element interacts with - especially when reverse engineering an existing code base.

Select the *Place Target Element in Diagram* option. Move the mouse cursor to the desired position in the diagram and left click to place the element. Alternatively, press the *Esc* key to cancel the action.

| Relationships | | | | |
|---------------|------------------|--------|------------|------|
| Relation | Target | Type | In Diagram | Role |
| ControlFlow | Import Options | Object | Yes | |
| ControlFlow | Import Options | Object | Yes | |
| ControlFlow | Code Engineering | Object | Yes | |
| ControlFlow | Code Engineering | Object | Yes | |

6.1.3.20 Swimlanes

EA diagrams support Swimlanes for all diagram types. Swimlanes are vertical or horizontal bands in a diagram that divide the diagram into logical areas or *partitions*. In the example below the activities relating to particular entities within the model (eg. the User, or the back end Repository) are placed within a containing swim lane to indicate their association.



To manage swimlanes, use the *Diagram Swimlanes* dialog, accessed by selecting *Configure Swimlanes...* from the *Diagram* menu.

This dialog allows you to set the orientation (vertical or horizontal), line color and width of the swimlanes, and also specify font color, bold font, hide names, hide the classifier and show the name in the title bar. You may also lock the swimlanes to prevent further movement. Use the *New*, *Modify* and *Delete* buttons to change aspects of the selected swim lane. Use the up and down buttons to switch the order of swimlanes within the diagram.

The screenshot displays two panels from the Enterprise Architect 6.5 User Guide. The top panel, titled "General", contains settings for diagram orientation, line color, font color, line width, and various checkboxes. The bottom panel, titled "Swimlanes", features a table with columns for Name, Classifier, and Partition, and buttons for New, Modify, and Delete.

General

Orientation: Vertical Hide Names

Line Color: Locked

Font Color: Bold Font

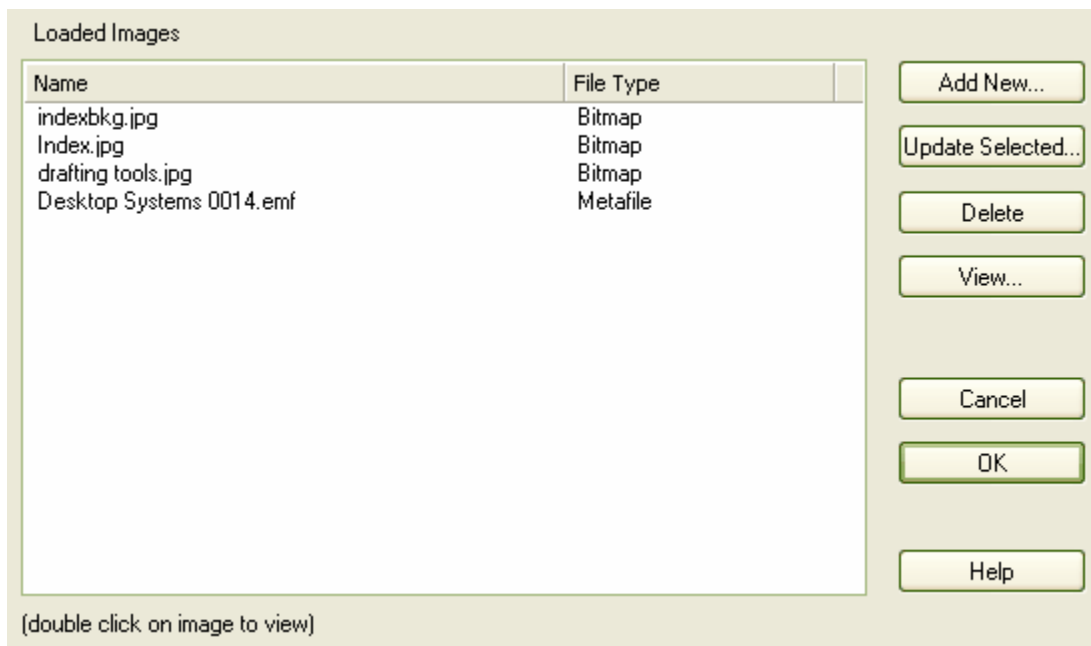
Line Width: 1 Hide Classifier Show Names in Title Bar

Swimlanes

| Name | Classifier | Partition |
|------------|------------|-----------|
| User | | 1 |
| System | | 2 |
| Repository | | 3 |

6.1.3.21 Using the Image Manager

The Image Manager allows users of EA to place alternate images in diagrams rather than standard UML elements. Often you may wish to place a custom background image on a diagram - or have a UML element display a custom image (eg. a Router or PC).



The following options are available when using the Image Manager dialog:

| Dialog Button | Description |
|-----------------|---|
| Add New | If you don't have images loaded, use the <i>Add New</i> button to search for and import another image. Images may be in BMP, PNG, EMF, WMF, TGA, PCX or JPG format. Internally they are always stored in PNG format to conserve space. |
| Update Selected | You may highlight an image and update the associated image using the <i>Update Selected</i> button. |
| Delete | If you delete an image you are first warned about how many elements use the image. If you continue, those elements will have the information about the associated image deleted and they will revert back to their previous appearance. |
| View | To preview an image, select the image name in the <i>Loaded Images</i> section of the dialog then press the <i>View</i> button or double click on the image name. |
| Cancel | The <i>Cancel</i> button closes the image manager |
| OK | Confirms the selection of the alternate image for the element selected in the diagram. |
| Help | Opens the help file to this topic. |

See Also

- [Select Alternate Image](#)
- [Create Custom Diagram Background](#)
- [Import Image Library](#)

6.1.3.21.1 Select Alternate Image

By using the Image Manager a custom image may be used for elements on diagrams. To perform this operation use the following procedure:

1. Right click on the element within the diagram to bring up its context menu, mouse over the *Appearance* item and from the submenu choose the *Select Alternate Image* option, or select the element in the diagram and use the *Ctrl + Shift + W* hotkey combination.
2. Use the *Image Manager* dialog to select an appropriate image as the alternate for the element. For more information regarding the use of the Image Manager see the [Using the Image Manager](#) topic.
3. Press the *OK* button when have selected the desired image.

Note: If you are creating many elements of the same type that will have this particular image, then you should use a [custom stereotype](#) with an associated metafile.

6.1.3.21.2 Create Custom Diagram Background

By using the Image Manager a custom, non-tiled background may be created for diagrams. To perform this operation use the following procedure:

1. Create a Boundary object from the *Use Case* section of the [UML Toolbox](#), do not use the Boundary object from the other sections of the UML toolbox.
2. Stretch the Boundary to a size which will be able to contain all of the elements that you intend to place on the diagram and drag it to the edges of the diagram workspace.
3. Right click on the Boundary element and from the context menu mouse over the *Z-Order* menu item, select *Send to Bottom* from the *Z-Order* submenu, this will ensure that the boundary object is not displayed in front of any other element in the diagram.
4. Press the *Ctrl + Shift + W* hotkey combination or right click on the boundary element to bring up the elements context menu, mouse over the *Appearance* menu item and from the sub menu choose the *Select Alternate Image* option.
5. Use the *Image Manager* dialog to select an appropriate image as the diagram background, ensure that the image size is large enough to span the desired size of the diagram background. For more information regarding the use of the Image Manager see the [Using the Image Manager](#) topic.
6. Press the *OK* button when have selected the desired image.

6.1.3.21.3 Import Image Library

Using the Image Library allows users of EA to create attractive diagrams with custom images. Sparx Systems now offers the a bundled clip art collection of UML based images as an Imported Image Library available from www.sparxsystems.com.au/resources/image_library.html. Image libraries allow the user to import a collection of images into the Image Manager in one process.

Note: Images contained within the Image Library are copyright of Sparx Systems and are only available for use when used in conjunction with EA and supplied on the basis that they will not be used under any other circumstance.

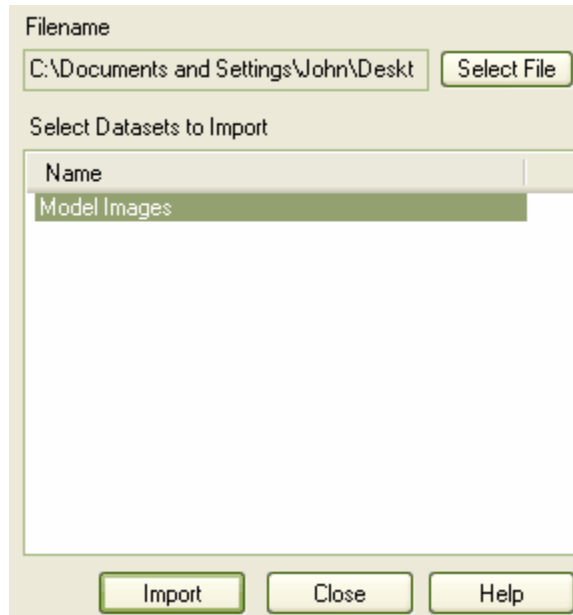
Importing an Image Library

To import an Image Library users will need a suitable Image Library file. To import the Image Library, follow the steps below:

1. Download the Image Library from www.sparxsystems.com.au/resources/image_library.html.
2. From the Tools menu select the Import reference data option. The Import Reference Data dialog will

open.

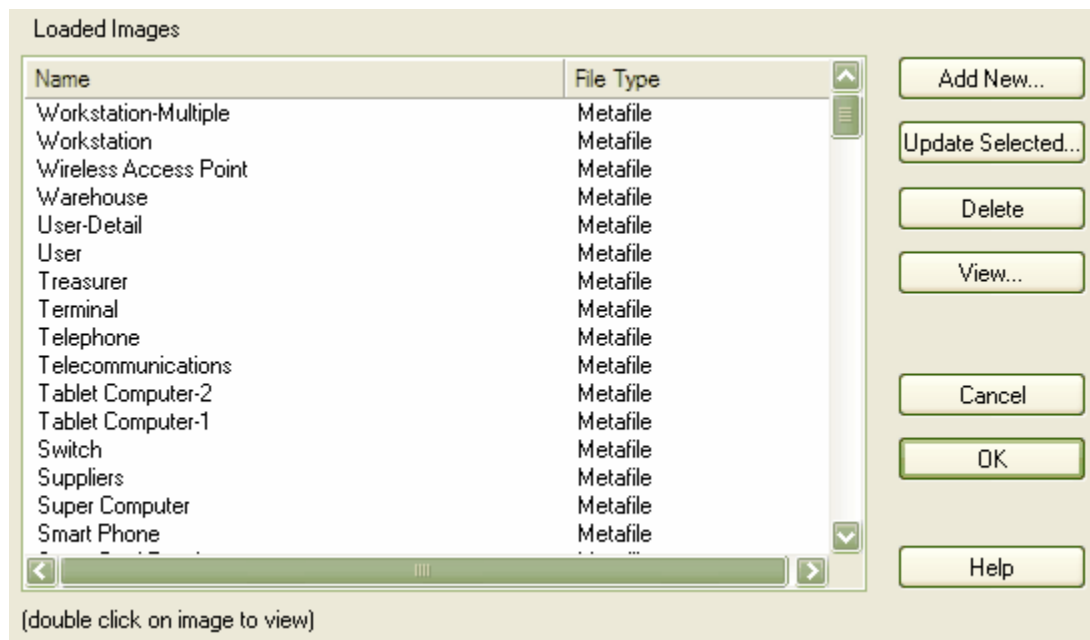
3. Locate the XML Image Library file to import using the Select File button, The file name will be ImageLibrary.xml in the directory which you saved the file to.
4. Select the data set containing the Image Library. Then press the Import button.



Using the Image Library

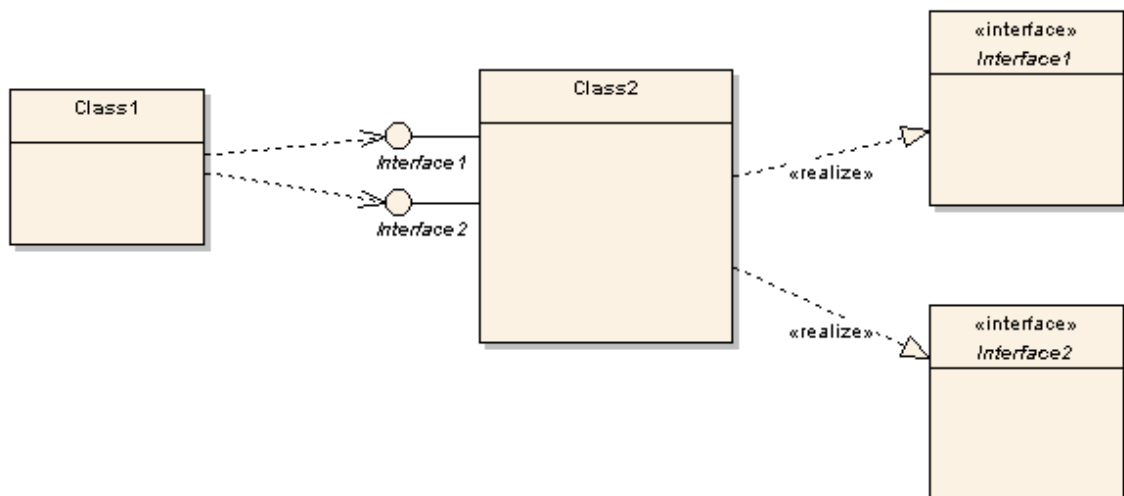
To use the images contained within the Image Library use the following Instructions:

1. Create a diagram that you wish to associate with the images contained in the Image Library.
2. Select the element that you wish to change from the default appearance to one of the images contained from within the library.
3. Press the *Ctrl + Shift + W* hotkey combination or right click the selected element to bring up its context menu and then from the *Appearance* submenu choose the *Select Alternate Appearance* option.
4. From the *Image Manager* dialog select the appropriate image by highlighting the image from the *Name* field and then press the *OK* button.



6.1.3.22 Show Realized Interfaces for a Class

You can display each interface directly realized by a class as a "lollipop" style interface node which protrudes from the left-hand side of the class. Connectors can be directly attached to the node, indicating usage of the interface part of the class or component. See the example below:



In this example, Class2 realizes Interface1 and Interface2. This is represented by the interface nodes protruding from the left hand side of the class. Class1 is dependent on these two interfaces, which is shown by the dependency arrows linking to the nodes.

To show nodes for the interfaces a class realizes as in the above diagram, right click on the class and select *Show Realized Interfaces*. This setting will only apply to the selected class, and can be changed at any time.

6.1.3.23 Label Menu Section

The Labels menu associated with connectors contains the following options:

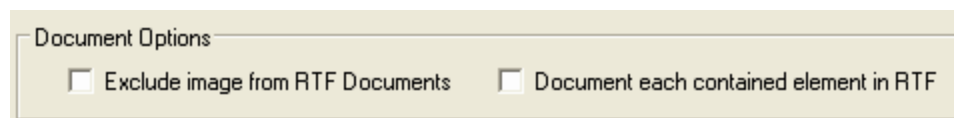
| Menu Option | Description |
|------------------|--|
| Set Label Color | Enables the user to specify a color for the label. |
| Hide Label | Hide the label, to unhide the label use the Set Label Visibility option. |
| Bold | Sets the label font to bold. |
| Text Alignment | Aligns the text within the label text area, the options available from the submenu allows the user to specify left, center and right alignment. |
| Label Rotation | The submenu allows the label to be orientated in the horizontal or vertical planes, with the vertical plane offering the option of clockwise are anti clockwise positioning. |
| Direction | Sets a small arrow at the end of the label pointing wither to the label source or destination dependent upon selection from the available options. |
| Default Position | Moves the label to the default location. |
| Default Color | Sets the label color to the initial default color. |

The Labels menu associated with embedded elements contains the following options

| Menu Option | Description |
|------------------|--|
| Set Label Color | Enables the user to specify a color for the label. |
| Hide Label | Hide the label, to unhide the label use the show label option in the Embedded Elements context menu. |
| Bold | Sets the label font to bold. |
| Text Alignment | Aligns the text within the label text area, the options available from the submenu allows the user to specify left, center and right alignment. |
| Label Rotation | Allows the label to be orientated in the horizontal or vertical planes, with the vertical plane offering the option of clockwise are anti clockwise positioning. |
| Default Position | Moves the label to the initial default location. |
| Default Color | Sets the label color to the default color. |

6.1.3.24 Document Options

This section refers to options (for a particular diagram) used when generating RTF reports from the [Diagram Properties](#) dialog.



These two options set the following:

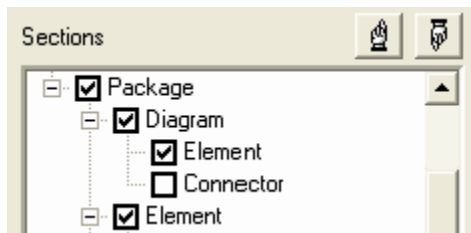
Exclude image from RTF Documents

Selecting this will exclude the image of the current diagram from any RTF reports.

Document each contained element in RTF

This sets the RTF generators to include details of elements that are defined externally to the package for this diagram. This allows the [option](#) to display these elements. The relevant section in the generator must be enabled before this section of the report will be generated.

To enable this section in your document generation templates, you must edit your template ([Project | Documentation | Rich Text Format Report | Manage Templates](#)).



Under Sections on the left-hand side of the editor window, check the box *Package | Diagram | Element*.

Note: *Legacy RTF Generator*

If using the [Legacy RTF generator](#), this section will be included after checking the *Document each contained element in RTF*. No further steps are required.

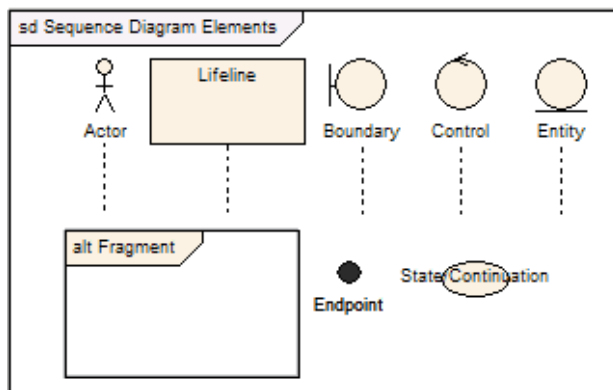
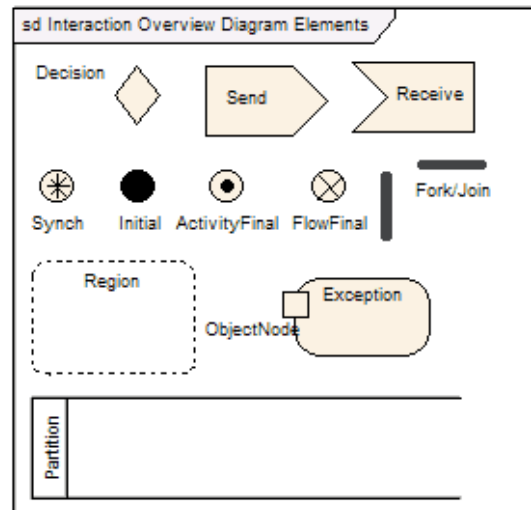
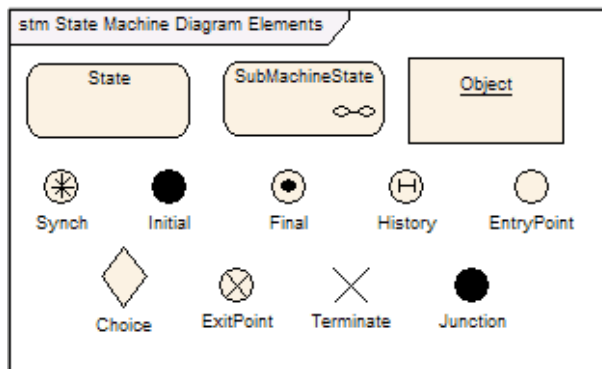
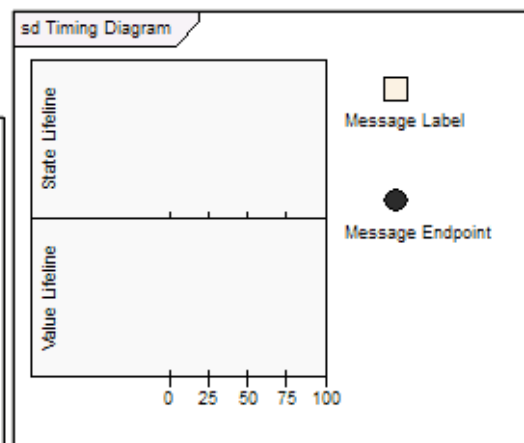
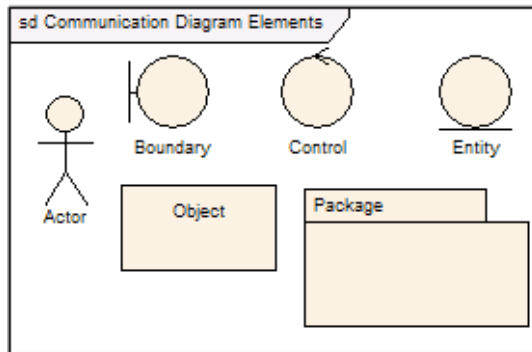
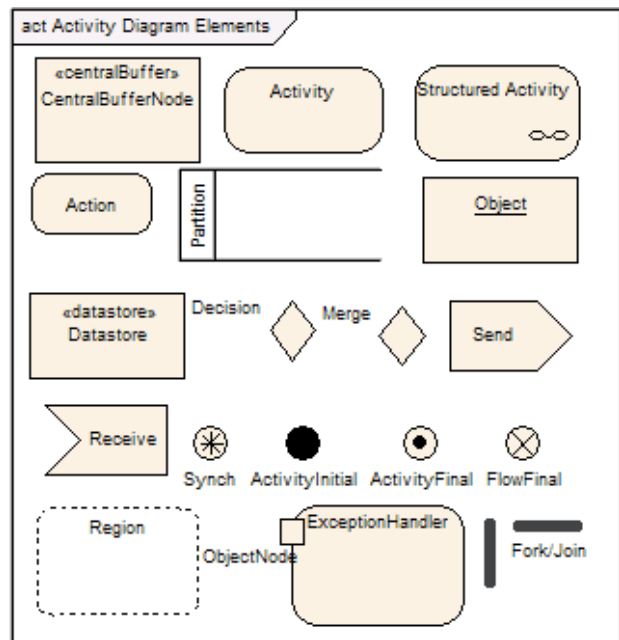
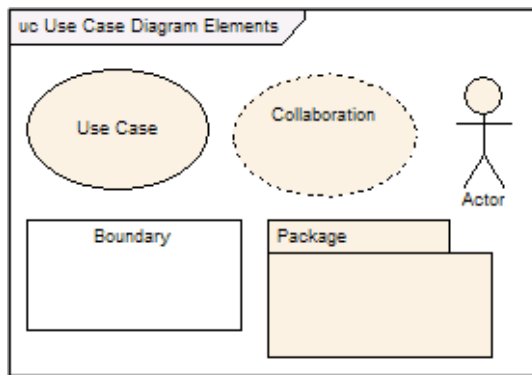
See Also

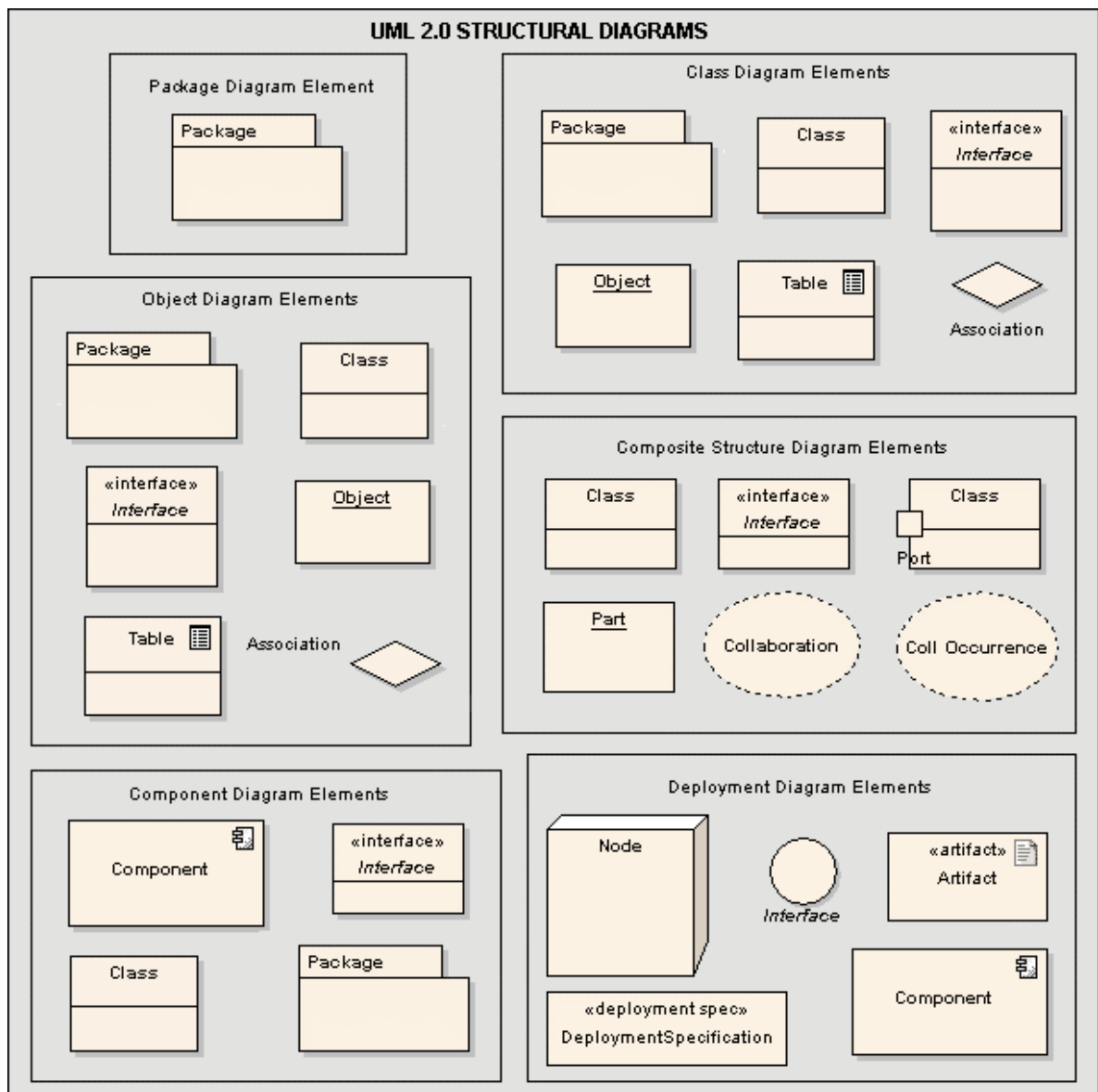
- [Diagram Properties](#)
- [RTF Dialog Options](#)

6.2 Working with UML Elements

UML Models are constructed from elements, each of which has its own meaning, rules and notation. Elements can be used at different stages of the design process for different purposes.

The basic elements for UML 2.0 are:





6.2.1 Element Context Menu

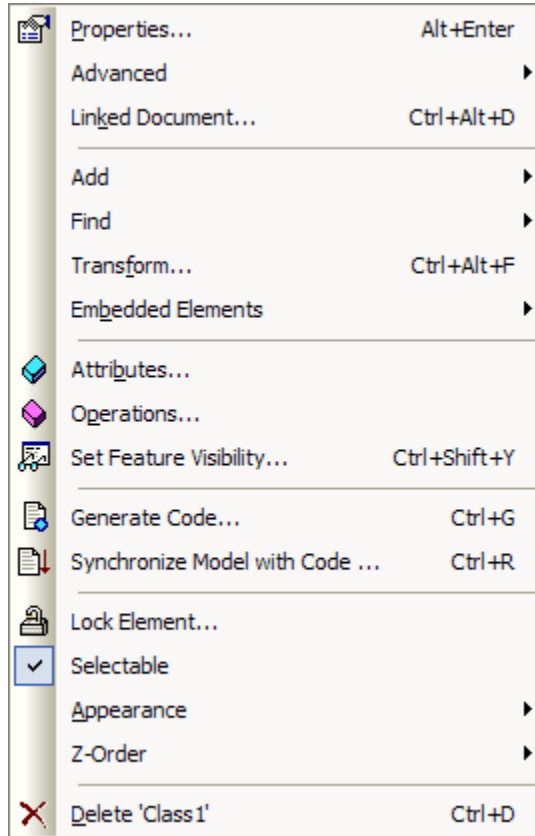
Right click on a single element in a diagram to open the element context menu. If two or more elements are selected, a different, [multiple selection context menu](#) will be displayed.

The element context menu is split into a number of distinct sections:

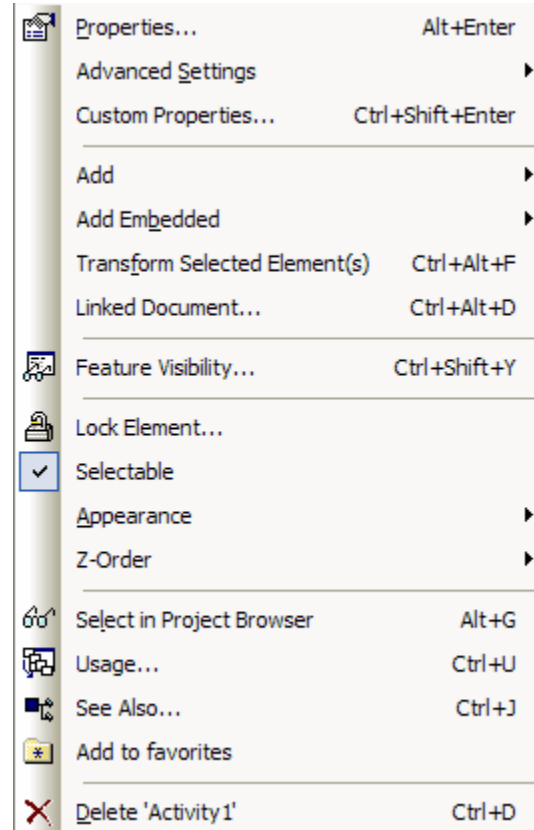
- [Properties](#)
- [Embedding](#)
- [Features](#)
- [Code Engineering](#)
- [Appearance](#)
- [Common Actions](#)
- Delete - you can delete the element from this menu option.

Note: Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

Example Context Menu for a Class:



Example Context menu for an Activity:



See Also:

- [Multiple Selection Context Menu](#)

6.2.1.1 Properties Menu Section

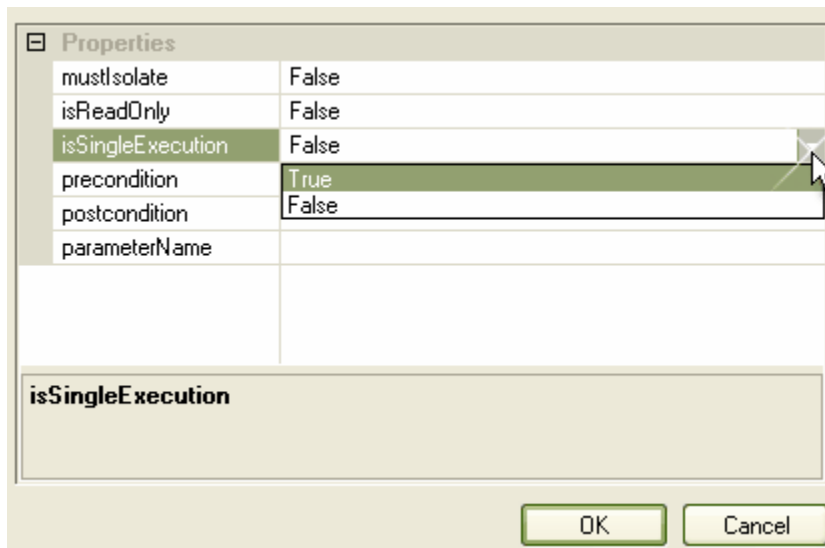
The Properties menu section on the element context menu may contain the following options:

| Menu Option | Description |
|-------------------|---|
| Properties | Opens the Properties dialog for the selected element. |
| Advanced Settings | Opens the Advanced Settings sub-menu. |
| Custom Properties | Opens the Custom Properties dialog. |

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.1.1 Custom Properties Dialog

Some element and connectors feature the *Custom Properties* option in their context menu. The example below is the Advanced Properties dialog belonging to an Activity element. Properties will vary depending on the kind of element or connector.



As shown above, properties can be altered either with a drop down list or by typing in the field to the right of the property.

6.2.1.1.2 Advanced Settings

The Advanced Settings menu section on the element context menu may contain the following options:

| Menu Option | Description |
|--------------------------------|---|
| Element Parent | Allows you to set the element parent . |
| Instance Classifier | Set the instance classifier for the element. |
| Composite Element | Set the element as a composite element . |
| Link Class to Association | Available if the element is a class. Allows you to link the class to a new Association . |
| Use Rectangle(Circle) Notation | Use rectangle notation for the element. |
| Partition Activity | Define an Activity Partition . |
| Set Run State | Add a new instance variable to the element using the Define Run State dialog. |
| Set Multiplicity | Define the multiplicity for the element. |
| Convert to Instance | Converts this classifier to an instance. |
| Attribute Initializers | Allows you to pre-define initial values for attributes that can be used to override existing defaults. |
| Make Sender/Receiver | Toggles the element to from a sender to a receiver and vice versa. |
| Accept Time Event | Allows you to change the notation for an Accept event action to a Accept time event action |
| Set Object State | Allows you to set the state of an instance classifier based on the child states for that object |
| Define Concurrent Substates | Allows you to define a set of sub states that can be held simultaneously within that composite state. |
| Use State Label Notation | Allows you to display State Table Notation for a state object |
| Deep History | Changes the type of Pseudo state to a Deep History. Applies only when right clicking on a history Pseudo State. |

Note: Not all menu options shown here will be present on all element context menus. Context menus vary

slightly between element types. The Partition Activity section will only appear for an Activity element, for example.

6.2.1.2 Embedding Menu Section

The Embedding menu section on the element context menu may contain the following options:

| Menu Option | Description |
|-----------------------------|--|
| Add | Opens the Add Sub-menu. |
| Add Embedded | Opens the Add Embedded Sub-menu. |
| Transform Selected Elements | Converts selected model fragments from one domain to another, see MDA Transforms |
| Linked Document | Creates a Document and links it to this element. |
| Delete Linked Document | Delete the Document linked to this element. |

Add Sub-Menu

| Menu Option | Description |
|-------------------------|---|
| Add Tagged Value | Allows you to add a tagged value . |
| Attach Note | Creates and Attaches a blank Note to the element. |
| Attach Constraint | Creates and Attaches a blank Constraint to the element. |
| Activity Diagram | Creates an Activity diagram that is owned by the element. |
| Sequence Diagram | Creates a Sequence diagram that is owned by the element. |
| Communication Diagram | Creates a Communication diagram that is owned by the element. |
| Statechart | Creates a Statechart that is owned by the element. |
| Insert Related Elements | Opens the Insert Related Elements dialog. |

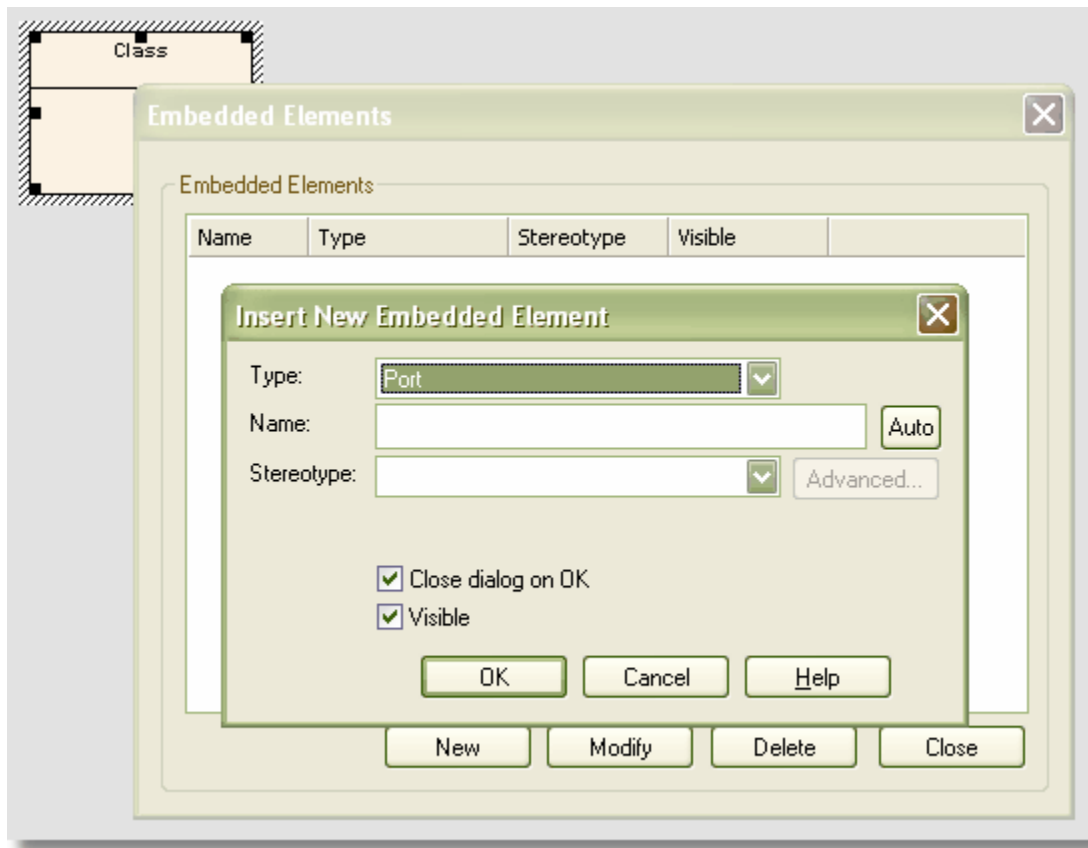
Add Embedded Sub-Menu

| Menu Option | Description |
|--|--|
| Embedded Elements | Opens the Embedded Elements window. |
| Show Realized Interfaces | Displays each interface directly realized by a class. |
| Show Dependent Interfaces | Displays each dependency relationship for that model element as a lollipop style node attached to its left-hand side |
| Port | Adds an embedded Port to the element |
| Required Interface | Adds an embedded Required Interface to the element |
| Provided Interface | Adds an embedded Provided Interface to the element |
| Action Pin | Adds an embedded Action Pin to the element |
| Expansion Node | Adds an embedded Expansion Node to the element |
| Object Node | Adds an embedded Object Node to the element |
| Activity Parameter | Adds an embedded Activity Parameter to the element |
| Entry Point | Adds an embedded Entry Point to the element |
| Exit Point | Adds an embedded Exit Point to the element |

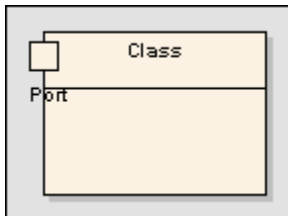
Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.2.1 Embedded Elements

The *Embedded Elements* dialog allows you to embed particular elements into other elements. For example, a Port can be embedded into a Class. The *Embedded Elements* option is available on the context menu of some elements.



In the *Embedded Elements* dialog, press *New* to create a new embedded element. Enter details such as type, name and stereotype, and press *OK*. The embedded element will now show on the primary element as shown below.



You can add as many embedded elements as necessary. Modify or delete embedded elements using the *Embedded Elements* dialog.

6.2.1.2.2 Insert Related Elements

The *Insert Related Elements* dialog can be accessed from most element context menus. This dialog allows you to insert linked elements from other diagrams into the current diagram.

You can specify the following details:

| Element | Description |
|---------------------------------------|---|
| Insert linked classes to <<x>> levels | Select the level you want to insert linked elements - levels 1-5 are available. |
| Link Type | Select the type of link you want the inserted elements to be connected by. |
| Link Direction | Select whether you want the links to be a single direction or bi directional. |
| Element Type | Select the element type you want to insert. |
| Limit to this namespace | Limit the namespace you want the inserted elements to come from. |
| Layout Diagram When Complete | Select whether you want EA to layout the diagram after the elements have been inserted. |

6.2.1.3 Features Menu Section

The Features menu section on the element context menu may contain the following options:

| Menu Option | Description |
|--------------------|--|
| Attributes | Opens the Attributes dialog. |
| Operations | Opens the Operations dialog. |
| Feature Visibility | Opens the Set Feature Visibility dialog. |

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.4 Code Engineering Menu Section

The Code Engineering menu section on the element context menu may contain the following options:

| Menu Option | Description |
|--|--|
| Generate Code (forward engineer) | Generate source code for the selected element. |
| Synchronize with Code (reverse engineer) | Reverse engineer source code for the selected element. |
| View/Edit Source Code | Opens the source editor if a file exists for that selected element. |

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.5 Appearance Menu Section

The Appearance menu section on the element context menu may contain the following options:

| Menu Option | Description |
|--------------|---|
| Selectable | Toggle whether the element is selectable. Selectable means shows the context menu - if it is unselectable, only the <i>Selectable</i> menu item shows on right click. |
| Appearance | See table below for sub-menu. |
| Z-Order | Set the Z-Order of the element. |
| Lock Element | Locks the element so it can't be edited. It can be unlocked by selecting Lock Element again. |

Appearance Sub-Menu

| Menu Option | Description |
|---|--|
| Adjust Appearance | Change the element's default appearance . |
| Select Alternate Image | Select an alternate image using the image manager . |
| Set Font | Change the font used for an element, to select the font for multiple elements within a diagram select the elements by holding down the <i>CTRL</i> key and then use the set font command . |
| Copy Appearance to Painter | Copy the element appearance to the painter |
| Copy Image of Selected Object(s) to Clipboard | Copy the element to the clipboard, so you can paste it into an external application. |

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.6 Common Actions Menu Section

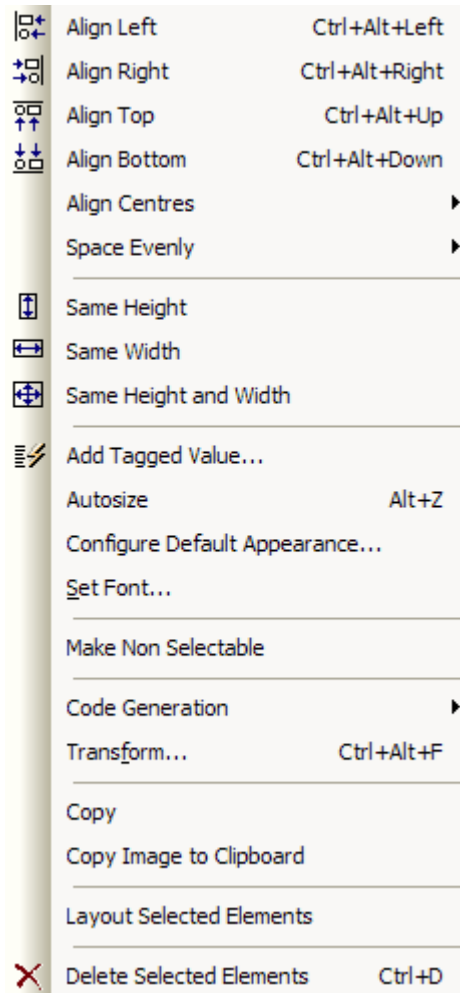
The Common Actions menu section on the element context menu may contain the following options:

| Menu Option | Description |
|---------------------------|--|
| Select in Project Browser | Selects the currently selected element in the Project Browser. |
| Usage | Opens the Element Usage dialog |
| See Also | Allows you to set up Cross References . |
| Add to Favorites | Add the element to the Favorites folder . |

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.7 Element Context Menu - Multiple Selection

Right clicking on a selection of two or more elements in a diagram will cause the following context menu to be displayed:



This allows you to:

- Align elements (left, right, top, bottom or center)
- Space elements evenly (across or down)
- Set the appearance for multiple elements at once
- Make the selected elements on the diagram non selectable. To make them selectable again, right click on the diagram and select, **Make All Elements Selectable**.
- Specify the visibility of features for all selected elements
- Generate code for all selected elements at once
- Copy all selected elements to the clipboard
- Delete all selected elements

Tip: It is much faster to assign an appearance or characteristic to a group of elements than doing it one at a time.

6.2.2 Common Element Tasks

This topic covers various common UML tasks you can perform in Enterprise Architect.

6.2.2.1 Creating Elements

Elements within a model are typically arranged on diagrams to visually communicate the relationships between a given set of elements. Enterprise Architect provides simple mechanisms for creating elements in the model, using diagrams or the *Project Browser*.

Creating Elements on a diagram

The fastest and simplest ways to create elements directly on a diagram is via the *Quick Linker* and the *UML toolbox*. The following sections describe these and other approaches for creating elements on a diagram:

- [Creating Elements in place using the Quick Linker](#)
- [Creating Elements using the UML Toolbox](#)
- [Creating Elements using the Diagram Context Menu](#)
- [Creating a group of Elements using UML Patterns](#)
- [Creating Domain Specific Elements from UML Profiles](#)

Adding Elements directly to a package

Sometimes it is useful to add elements to a package, without a diagrammatic representation. This can be accomplished via the *Project Browser* and is explained in the following section:

- [Adding Elements directly to a package](#)

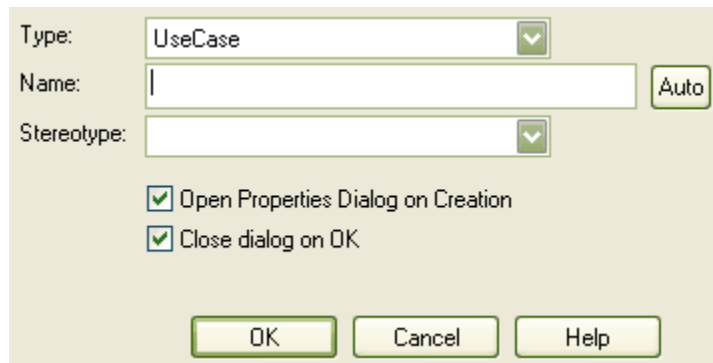
6.2.2.2 New Elements

New Elements may be quickly added to a package without the necessity of adding a diagram element at the same time. This is particularly useful if you wish to define a group of requirements, changes, issues, base classes or other element type that may not require diagrammatic representation in the model.

Add a New Element to a Package

To add a new element to a package, follow the steps below:

1. In the Project Browser, right click an appropriate package to open the context menu.
2. Select *New Element*.
3. Select the element *Type* from the drop down list.
4. Enter a *Name* for the new element and an optional *Stereotype*.
5. Check the *Open Property Dialog* option if you want the property dialog to open immediately after the element is created.
6. Clear the *Close Dialog on OK* option if you want to add multiple elements in one session.



Type: UseCase
Name: Auto
Stereotype:
 Open Properties Dialog on Creation
 Close dialog on OK
OK Cancel Help

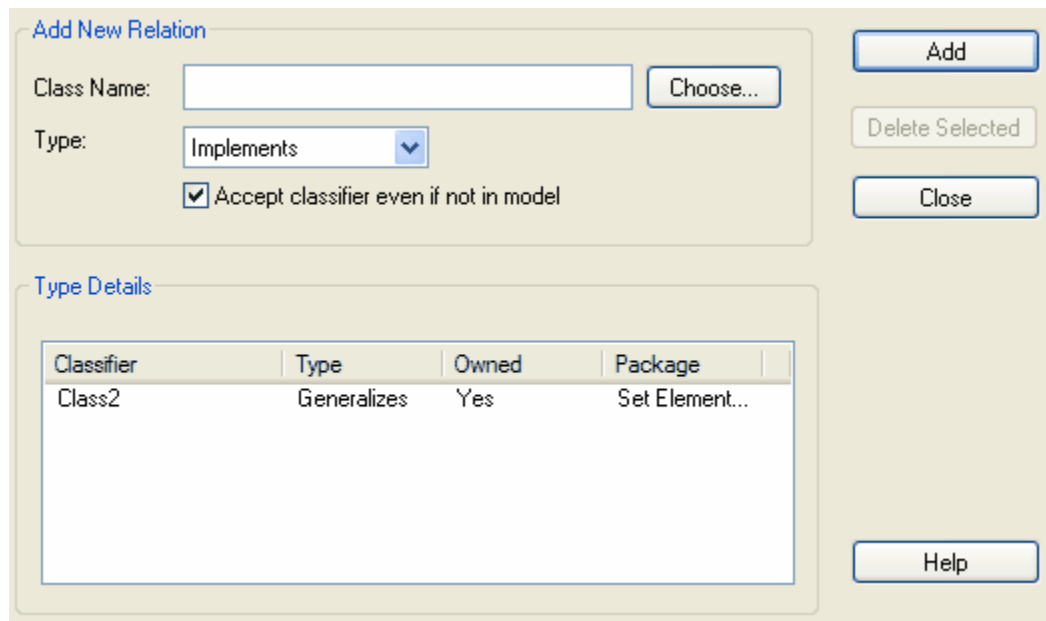
6.2.2.3 Set Element Parent

You can manually set an element's parent or an interface it realizes, using the *Type Hierarchy* dialog.

Set the Element Parent

To set the element parent, follow the steps below:

1. Select a generalizable element in a diagram.
2. From the *Element* menu, select *Advanced | Set Parents and Interfaces*. Alternatively, press *Ctrl+I*.
3. The *Type Hierarchy* dialog will open.



Add New Relation

Class Name: Choose...
Type: Implements
 Accept classifier even if not in model

Type Details

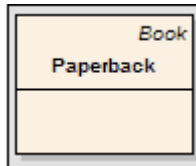
| Classifier | Type | Owned | Package |
|------------|-------------|-------|----------------|
| Class2 | Generalizes | Yes | Set Element... |

Add
Delete Selected
Close
Help

4. You can elect to enter a parent or interface name by either manually by typing it in, or by using the *Choose...* function to locate an element within the current model.
5. Set the *Type* of relationship (implements or inherits) from the drop down list.
6. Press *Add* to add the relationship.

7. Press *Delete Selected* to remove the current selected relationship.

Note: Parents that do not have their corresponding related element in the same diagram will display their parentage in the top right corner of the child element, as shown below:



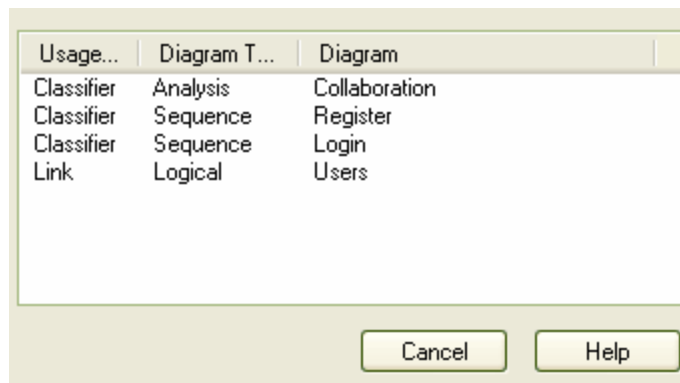
6.2.2.4 Show Element Usage

You may display the usage of an element using the *Element Usage* dialog. This lists all occurrences of the element throughout the model, and allows you to easily navigate to any occurrence.

Show Element Usage

To show element usage, follow the steps below:

1. Select an element in a diagram.
2. From the *Element* menu, select *Find in Diagrams*. Alternatively, press *Ctrl+U*.
3. The *Element Usage* dialog will open, displaying all occurrences of the current element in the model.



4. Double click a line item to open the relevant diagram and display the selected element.

Note: You may also access this feature from the *Project Browser* - select an element in the tree and select *Show Usage* from the *Element* menu.

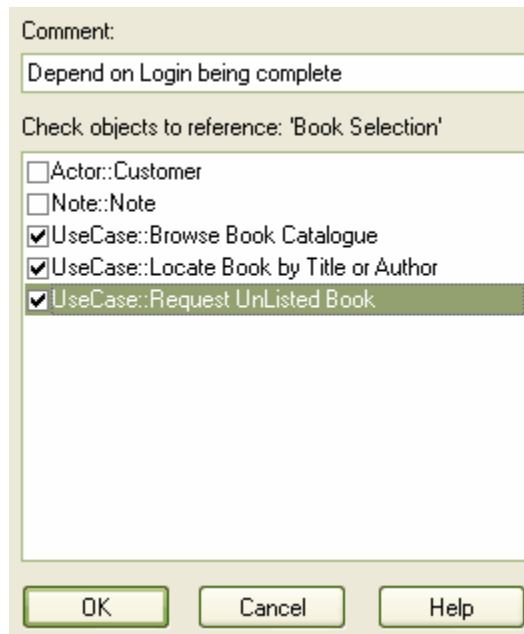
6.2.2.5 Set Up References (Cross References)

It is possible to set up a cross reference from one element in EA to another.

Set Up a Cross Reference

To set up a cross reference, follow the steps below:

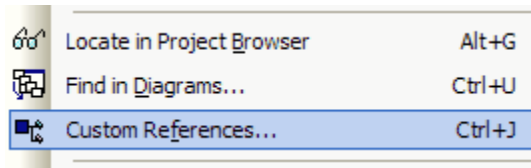
1. In the Project Browser, locate the target element or diagram (this will be the subject of the cross reference).
2. Open a diagram that contains the element(s) that will have the currently selected element as a reference.
3. Right click on the element to open the element context menu.
4. Select *Add element as elements(s) reference....* (In the case of a diagram select *Add diagram as element(s) reference....*)
5. In the *Set Reference* dialog, elements you wish to have include the currently selected item in the explorer as a reference.
6. Enter an optional *Comment* to describe the purpose of the reference.



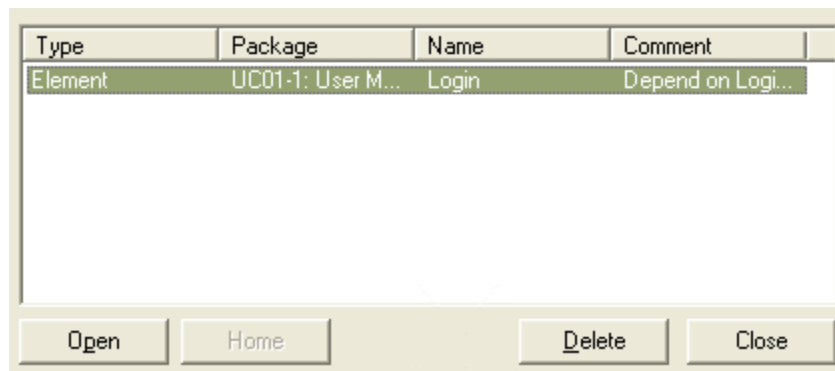
Use the Cross Reference

To use the cross reference, follow the steps below:

1. Select an element in a diagram.
2. From the *Element* menu, select *Custom References*. Alternatively, press *Ctrl+J*.



3. The *Cross Reference* dialog will appear and display a list of elements that have been set as cross references.
4. You can open a selected element by highlighting it and pressing *Open*.
5. If you have a diagram cross reference, you can *Open* that diagram.
6. If you have a string of diagram links, press *Home* to return to the original diagram.



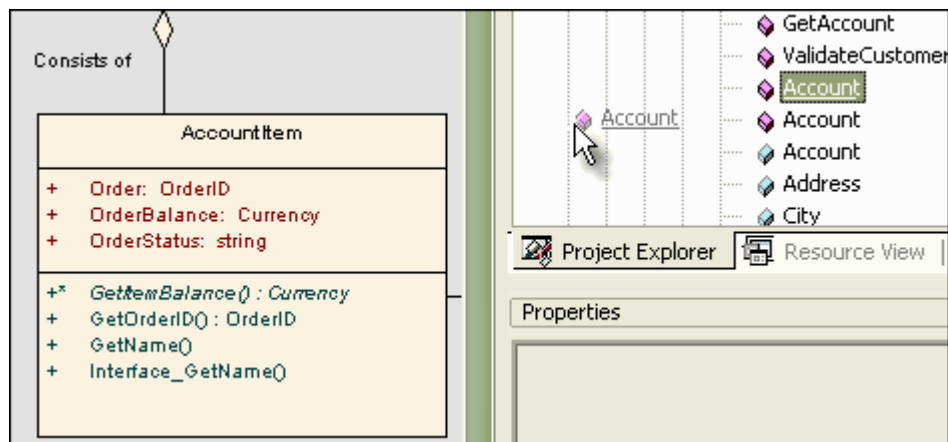
6.2.2.6 Copying Attributes and Operations Between Elements

Using drag and drop, you can copy an attribute and/or an operation from an element in the Project Browser on to another element in a diagram. or to another element within the Project Browser.

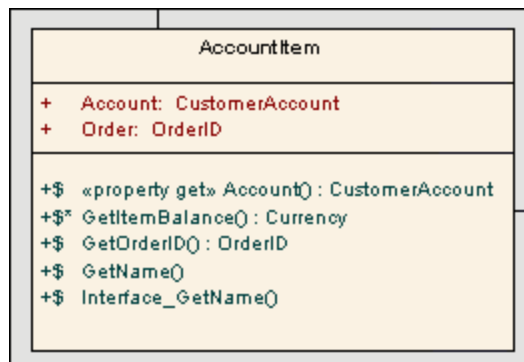
Copy an Element Feature

To copy an element feature, follow the steps below:

1. Open a diagram which contains the target element (in the example below, the AccountItem class is the target and Customer element is the donor).
2. Left click on the Attribute or Operation and drag to the target element.
3. Release the mouse button.



The image below shows AccountItem after the attribute 'Account' has been dropped from the browser on to it.



Move Multiple Element Features in the Project

To copy an element feature, follow the steps below:

1. Open a diagram which contains the target element (in the example below, the AccountItem class is the target and Customer element is the donor).
2. Left click on the Attribute or Operation and drag to the target element.
3. Release the mouse button.

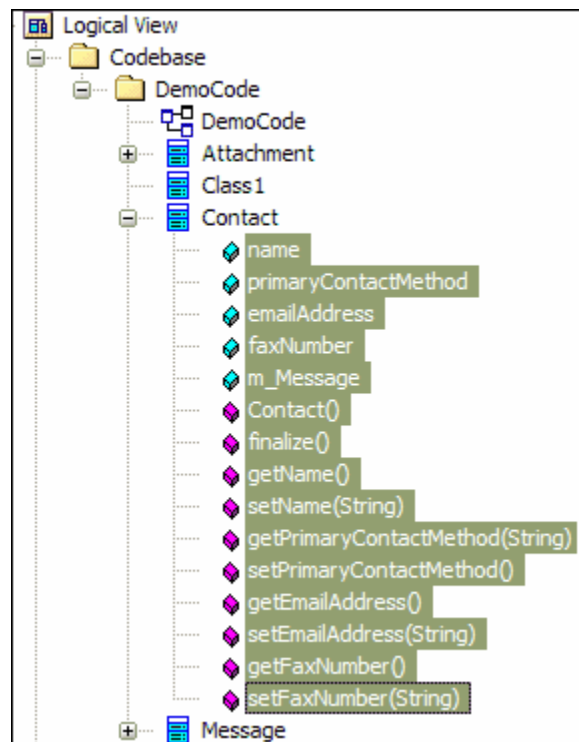
6.2.2.7 Moving Attributes and Operations between Elements

By using drag and drop, you can move attributes and/or operations from an element in the Project Browser on to another element within the Project Browser.

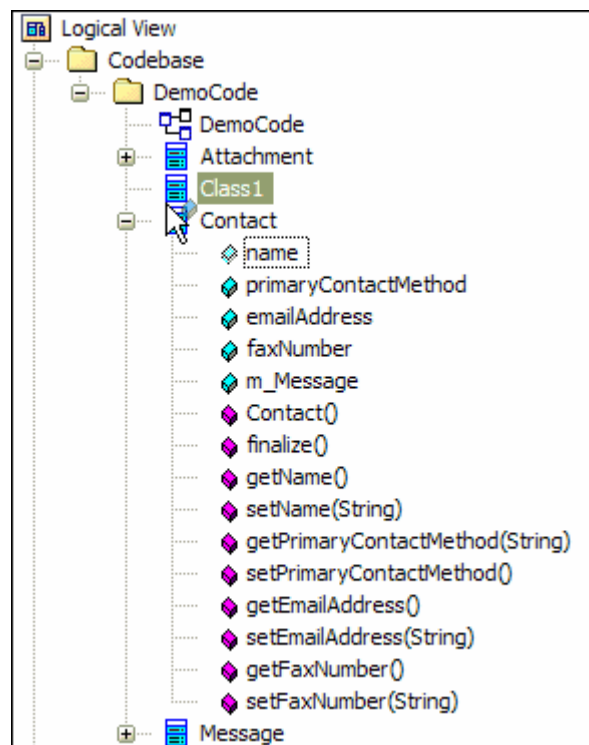
Move Multiple Element Features in the Project Browser

To move element features, follow the steps below:

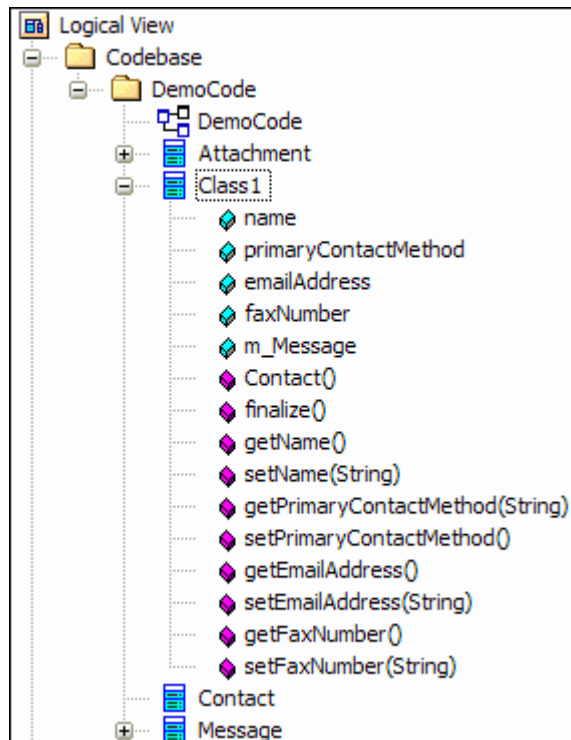
1. Locate the attributes and/or operation in the Project Browser that you wish to move from the target element and select them by holding down the **Ctrl** (single item select) or the **Shift** key (multiple item select) .



2. Holding down the *right mouse button* and the *Shift* or *Ctrl* key move the attributes and/or operations to the target element. This move will display only the last selected element feature during the move, however all of the selected features will be moved.



3. Release the mouse button. The image below shows the final stage of the attribute and operations move between the Contact class and Class1.



6.2.2.8 Moving Elements between Packages

To move an element between packages, follow the steps below:

1. In the *Project Browser*, locate the element you wish to move.
2. Left click on the element and drag the mouse cursor to the package you wish to drop the element into.
3. Release the mouse button.

EA will move the element into the new package.

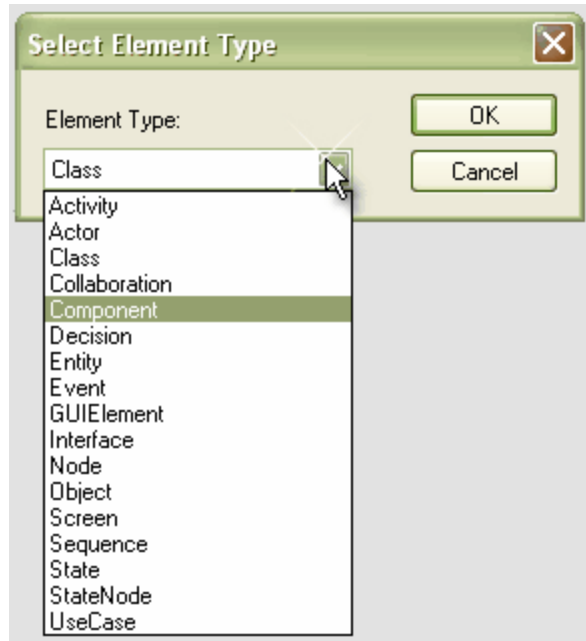
Tip: You can also drag the element under a host element in the new package - for example drag an element under a class

6.2.2.9 Changing Element Type

To change an element type, follow the steps below:

1. In the diagram view, element you wish to change.
2. From the *Element* menu, select *Change Type*.

3. Select the required *Element Type* from the drop down list.
4. Press *OK*.

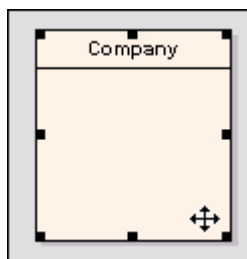


The target will be transformed into the required type.

6.2.2.10 Moving Elements

Any one of the following options will allow you to move an element within a diagram. Select an element or group of elements in the diagram view, then:

- Use the mouse to drag to the desired position (mouse cursor switches to the four-arrow icon as shown below), or



- Hold down the *Shift* key, and use the *Arrow* keys to move to the desired position, or
- Use the *Left*, *Right*, *Up* and *Down* options in the *Element | Move* submenu, or
- Multiple elements may be aligned to one another using the *Element | Alignment* submenu, the Alignment options in the right-click context menu, or the Alignment buttons on the *Diagram* toolbar



6.2.2.11 Aligning Elements

To align multiple elements, follow the steps below:

1. Select a group of elements by drawing a selection box around them all. (Or select one by one by holding down the **Ctrl** button and left clicking each element)
2. To open the context menu, **right click** on the element(s) in the group you wish to align others to.
3. Select the alignment function you require.

All selected elements will be aligned to the one beneath the cursor.

Tip: You can also use the diagram toolbar. The first four buttons are used to align elements, and are made available when more than one element is selected in a diagram.

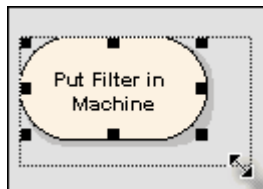


Tip: You can also select **Alignment** from the **Element** menu.

6.2.2.12 Resizing Elements

Any one of the following options will allow you to resize an element. Select an element or group of elements in the diagram view, then:

- Use the resize handles which appear at each corner and side to resize the element(s) by dragging with the mouse (mouse cursor switches to the double-ended arrow as shown below), or



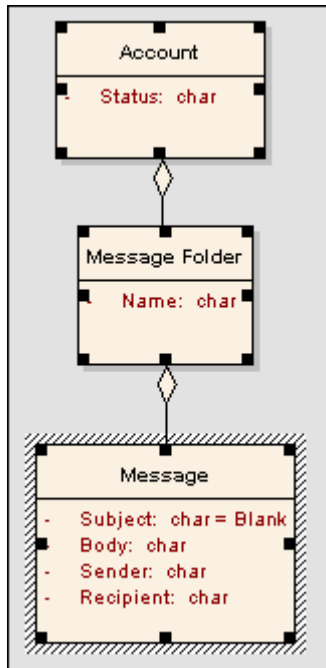
- Hold down the **Ctrl** key, and use the **Arrow** keys to resize as desired, or
- Use the **Wider**, **Narrower**, **Taller** and **Shorter** options in the **Element** | **Move** submenu, or
- Autosize selected element(s) using the option in the **Element** | **Appearance** submenu, or by pressing **Alt +Z**. (With multiple elements selected, **Autosize all selected** appears in the right-click context menu), or
- Set multiple elements to the same height, width or both, using these options in the **Element** | **Make Same** submenu, or the options in the right-click context menu

Resizing a Set of Objects to a Specific Size

Right clicking a selected set of object allows you to resize them to the same height, width or both. When you select multiple elements using **Control+Click**, then resize the height and/or width, the height and/or width of the selected (hatched) object on which the right-click is performed will be used to set the height and/or width of the other selected objects.

For example, in the diagram below, the Message class's height and width will be used to set the height and width of the Account and Message Folder classes. The aim is to make the Account and Message Folder

elements the same height and width as the Message element.



To do this follow the steps below:

1. Set one element to the size you want (eg. Message as above).
2. Select all other elements (eg. Account and Message Folder as above).
3. Right-click the pre-sized element (eg. Message).
4. Select your resizing option (same height, width etc.).

See Also

- [Highlight Context Element](#)

6.2.2.13 Deleting Elements

To delete an element from a diagram, follow the steps below:

1. In the active diagram, select the element you wish to delete.
2. Press the *Delete* key, or right click to open the context menu and select *Delete <element name>*.

Note: This does not delete the element from the model, only from the current diagram.

To delete an element from the model, follow the steps below:

1. In the Project Browser, right click on the element you wish to delete.
2. From the context menu, select *Delete <element name>*.

3. Confirm *Yes* when prompted.

-OR-

4. Press the *Ctrl + Delete* key combination on a selected element in a diagram to completely remove an element from the model.

To delete an multiple elements from a diagram and model, follow the steps below:

1. Open the diagram with the elements that you wish to remove from the model.
2. Select all of the elements in a diagram by pressing the *CTRL + A* hotkey combination to target specific elements use the *CTRL + left mouse* click combination.
3. Press the *CTRL + Delete* to completely remove the elements from the model .

To delete an multiple elements from the Project Browser and model, follow the steps below:

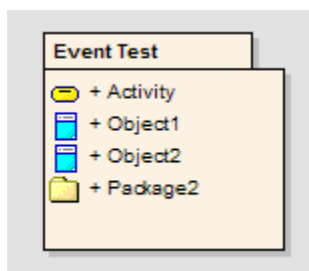
1. Select the items in the Project browser by holding down the *CTRL* or *SHIFT* key and holding down the right mouse button. The *CTRL* key will enable the user to select multiple items from the Project Browser one at a time whereas the *SHIFT* key select all of the items in the Project Browser tree between the first clicked item and the last clicked item.
2. Press the *CTRL + Delete* to completely remove the elements from the model .

Note: If you delete an element here by using the *CTRL + Delete* hotkey combination, all its properties and connectors are deleted as well.

6.2.2.14 Customize Visible Elements

Some elements are hidden from view in packages and in RTF documents by default - this includes Events, Decisions, Sequence elements and Associations. You have the option of turning these elements back on.

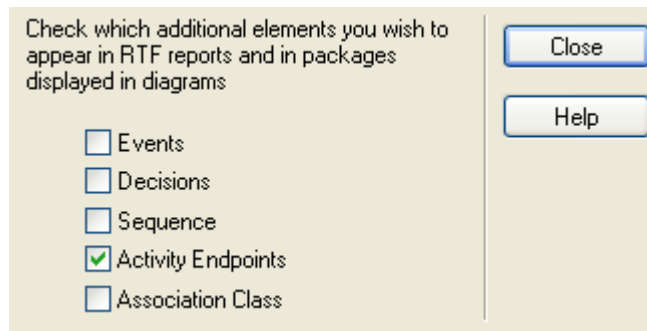
For example, some events and decisions contained in a package do not appear in the package view - as in the example below.



Customize Visible Elements

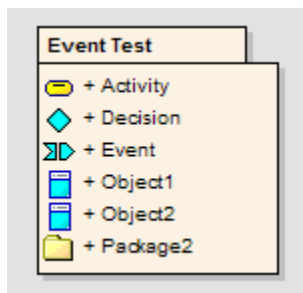
To show additional elements, follow these steps:

1. From the *Tools* menu, select *Options* to open the *Local Options* dialog.
2. Select the *Objects* tab.
3. Press *Advanced* to open the *Advanced Settings* dialog:



4. Check the elements you wish to show in packages and in RTF documents.
5. Press *Close*. Close the *Local Options* dialog.
6. Reload the current diagram if required.

The package from the example above will now show the Event and Decision elements it contains:



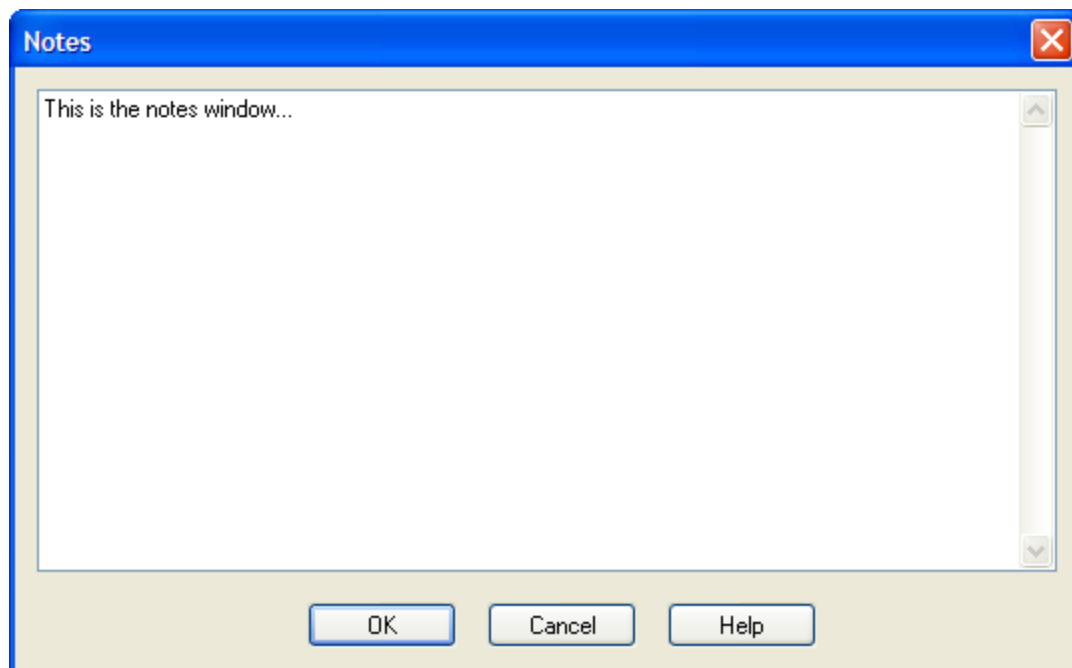
6.2.2.15 Creating Notes and Text

You can create notes and text in EA - the two are slightly different.

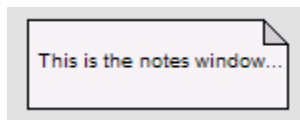
Creating a Note

One way to create a note is:

1. Right click on the background of a diagram to open the context menu.
2. From the *Insert Element at Cursor* submenu, select *Note*. The *Notes* window appears.



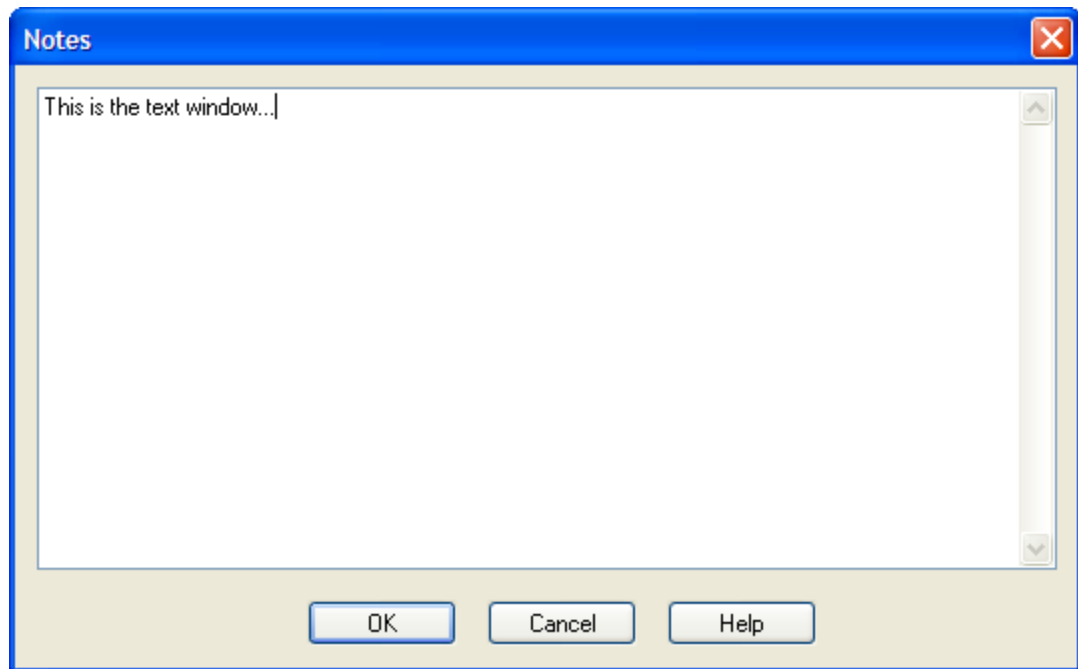
3. Type your note then press *OK* to save.
4. Your note will appear on the diagram in the following format:



Creating Text

To create text, follow the steps below:

1. Right click on the background of a diagram to open the context menu.
2. From the *Insert Element at Cursor* submenu, select *Text*. The *Notes* window appears.



3. Type your note then press **OK** to save.
4. Your text will appear on the diagram in the following format:

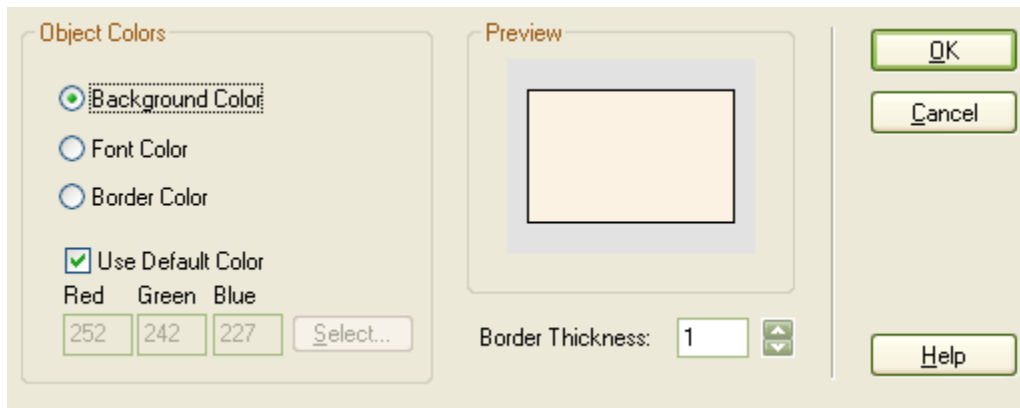
This is the text
window...

6.2.2.16 Configure an Element's Default Appearance

The global appearance of all elements throughout a model can be changed via the *Local Options* dialog, select **Tools | Options** from the main menu, then select *Standard Colors* from the options tree.

In order to over ride the default appearance of an element, right click on the element and select *Appearance | Adjust Appearance* from the submenu. Alternatively you can use the [Format toolbar](#). To access the Format Toolbar, select **View | Toolbars | Format Tool** from the main menu.

Note: Multiple elements can be adjusted at the same time. Select multiple elements and right click on one of them, select *Set Appearance* from the context menu.



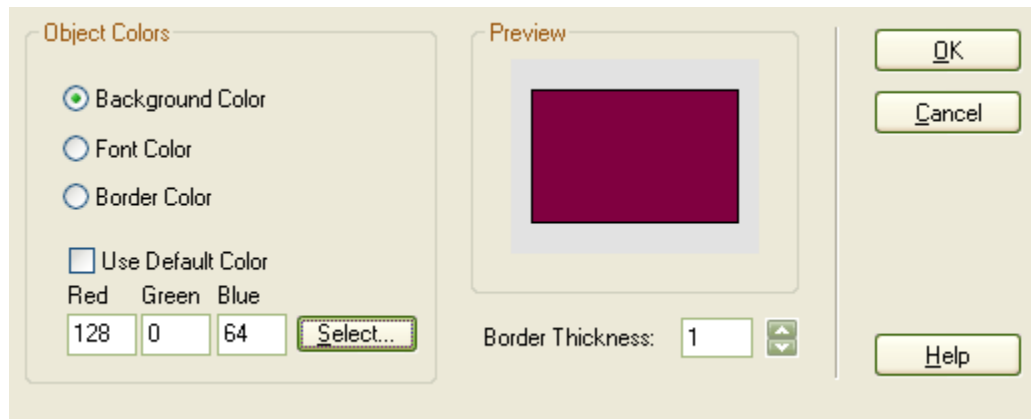
Changing a the Background, Font or Border Color

You can set the background color, the font color or the border color by following the steps below:

1. Select the *Background Color*, *Font Color* or *Border Color* radio button as appropriate.
2. Uncheck the *Use Default Color* option to enable the *Select* button.
3. Press *Select* to open the *Color* dialog.



4. Select the color you want (use *Define Custom Colors* for a more exact color) and press *OK*.
5. You will see in the *Preview* that the element's color has changed to the selected color. This won't be applied to the actual element until you press *OK*.

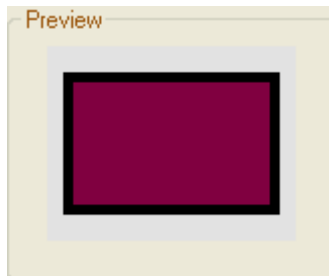


Note: The color can be easily be returned to the default color by checking the *Use Default Color* checkbox.

Note: You can change it to a different color by pressing *Select* again.

Changing the Border Thickness

To change the border thickness, change the number in the *Border Thickness* field on the right. Use the scroll arrows to increase or decrease the number, or type a new number directly into the text box.



You can see the effect in the preview pane. This won't be applied to the actual element until you press *OK*.

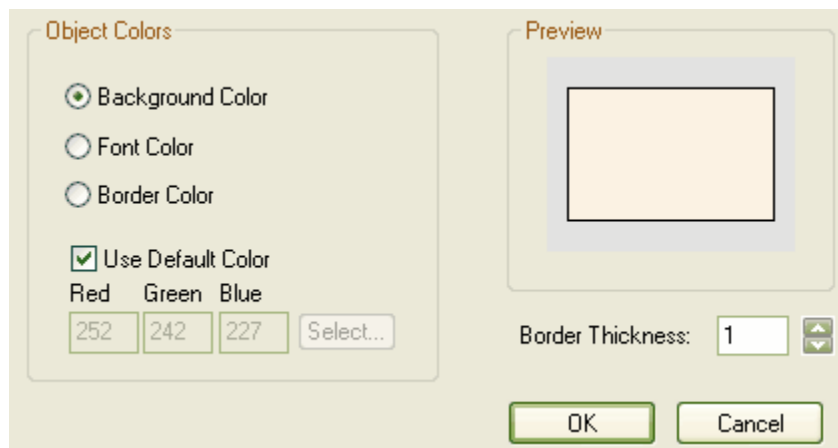
6.2.2.17 Get/Set Project Custom Colors

There may be times when more than one person is working on a project and they may wish to use specific colors for elements within the project. The *Get* and *Set Project Custom Colors* items on the *Project* menu allow you to *set* specific colors then *get* the colors in a different session, without having to remember RGB values.

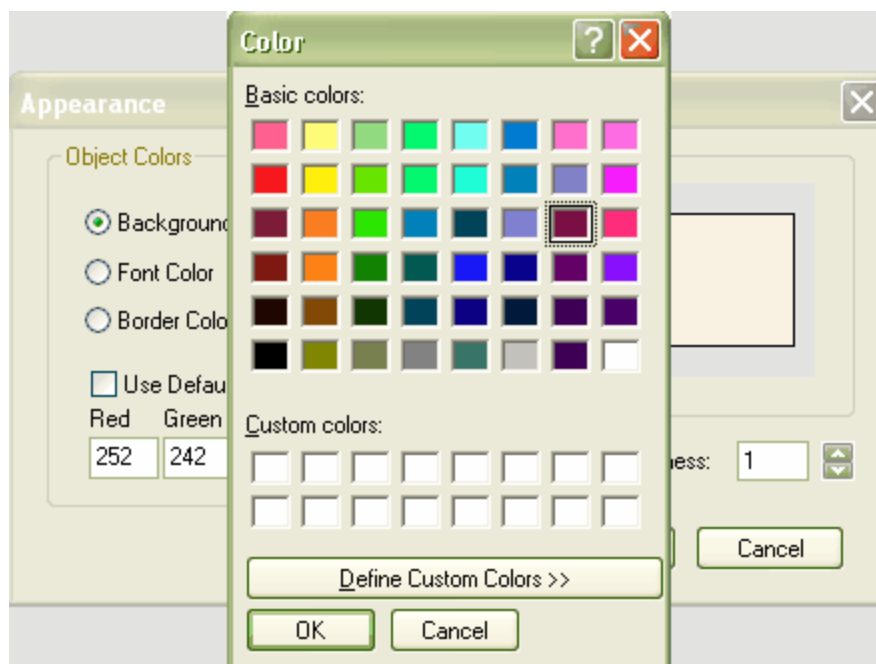
Set a Project Custom Color

Follow the steps below to set your project's custom colors:

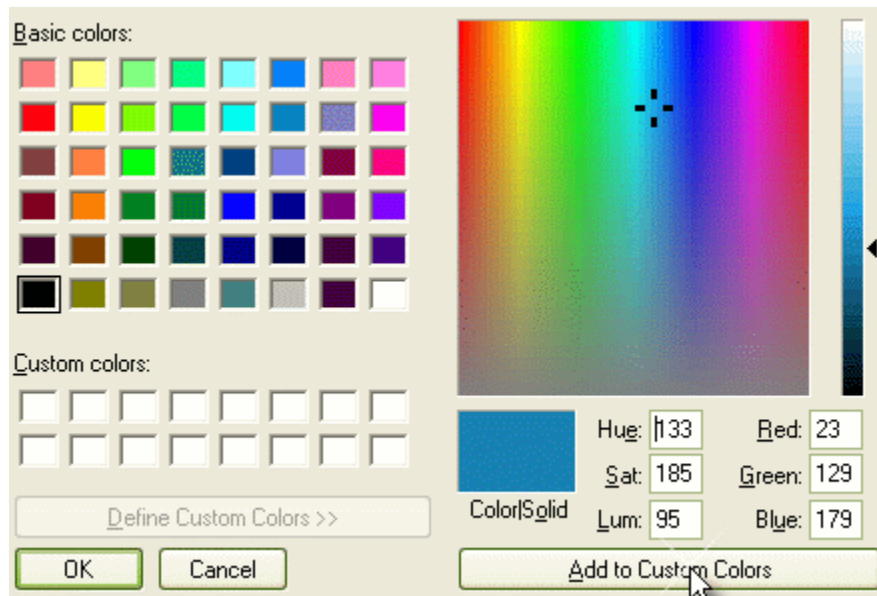
1. Select an element you would like to color.
2. From the *Element* | *Appearance* submenu, select *Configure Default Appearance...* to open the *Appearance* dialog.



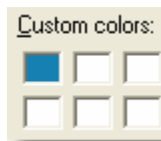
3. Uncheck the *Use Default Color* option to enable the *Select* button.
4. Press *Select* to open the *Color* dialog.



5. Press *Define Custom Colors >>*.
6. Create the color in the color mixer on the right - see below.



7. Press **Add to Custom Colors**. This will add the color to the **Custom colors** listing on the left hand side of the dialog - see below:



8. Press **OK** to close the **Color** dialog, then **OK** to close the **Appearance** dialog.
9. From the **Project** menu, select **Set Project Custom Colors** to save the custom color you have created.

Get a Project Custom Color

Follow the steps below to get your project's custom colors:

1. From the **Project** menu, select **Get Project Custom Colors**. This will apply any saved custom colors to this project.
2. From the **Element | Appearance** submenu, select **Configure Default Appearance...** to open the **Appearance** dialog.
3. Uncheck the **Use Default Color** option to enable the **Select** button.
4. Press **Select** to view the applied custom color(s) - they will appear as circled below:



6.2.3 Attributes and Operations

Attributes are features of classes and some other elements that correspond to the data and state an element maintains. Operations are features that indicate the behavior and functionality an element has.

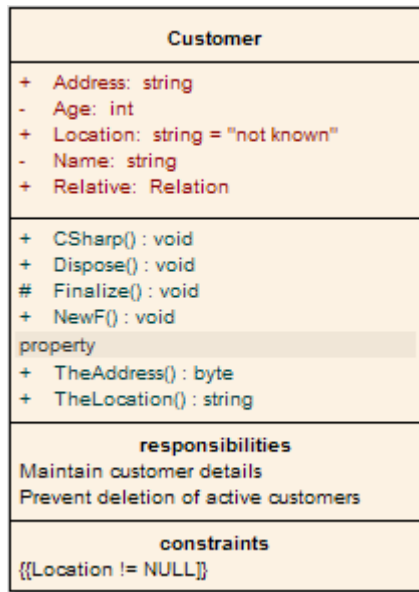
6.2.3.1 Attributes

Attributes are features of a class or other element that represent the properties or internal data elements of that element. For a Customer class, CustomerName and CustomerAddress may be attributes. Attributes have several important characteristics, such as type, scope (visibility), static, derived and notes.

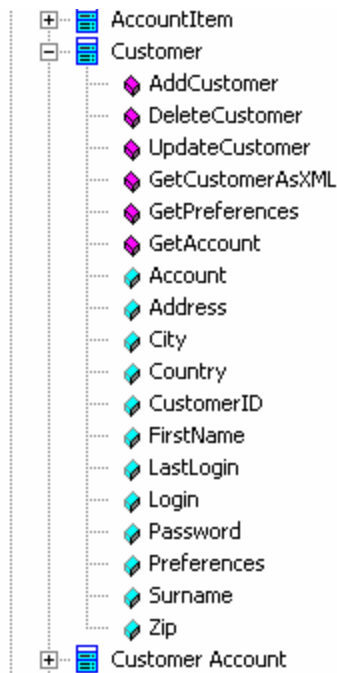
Creating and Modifying Element Attributes

1. In the diagram view, right click on the element to be edited.
2. From the context menu *Element Features* submenu, select *Attributes* to open the *Attributes* dialog.
3. Alternatively, press *F9*.
4. Alternatively, drag the attribute from the Project Browser onto the Element.

Note: This option will only be available on the context menu if the element supports Attributes



Note: Attributes are displayed in the Project Browser beneath the element:



6.2.3.1.1 Attributes Main Page

The *General* tab of the *Attributes* dialog is shown below:

The screenshot shows the 'Attributes' dialog box with the 'General' tab selected. The 'Name' field is 'providerName', 'Type' is 'String', 'Scope' is 'Public', and 'Initial' is '\"RetirementSpecialists\"'. Below the fields are 'New', 'Save', and 'Delete' buttons. At the bottom are 'Close' and 'Help' buttons. An 'Attributes' section contains a table with columns 'Name', 'Type', and 'Initial Value'. The table lists 'providerName' (String, \"RetirementSpecial...\") and 'availableFunds' (Vector, null).

| Name | Type | Initial Value |
|----------------|--------|------------------------|
| providerName | String | \"RetirementSpecial... |
| availableFunds | Vector | null |

| Control | Description |
|-----------------|---|
| Name | Attribute name |
| Type | Data type of attribute - select from the drop down list |
| Build button | Opens the <i>Select Attribute Type</i> dialog |
| Scope | Public/Protected/Private/Package |
| Stereotype | Optional Stereotype of the attribute |
| Containment | Containment type(by reference/value) |
| Derived | Indicates attribute is a calculated value |
| Static | Attribute is a static member |
| Property | Select automatic property creation |
| Const | Attribute is a constant |
| Alias | An optional alias for the attribute |
| Initial | An optional initial value |
| Notes | Free text notes |
| Attribute List | List of defined attributes. Select an attribute to make it current |
| Up/Down buttons | Use to change the order of attributes in the list |
| New | Create new attribute |
| Save | Save new attribute, or save modified details for existing attribute |
| Delete | Delete currently selected attribute |

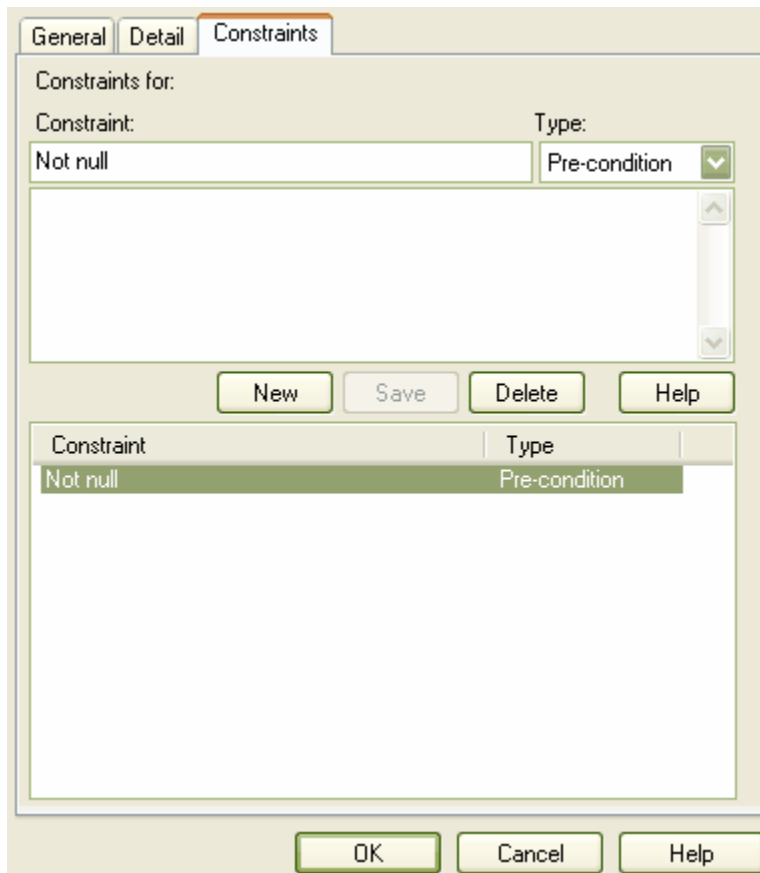
6.2.3.1.2 Attributes Detail

The *Detail* tab of the *Attributes* dialog has some additional details relating to collections.

| Control | Description |
|---------------------------|--|
| Lower Bound | A lower limit |
| Upper Bound | An upper limit to the number of elements in the collection |
| Ordered Multiplicity | Set if the collection is ordered |
| Attribute is a Collection | Check if the attribute is a collection |
| Allow Duplicates | Set if duplicates are allowed |
| Container Type | The container type |
| Save | Save changes |

6.2.3.1.3 Attribute Constraints

Attributes may also have *Constraints* associated with them. Typically this will indicate such things as maximum value, minimum value, length of field etc.



| Control | Description |
|-----------------|---------------------------------------|
| Constraint | Constraint name |
| Type | Constraint type |
| Notes | Constraint details |
| Constraint list | A list of constraints already defined |
| New | Create new attribute constraint |
| Save | Save new constraint details |
| Delete | Delete currently selected constraint |
| Help | Opens this help document |

6.2.3.1.4 Attribute Tagged Values

An attribute may have *Tagged Values* defined for it. Tagged values are a convenient means of extending the properties a model element supports. This in turn can be used by code generators and other utilities to transform UML models into other forms.

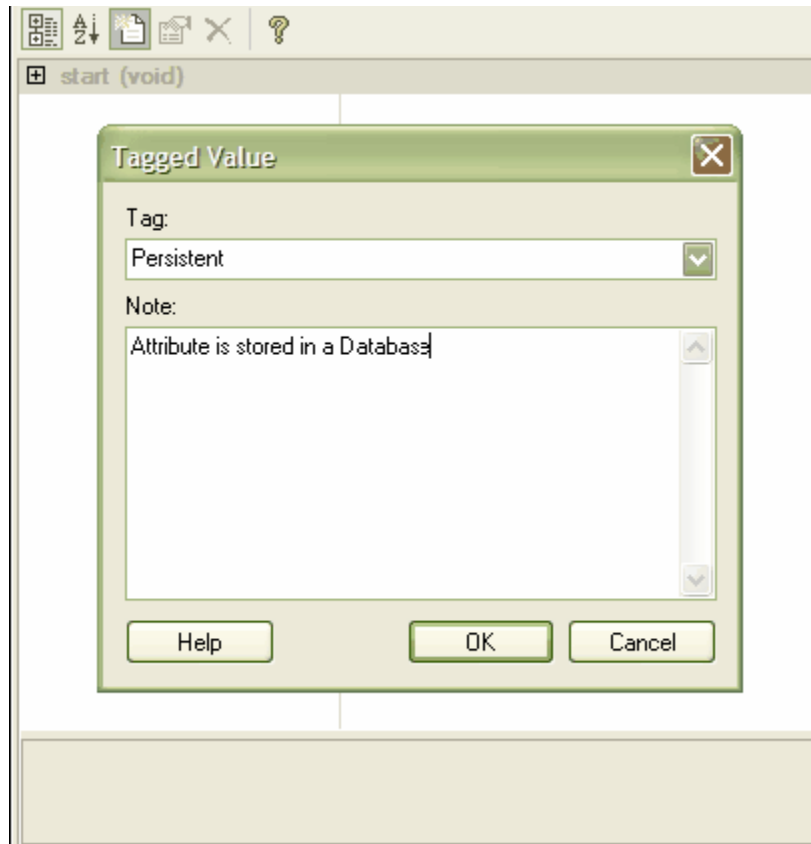
Tip: Tagged values are supported for Attributes, Operations, Objects and Connectors.

Add a Tagged Value

To add a tagged value for an attribute, use the following steps :

1. Ensure the *Tagged Values* window is open by selecting **View | Tagged Values** (or press the **Ctrl + Shift + 6** hotkey combination).

2. Select the attribute by double clicking on the attribute in a diagram or on the attribute in the Project View.
3. The *Tagged Values* window will now have the attribute selected, press either the *New Tags* button or the *Ctrl + N* hotkey combination.
4. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
5. Then press *OK* the button to confirm the operation.



Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

6.2.3.1.5 Creating Properties

EA has some capabilities for automatically creating properties in various languages. Property creation is controlled from the *General* tab of the *Attribute* dialog. Select the *Property* option to activate this feature.

Name: TakeNeed
Type: boolean Derived Static
Scope: Private Property Const
Stereotype:
Containment: Not Specified
Initial:
Notes: Private declarations
Attributes

This opens the *Create Property Implementation* dialog (shown below). By default the class language is picked up as the default, however you may change this and generate for any language. Each language has slightly different syntax and generates slightly different results. For example, Java and C++ generate get and set functions, C# and VB.Net create property functions and Delphi creates get and set functions as well as a specialized Delphi property tagged value.

Name: City
Type: String Derived Property
Scope: Public
Stereotype:
Create Property Implementation
Language
 C++
 Java
 Visual Basic
 C#
 Delphi
 VB Net
Property Details
Name: City
Getter: GetCity()
Setter: SetCity(Value: String)
Stereotype: Published
Get Scope: Public Set Scope: Public

Enter your required details and press *OK*. EA will generate the required operations and/or properties to comply with the selected language. Note that get and set functions will be stereotypes with <<property get>> <<property set>> etc. making it easy to recognize property functions. You may also hide these specialized functions by checking the *Hide Properties* check box in the *Diagram Properties* dialog for a specific diagram. This makes it easier to view a class, uncluttered by many get and set methods.

The screenshot shows the 'Appearance Options' dialog box. It is organized into three main sections:

- Appearance Options:** A list of 15 checkboxes. The first three are checked: 'Use Stereotype Icons', 'Scale Printing to 1 Page', and 'Show Page Border'. The remaining 12 are unchecked.
- Visible Class Members:** A separate section with 4 checkboxes. All four are checked: 'Public', 'Protected', 'Private', and 'Package'.
- Show Parameter Detail:** A dropdown menu currently set to 'Type Only'.

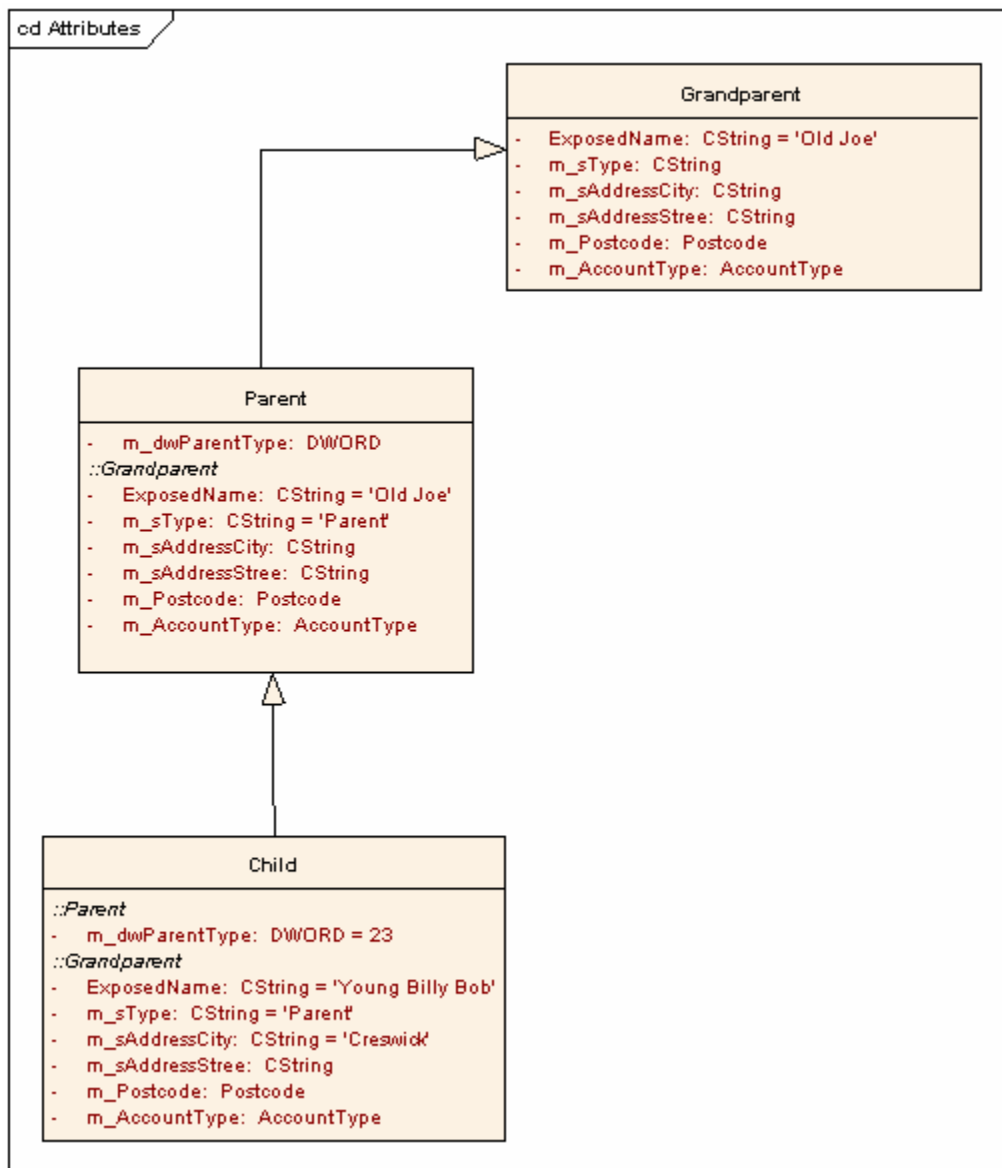
| Section | Option | Checked |
|-----------------------|-----------------------------|-----------|
| Appearance Options | Use Stereotype Icons | Yes |
| | Scale Printing to 1 Page | Yes |
| | Show Page Border | Yes |
| | Use Alias if Available | No |
| | Hide Property Methods | No |
| | Hide Collaboration Numbers | No |
| | Hide Element Stereotype | No |
| | Hide Qualifiers | No |
| | Highlight Foreign Objects | Yes |
| | Show Package Contents | Yes |
| | Show Details on Diagram | No |
| | Show Sequence Notes | No |
| | Hide Additional Parents | No |
| | Hide Relationships | No |
| | Hide Stereotype on Features | No |
| Show Table Owner | No | |
| Visible Class Members | Hide Attributes | No |
| | Hide Operations | No |
| | Show Tags | No |
| | Show Requirements | No |
| Show Parameter Detail | Show Constraints | No |
| | Show Testing | No |
| | Show Maintenance | Yes |
| | Dropdown Menu | Type Only |

Note that for Delphi you need to enable the 'tagged values' compartment to see the generated properties. See [Compartments](#) for how to do this.

6.2.3.1.6 Displaying Inherited Attributes

When displaying a class with attributes in a diagram, it is possible to also show the inherited Attributes from all parents in the elements type hierarchy (ancestors).

To show inherited Attributes, use the [Specify Feature Visibility](#) dialog.



Note that it is also possible to override an inherited attributes "initial value". This is done using the context menu item "Set Attribute Initializers" from the element context menu. This only applies to elements which have Attributes. In the Attribute Initializer dialog, select the variable name and enter a new initial value. An optional note may also be entered. When you display inherited Attributes, EA will merge the list of Attributes from all ancestors and also merge the Attribute initializers, so that the final cchildclass displays the correct Attribute set and initial values.

6.2.3.2 Operations

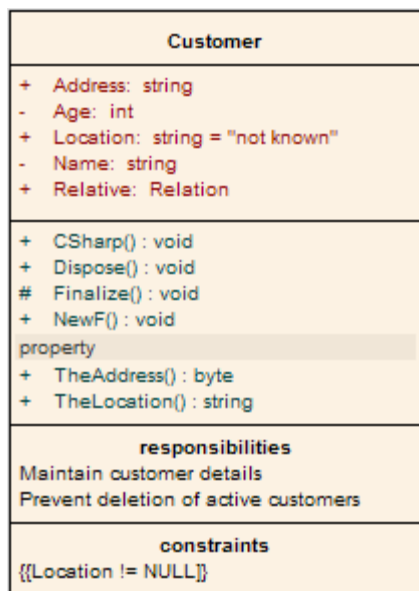
Operations are features of a class or other element that represent the behavior or services an element supports. For a Customer class, UpdateCustomerName and GetCustomerAddress may be operations. Operations have several important characteristics, such as type, scope (visibility), static, abstract and notes.

How to Access Operations

If an element supports operations (typically classes and interfaces), the *Element Features* submenu of the right click context menu will contain the *Operations* item. Select this to open the *Operations* dialog. Alternatively, press **F10**. For more information regarding this dialog, see below.

How Operations Appear in Diagrams

Elements with operations (typically classes) will display their features in diagrams in the manner shown below. As the diagram illustrates, some characteristics will display in shorthand form, for example 'static' will display as '\$', abstract as '*'

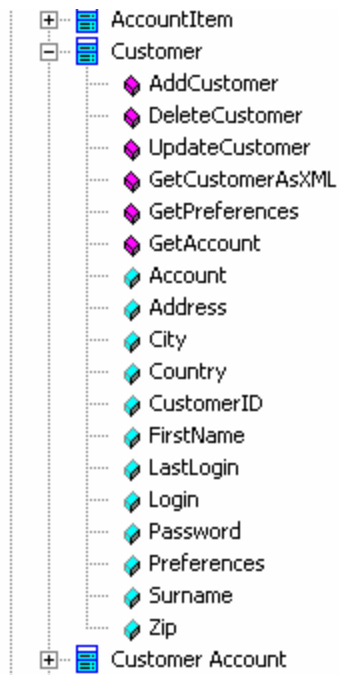


Operations in the Project Browser

Classes with operations will have their features collected beneath them in the Project Browser. Right click on an operation and select *Operation Properties...* to open the *Operations* dialog and edit details for the

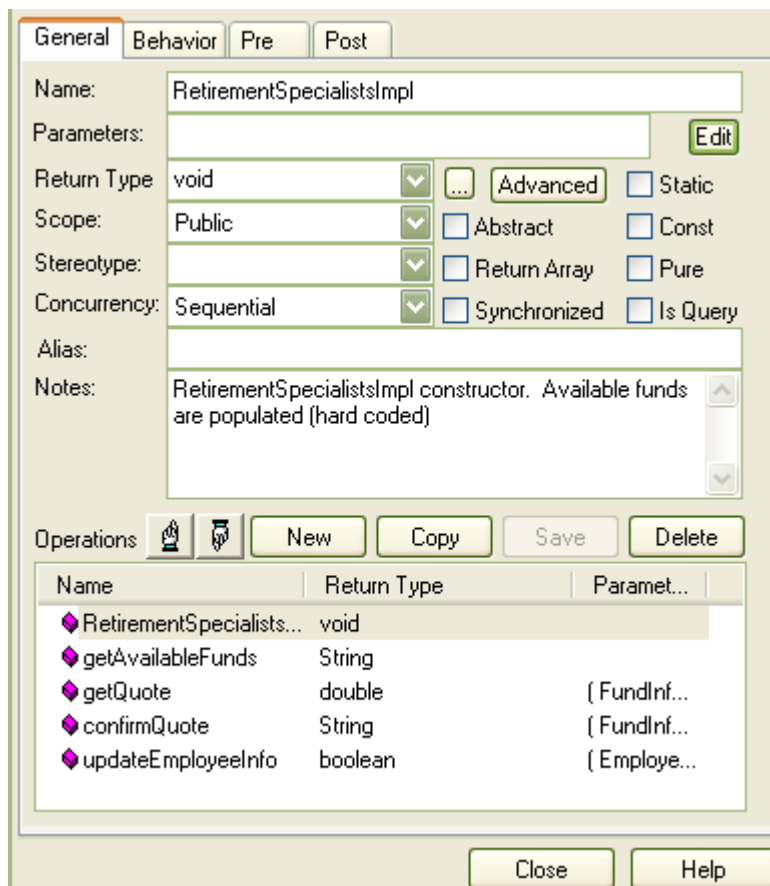
feature.

From the Project Browser, you can drag operations onto new Elements to give it the same operation.



6.2.3.2.1 Operations Main Page

The *General* tab of the *Operations* dialog allows you to define new operations and set the most common properties, including name, access type, return etc.



Tip: To go directly to the parameters page, double click an operation in the operations list.

| Control | Description |
|--------------------------------|--|
| Name | Operation name |
| Parameters | The parameter list, see Operation Parameters for information regarding what this string can contain. |
| Parameters Build Button (Edit) | Opens the Parameter Builder dialog |
| Return Type | Data type returned by operation |
| Return Type Build Button | Opens the <i>Set Element Classifier</i> dialog |
| Scope | Public/Protected/Private/Package |
| Stereotype | An optional stereotype for this operation |
| Concurrency | Concurrency of operation |
| Alias | An optional alias for the operation |
| Notes | Free text notes |
| Virtual/Abstract | If the operation's language is set to C++, this option maps to the C++ Virtual keyword. Otherwise this option is Abstract, pertaining to an abstract function. |
| Return Array | The return value is an array |
| Synchronized | A code engineering flag which relates to multi threading in Java |
| Static | Operation is a static member |
| Const | The return type of this method is constant |

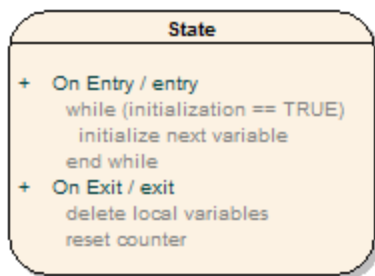
| | |
|-----------------|---|
| Pure | Relates to C++ pure virtual syntax - eg. virtual void myFunction() = 0; |
| IsQuery | This method does not modify the object |
| Operation List | List of defined operations |
| Up/Down Buttons | Use to change the order of operations in the list |
| New | Create new operation |
| Save | Save new operation, or save modified details for existing operation |
| Delete | Delete currently selected operation |

6.2.3.2.2 Operations Detail

The *Behavior* tab of the *Operations* dialog allows you to enter free text to describe the functionality that an operation will have. Use pseudo code, structured English or just a brief description.

You can also use this field to formally describe a Method or State action and have the text appear under the method/action name in a diagram.

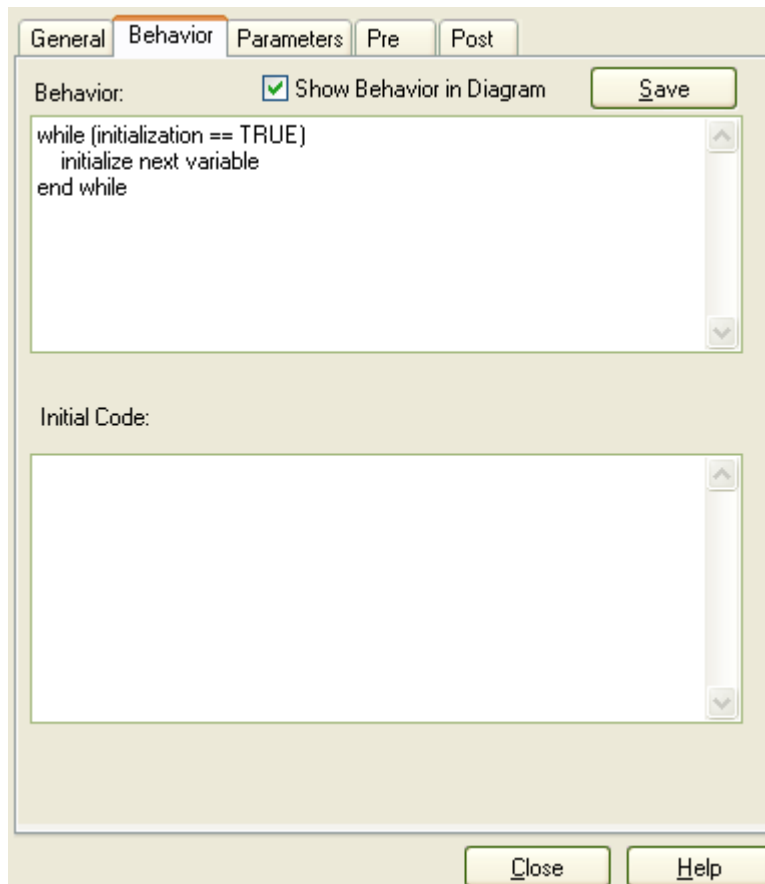
This example illustrates how to use this field to elaborate a method's function in a diagram.



Showing Behavior in a Diagram

To show behavior in a diagram, follow the steps below:

1. Create or locate the required operation.
2. Go to the *Behavior* tab of the *Operations* dialog.
3. Select the *Show Behavior in Diagram* option.
4. Press *Save*.

**See Also**

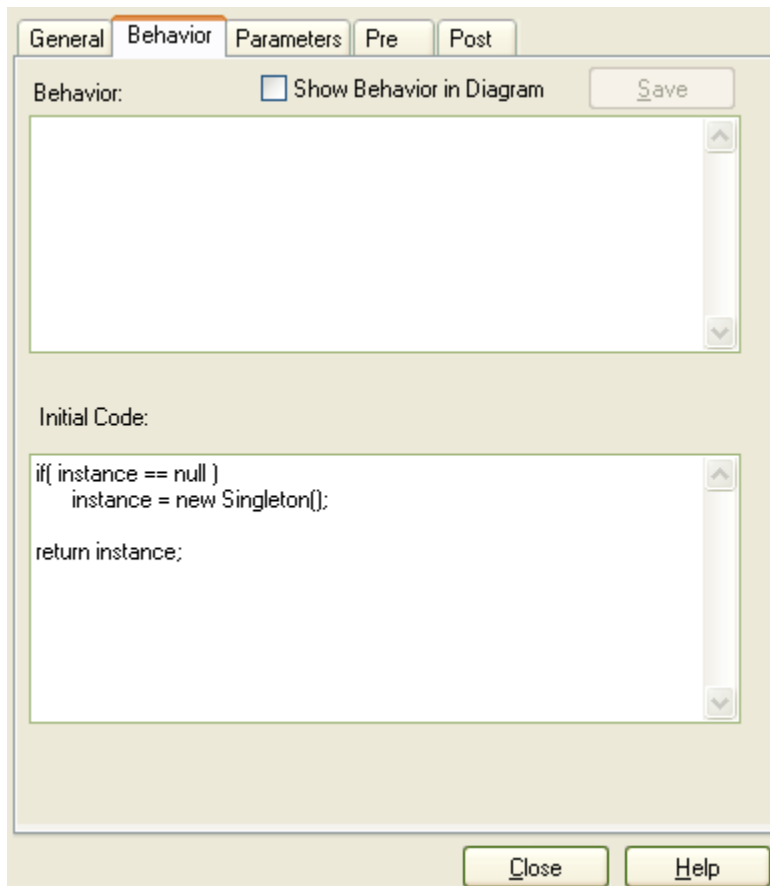
- [Initial Code](#)

6.2.3.2.1 Initial Code

The Initial Code field inserts code into an operation body when the operation is first generated to file. After this point, forward code generation will not replace the existing operation code with the Initial Code field. It should also be noted that the Initial Code field is not imported into the model during reverse engineering (or synchronization).

This field is most useful when combined with UML Patterns. Elements within a pattern often require the same stub code. You will notice that the language specific patterns available from www.sparxsystems.com.au/resources/developers/uml_patterns.html include initial code for some of the defined operations. This helps speed up the process of applying patterns from model to implementation. The Initial Code section is also useful for ensuring that the generated code is directly compilable.

This example shows the contents of the Initial Code field for the Instance() operation of the Singleton element in the C# Singleton pattern:

**See Also**

- [UML Patterns](#)

6.2.3.2.3 Operation Parameters

The *Parameters* tab in the *Operations* dialog lets you define the parameters an operation will have. The parameter list will be reproduced in code in the order they appear in the parameters list - so use the up and down arrows to move parameters into their required positions. Additionally, you may select the *Add new to end* option to force new parameters to appear at the end of the list instead of the head.

Tip: Set the amount of parameter detail to display in a specific diagram using the *Show Parameter Detail* drop down list on the *Diagram Properties* dialog. The setting applies only to the current diagram. The default is to show the type only.

| Control | Description |
|----------------|--|
| Name | Parameter name |
| Type | Data type of parameter |
| Default | Optional default value |
| Kind | Indicates the way a parameter is passed to a function <ul style="list-style-type: none"> • In = By Value • InOut = By Reference • Out is passed by Reference - but only the return value is significant |
| Fixed | The parameter is 'const' - even if passed by reference |
| Add new to end | Place new parameters at the end of the list instead of the start |
| Notes | Free text |

6.2.3.2.3.1 Operation Parameter Tagged Values

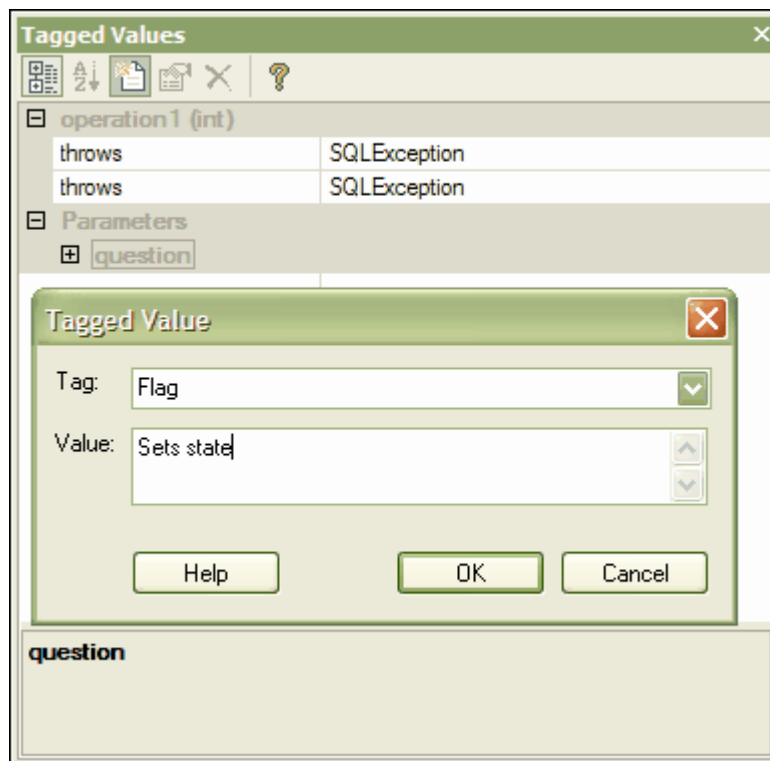
Operation parameters may have tagged values associated with them. Tagged values offer a convenient extension mechanism for UML elements - so you can define any tags you like - and then assign them values using this form.

Tagged values will be written to the XMI output, and may be input to other third party tools for code generation or other activity.

Add a Tagged Value

To add a tagged value for a parameter, use the following the steps :

1. Ensure the *Tagged Values* window is open by selecting *View | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the operation by double clicking on the operation containing the parameter in the diagram or on the operation containing the parameter in the Project View.
3. The *Tagged Values* window will now have the operation with parameter selected,
4. Select the parameter from the Parameters compartment in the Tagged Values window press either the *New Tags* button or the *Ctrl + N* hotkey combination.
5. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
6. Then press *OK* the button to confirm the parameter tag.

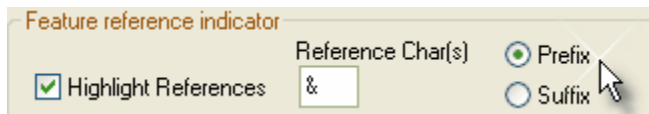


Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

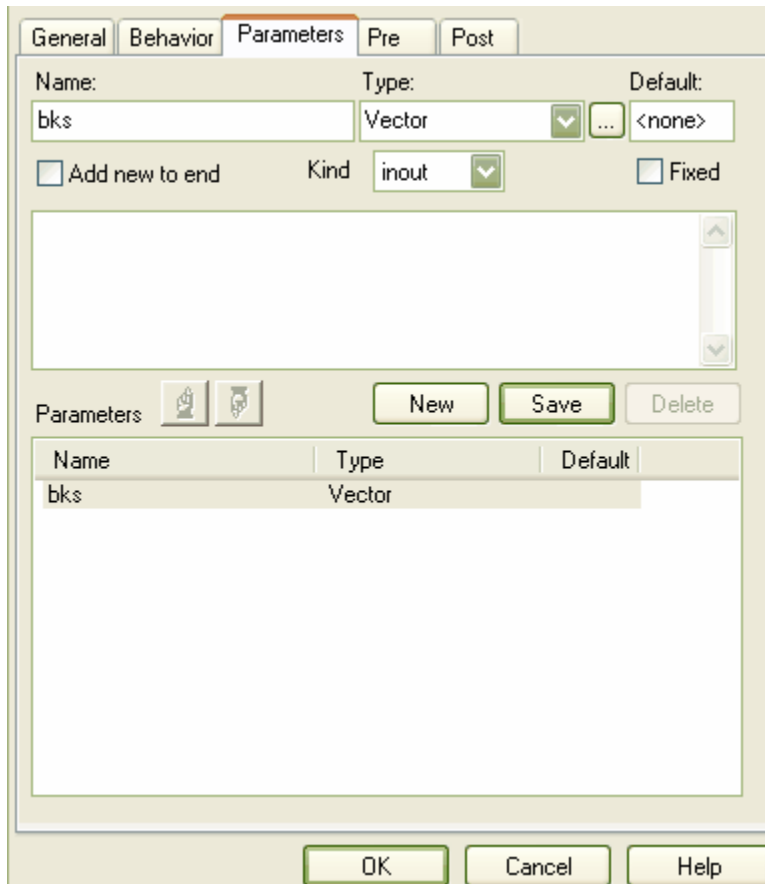
6.2.3.2.4 Operation Parameters by Reference

You can elect to highlight parameters declared as type 'inout' with an additional user-defined prefix or suffix. In the *Objects* section of the *Local Options* dialog (*Tools | Options*), there is a segment which allows you to set whether references are highlighted or not.

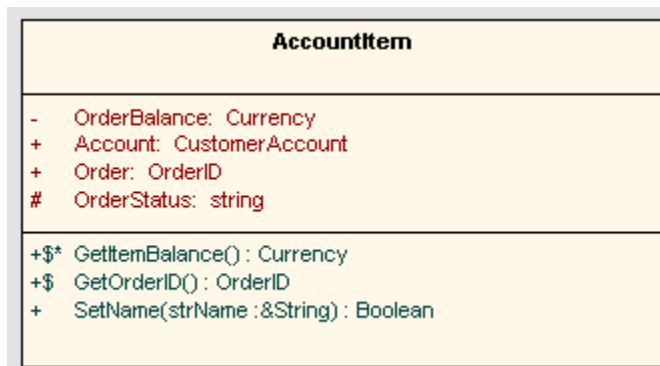
If you select the *Highlight References* option, you can also indicate whether a prefix or suffix will be used, and the actual character to use. In the example below, the '&' character as a prefix has been selected.



When you declare a parameter of type 'inout', it is assumed you are passing the parameter by reference, rather than by value. If you have elected to highlight references, then this will be displayed in the diagram view.



The example below shows that the parameter `strName` is a String reference - and is highlighted using the chosen character and position.



6.2.3.2.5 Operation Constraints

Operations may have pre- and post- conditions defined. For each type, give the condition a name, a type and enter notes.

Constraints define the contractual behavior of an operation - what must be true before they are called and what is true after. In this respect they are related to the state model of a class and can also relate to the guard conditions that apply to a transition.

The screenshot shows a dialog box with tabs for 'General', 'Behavior', 'Parameters', 'Pre', and 'Post'. The 'Pre' tab is active. It contains a 'PreCondition:' label, a text field with 'Some Precondition', and a 'Type:' label with a dropdown menu showing 'Book'. Below this is a large empty text area. At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

| Pre-Condition | Type |
|---------------|------|
|---------------|------|

6.2.3.2.6 Operation Tagged Values

Operations may have tagged values associated with them. Tagged values offer a convenient extension mechanism for UML elements - so you can define any tags you like - and then assign them values using this form.

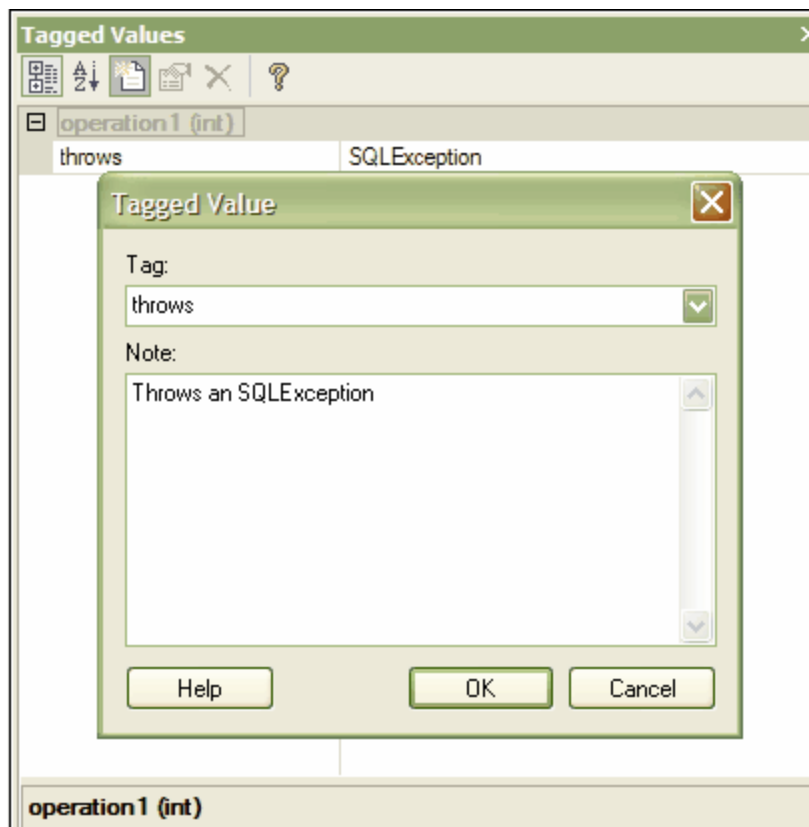
Tagged values will be written to the XML output, and may be input to other third party tools for code generation or other activity.

Tip: Tagged values are supported for Attributes, Operations, Objects and Connectors.

Add a Tagged Value

To add a tagged value for an operation, use the following steps :

1. Ensure the *Tagged Values* window is open by selecting *View | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the operation by double clicking on the operation in a diagram or on the operation in the Project View.
3. The *Tagged Values* window will now have the operation selected, press either the *New Tags* button or the *Ctrl + N* hotkey combination.
4. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
5. Then press *OK* the button to confirm the operation.

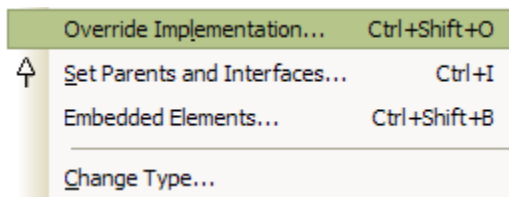


Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

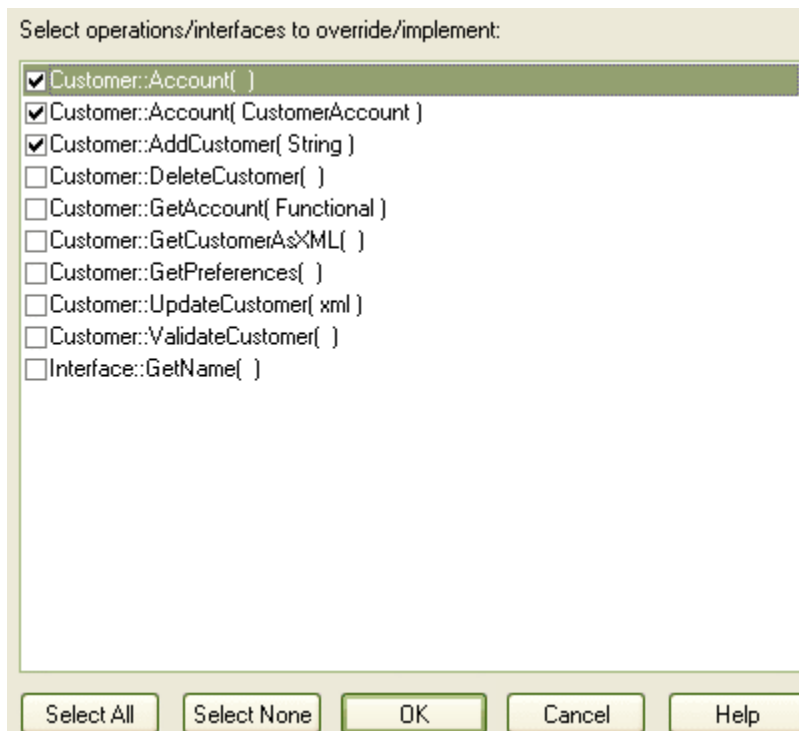
6.2.3.2.7 Override Parent Operations

It is possible in EA to automatically override methods from parent classes and from realized interfaces.

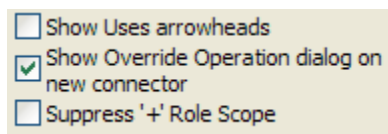
Select a class that has a parent or realized interface and from the *Main Menu*, select *Element | Advanced | Override Implementation*.



In the *Override Operations/Interfaces* dialog check the operations/interfaces that you wish to automatically override and press **OK**. EA will generate the equivalent function definitions in your child class.



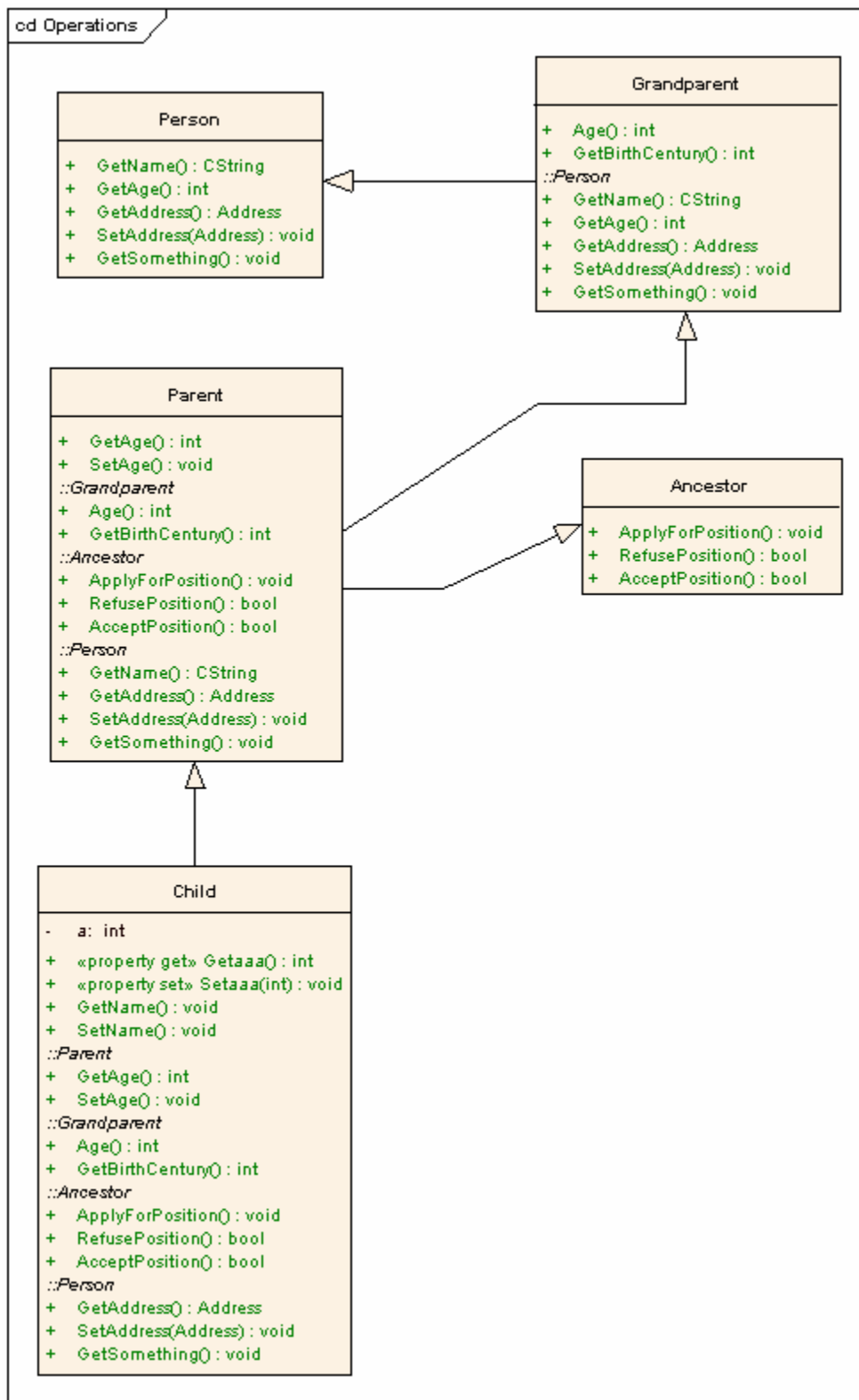
It is possible to configure EA to display this dialog each time you add a Generalization or Realization link between classes and their possible operations/interfaces to override/implement. Do this from the *Diagram* section of the *Local Options* dialog (*Tools | Options*).



6.2.3.2.8 Displaying Inherited Operations

It is possible to configure an element in a diagram to display the complete Operation set obtained from all ancestors in the elements type hierarchy, as well as those directly owned. To do this, use the [Specify Feature Visibility](#) function from the main menu, or use the Ctrl+Shift+Y shortcut key.

The following diagram illustrates this behavior when enabled for each element in a simple hierarchy.



6.2.4 Element Properties

This topic covers element properties and their settings, responsibilities, constraints, links, scenarios, tagged values, associated files, object files and classifiers and boundary element settings.

See: [Properties](#)

6.2.4.1 Properties

Below is the *Element Properties* dialog:

The screenshot shows the 'Element Properties' dialog box with the 'General' tab selected. The 'Name' field contains 'Login'. The 'Stereotype' field is empty. The 'Author' field contains 'John Redfern'. The 'Scope' field contains 'Public'. The 'Alias' field is empty. The 'Phase' field contains '1.0' and the 'Version' field contains '1.0'. The 'Status' dropdown is set to 'Proposed', 'Complexity' to 'Easy', and 'Language' to '<none>'. There is an unchecked checkbox for 'Abstract'. The 'Note' field contains the text: 'The on-line book store needs to be able to log on to the system. The connection to the system will be over HTTP (SSL)'. At the bottom, there are buttons for 'Apply', 'OK', 'Cancel', and 'Help'.

Any one of the following options allows you to view this dialog:

- Select an element in the diagram view and select *Properties...* from the *Element* menu.
- Right click on an element in the diagram view, and select *<element type> Properties* from the context menu.
- Select an element in the diagram view, and press *Alt+Enter*.
- Double-click on an element in the diagram view.
- Right click on an element in the Project Browser, and select *Properties...* from the context menu.

To suppress the Properties dialogue from appearing when placing a new element, uncheck the *Edit Object on New option*.

The following topics describe each of the tabs in this dialog in detail.

- [General](#)
- [Require](#)
- [Constraints](#)
- [Link](#)

- [Scenario](#)
- [Files](#)

6.2.4.2 General Settings

The *General* tab of the *Element Properties* dialog is shown below:

From here you can do the following:

| Control | Description |
|-----------------|--|
| Name | Change the element's name |
| Stereotype | Select a stereotype for the element (optional) |
| Abstract | Check to indicate element is abstract |
| Author | Enter or select the name of the original author |
| Status | Indicate the current status of the element (eg. Approved, Proposed...) |
| Scope | Indicate the element's scope (public, private, protected, package) |
| Complexity | Indicate the complexity of the element (used for project estimation). Assign Easy, Medium or Hard |
| Alias | Enter an alias (alternate display name) for the object. |
| Language | Select the programming language for the object |
| Keywords | A free text area that may be filtered in Use Case Metrics and Search dialogs - typically used for keywords, context information etc. |
| Advanced button | See Advanced Settings for details |
| Phase | Indicate the phase this element will be implemented in (eg. 1, 1.1, 2.0 ...) |
| Version | Enter the version of the current element |

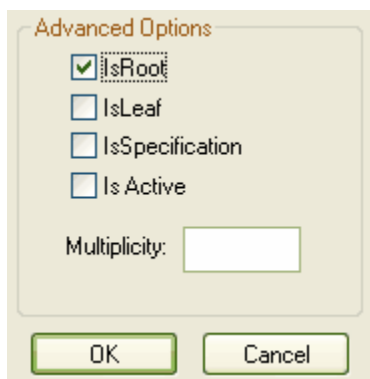
Notes

Enter any free text notes associated with the element

6.2.4.2.1 Advanced Settings

Some elements support some additional attributes. These are Generalizable elements - and by pressing *Advanced* on the *Element Properties* dialog you can set the following:

- *IsRoot* - the element is a root element and may not be descended from another
- *IsLeaf* - the element is 'final' and may not be a parent for other elements
- *IsSpecification* - the element is a specification
- *IsActive* - the element is active - eg. an [active class](#)
- *Multiplicity* - multiplicity setting for the element



6.2.4.3 Requirements

The *Require* tab of the *Element Properties* dialog is shown below. Use this page to create requirements that this element is designed to meet. Requirements are of two types - internal requirements (responsibilities) and external requirements (system requirements). EA will show both types, but you can only edit the internal type from this tab.

These form the functional requirements of the system to be built. The meaning of the requirement may vary depending on which element is the host - for example a business process requirements may mean something different to a Use Case requirement, which again may mean something different to a Class requirement. Use these as best suit your needs.

Use the [Specify Feature Visibility](#) function to show Requirements for an element on the diagram directly (it is also possible to show inherited Requirements in this fashion).

Note: [External requirements](#) are those linked to this element using a *Realization link*

The screenshot shows the 'Require' tab of the 'Element Properties' dialog. The 'Requirement' field contains '1. Login to System' and the 'Type' is set to 'Functional'. The 'Status' is 'Approved', 'Difficulty' is 'Medium', 'Priority' is 'Medium', and 'Last Update' is '1/11/2004'. Below these fields is a list of 'Defined' requirements:

| Requirement | Type | External |
|---|------------|----------|
| 1. Login to System | Functional | |
| 1.01 | Functional | Yes |
| 2. Go to the Registered screen for non- ... | Functional | |
| 3. Encrypt user details over HTTPS | Functional | Yes |
| 4. Logout of system | Functional | |

| Control | Description |
|----------------------|--|
| Requirement | Name and high level detail of requirement |
| Type | Functional, non-functional etc. |
| Status | Current status of requirement |
| Difficulty | Complexity of implementing current requirement |
| Priority | How urgent the requirement is |
| Last update | Date of last requirement update |
| Notes | Details of requirement |
| Defined requirements | List of defined requirements associated with element |

6.2.4.3.1 External Requirements

External requirements are those requirement elements that have been linked to the current element using a Realization link. By creating the link from the element to the requirement, the element now has a responsibility that it must implement as part of the system solution.

In Enterprise Architect, linked requirements are shown in the *Require* tab of the *Element Properties* dialog - but they are marked external and cannot be directly edited (on selection, the tab fields are grayed out).

Double click an external requirement in the list to activate the *Properties* dialog for the associated requirement, where you can view and modify the requirement details and check the requirement hierarchy details.

Properties Files

Short Description: 1.01 Log on to the website

Status: Approved Type: Functional

Difficulty: Medium Phase: 1.0

Priority: Medium Last Update: 1/11/2004

Author: John Redfern Created: 1/11/2004

Version: 1.0

Details:

The proposed system will allow a user to log on to the web site with a unique log on name and password. The session state for the current user will be maintained while the user is on-line or until there is a 30 minute period of inactivity.

If a user is not currently registered with the system, they will be required to register a login and password before proceeding.

OK Cancel Help

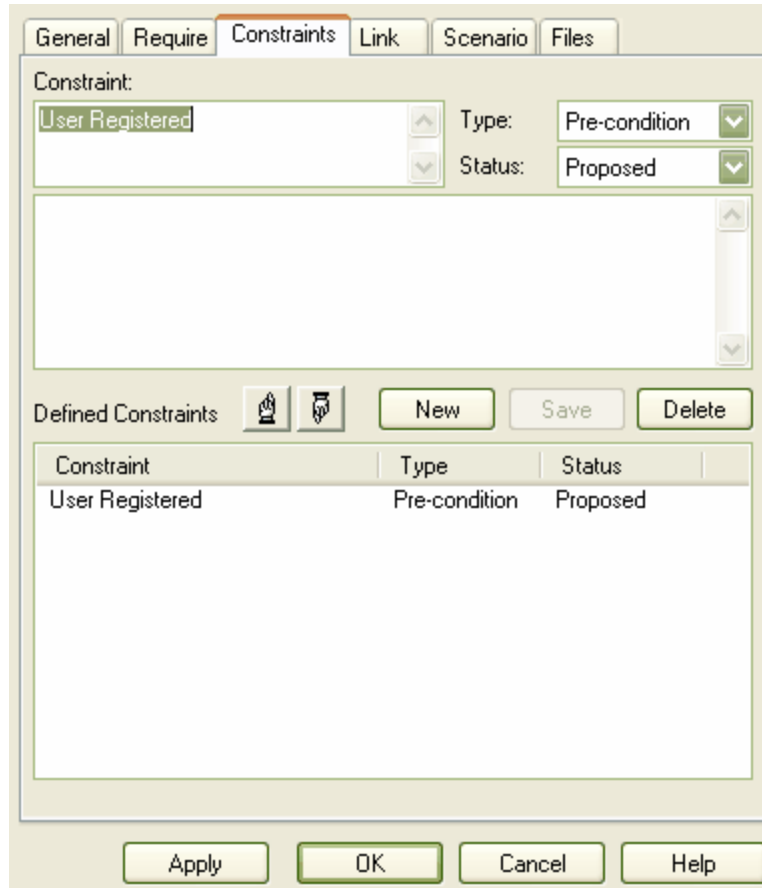
6.2.4.4 Constraints

The *Constraints* tab of the *Element Properties* dialog is shown below. Elements may have associated constraints placed on them. These are conditions under which the element must exist and function. Typical constraints are pre- and post- conditions which indicate things that must be true before the element is created or accessed and things which must be true after the element is destroyed or its action complete. Use the [Specify Feature Visibility](#) function to show Constraints for an element on the diagram directly (it is also possible to show inherited constraints in this fashion).

Adding Constraints to a Model Element

To add constraints to a model element, follow the steps below:

1. Open the *Element Properties* dialog.
2. Select the *Constraints* tab.
3. Enter constraint details.
4. Enter the *Type* and the *Status*.
5. Enter optional notes.
6. Press *Save*.

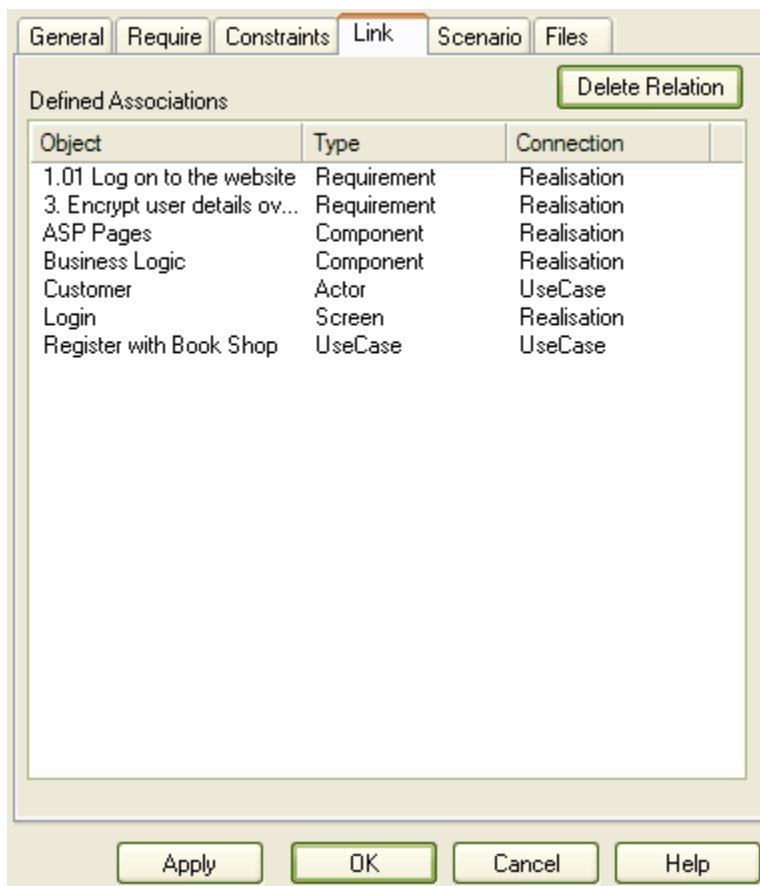


Constraints are used in conjunction with [responsibilities](#) to define the conditions and rules under which an element operates and exists.

| Constraint | Details of Constraint |
|--------------------|--|
| Type | Pre-condition, post-condition or invariant |
| Status | Current constraint status |
| Notes | Free text notes associated with constraint |
| Defined constraint | List of defined constraints |

6.2.4.5 Links

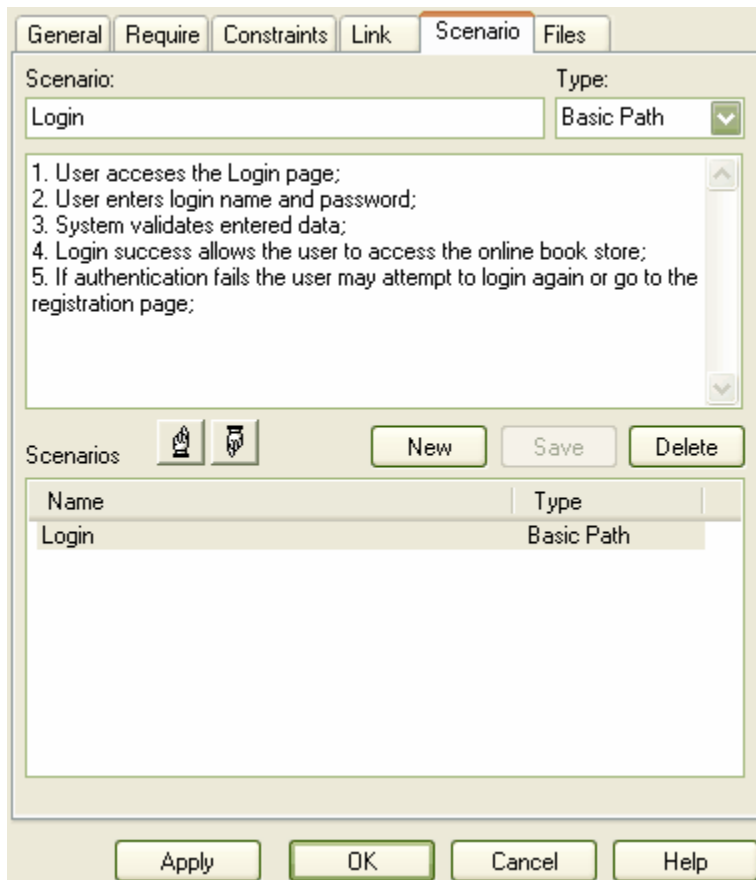
The *Links* tab of the *Element Properties* dialog is shown below. This displays a list of all connectors active for the current element. You may delete a connector here if required. Using the right click context menu you may also locate the related element in the Project Browser.



| Control | Description |
|------------------------|--|
| Defined Associations | List of relationships this element has |
| Delete relation button | Delete the selected relation |
| Object | The element this element is related to |
| Type | The type of the related element |
| Connection | The relationship type |

6.2.4.6 Scenarios

The *Scenario* tab of the *Element Properties* dialog is shown below. A scenario is a real world sequence of operations that describes how this element works in real-time. It may be applied to any element and can describe functional behavior, business work flows and end-to-end business processes.



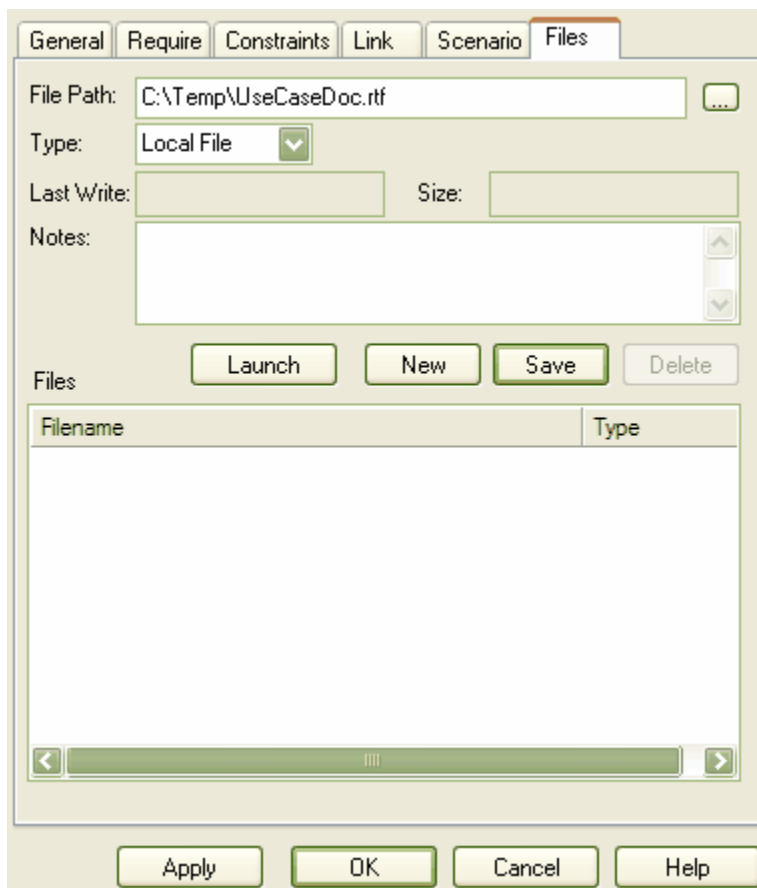
| Control | Description |
|---------------|--|
| Scenario | Name of scenario |
| Type | Basic path, alternate path etc. |
| Notes | Textual description of the scenario - usually depicted in steps of how the user will use the current element |
| Scenario list | List of defined scenarios |

6.2.4.7 Associated Files

The *Files* tab of the *Element Properties* dialog is shown below. An element may be linked to files held somewhere. Use this tab to set associated files for the current element.

Tip: *Linked files are a good way to link elements to additional documentation and/or source code.*

Note that you can also insert [hyperlinks](#) in diagrams to other files - and launch them directly from the diagram. This is an alternative method to that described here.



| Control | Description |
|----------------|--|
| File path | Name of file |
| Type | Local file or web address |
| Notes | Free text about the file |
| Attached files | List of file |
| Launch | Open the selected file - local files will open with their default application and web files will open in the default browser |

6.2.4.8 Tagged Values

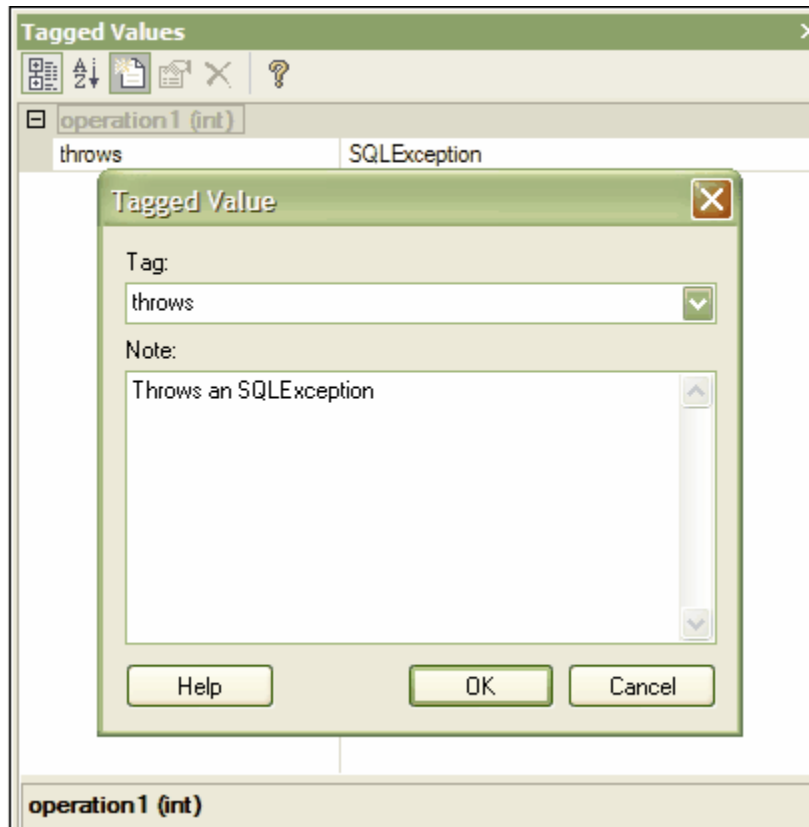
Tagged values are a convenient way of adding additional information to an element outside that directly supported by UML. The UML provides the Tagged Value element for just this purpose. Often these are used during code generation or by other tools to pass information or operate on elements in particular ways. For more information relating to using tags see [The Tagged Values Window](#) topic.

Add a Tagged Value

To add a tagged value for an element, use the following steps :

1. Ensure the *Tagged Values* window is open by selecting *View / Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the element by double clicking on the element in a diagram or on the element in the Project View.

3. The *Tagged Values* window will now have the operation selected, press the *New Tag* button.
4. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
5. Then press *OK* the button to confirm the operation.



Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

Tagged Values are the preferred method of extending the code generation capabilities of the CASE tool on a per element / per language basis. In future, EA will support more tagged values for different languages.

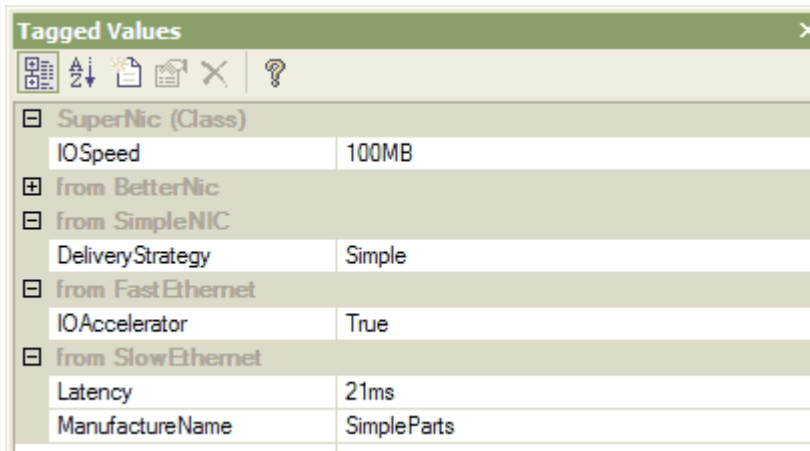
6.2.4.8.1 Advanced Tag Management

Tagged Values can also be managed within a type hierarchy and with respect to element instances. This additional management is done using the *Tag Management* dialog.

Using the *Tag Management* dialog it is possible to:

1. View tagged values inherited from parent classes or realized interfaces or applied stereotypes.
2. Override Tagged Values derived from parents or applied stereotypes with a unique value for the current element.
3. Delete tagged values from the current element (if a parent version of the Tagged Value exists, it will re-

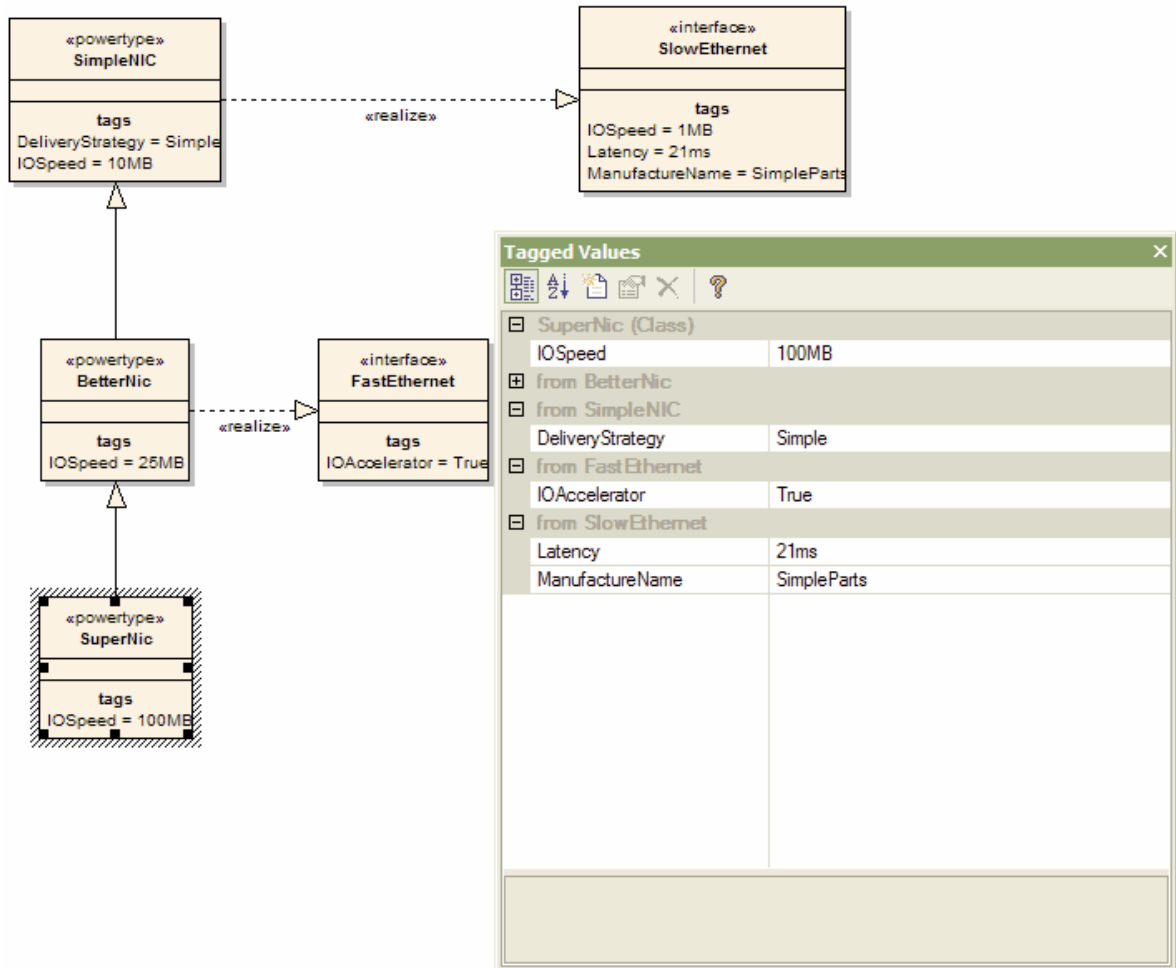
appear in the list after the override is deleted).



| Tagged Values | |
|-------------------|-------------|
| SuperNic (Class) | |
| IOSpeed | 100MB |
| from BetterNic | |
| from SimpleNIC | |
| DeliveryStrategy | Simple |
| from FastEthernet | |
| IOAccelerator | True |
| from SlowEthernet | |
| Latency | 21ms |
| ManufactureName | SimpleParts |

The diagram below illustrates a complex tag hierarchy and the way tagged values can be either inherited or overridden in specialized classes to create the final tagged property set for an element.

Note also that a similar concept applies to instances - in which case the full tag set is created from the directly owned tags, plus all of those merged in from the classifiers type hierarchy, additional stereotypes and realized interfaces.



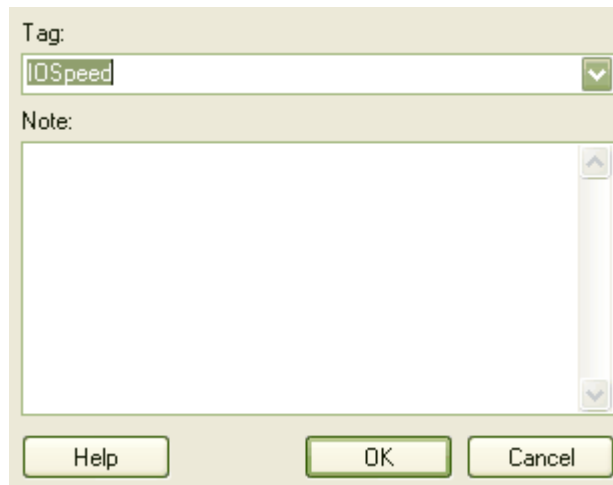
6.2.4.8.2 Quick Add of Tagged Values

It is possible to add a single tagged value to one or more elements with a special shortcut.

1. From an element context menu (or the context menu of a multi-selection) - choose the **Add | Tagged Value...** option. (Alternatively, select one or more elements and press **Shift+Ctrl+T**).
2. The Tagged Values window will appear - press the New Tag button,
3. This will allow the entry of a **Name for the tag** (required) and **Notes**.
4. To add a value for the tag by entering a value in the values section of the **Tagged Values Window** (which will be initially blank).
5. Press **OK** to add your new tagged value to all the currently selected elements.

Note: You can also use the Current Element toolbar ... the last button is a shortcut to the **Add Tagged Value** function.

To delete this property you must open the element property dialog, go to the **Tagged Values** tab and manually delete the item. There is currently no shortcut to delete tags from multiple elements at one time.



The Notes section will appear in the Tagged Values window in the Info section at the bottom Tagged Values window.

6.2.4.9 Object Classifiers

Many elements in UML model instances of classes - for example, objects, actors and sequence diagram objects. These elements represent real things in a run-time scenario - eg. a Person element named Joe Smith. In UML this is written as "Joe Smith : Person".

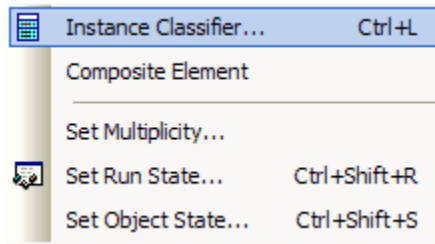
As the model develops from a rough sketch to a detailed design, many objects will become examples of defined classes - so in the early analysis phase you may model a "Joe Smith" and a "Jane Smith" - and later a "Person" class from which Joe and Jane will be instantiated.

Enterprise Architect allows you to associate an object with its template or class (its classifier). Doing this greatly increases the descriptive power of the model in capturing the functionality and responsibility of objects at run-time and their associated state. For example, if we describe a Person class with attributes such as Age, Name, Address and Sex, and functions such as GetAge, GetName etc., then when we associate our object with the Person class it is seen to have all the Person class behavior and state (as well as inherited state and behavior from Person's ancestors).

Tip: This is a powerful means of moving your model from the analysis phase into detailed design.

6.2.4.9.1 Using Classifiers

Objects which support classifiers have the option *Instance Classifier...* in the *Advanced* sub-menu of their diagram view context menu. Select this option to choose a single class as the classifier or template for this object.



When you do this, the object name will be displayed as "Object : Class" - for example a Person object named Joe Smith will be displayed as "Joe Smith : Person".

Several Changes Occur if an Object has a Classifier:

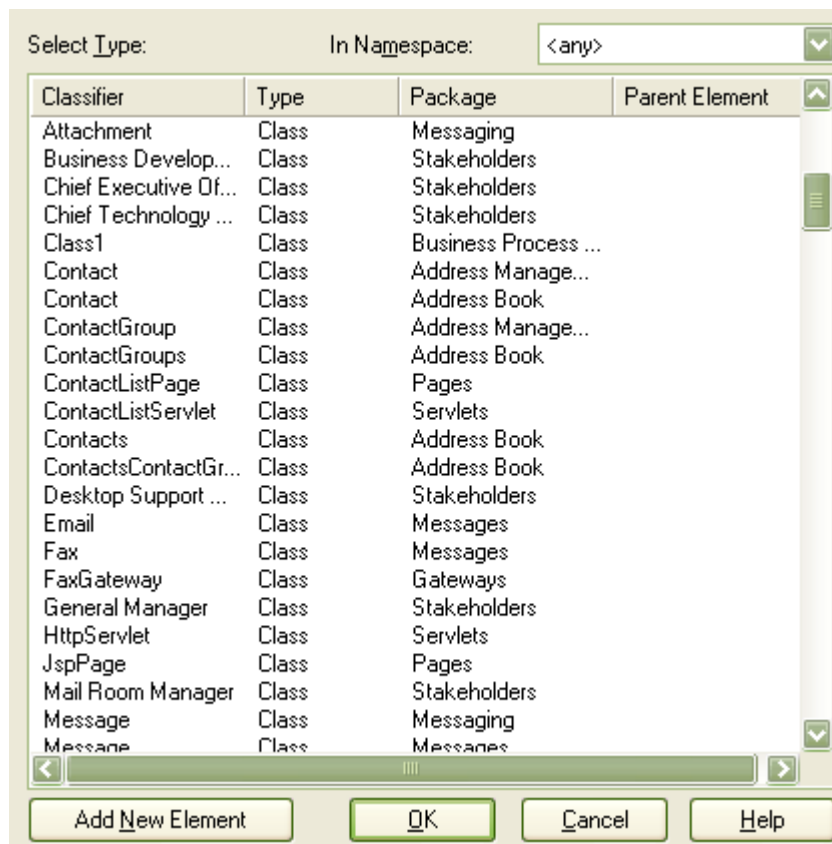
In the context menu for the object, the *Element Features | Attributes* and *Element Features | Operations* menu items will appear. If you select either of these, you will be given the *Attributes* or *Operations* dialog for the classifier, not the object. It is important to remember that the object is only an instance of a class at runtime, so the appropriate attributes and operations are those of the classifier, not the object.

If you set the classifier for an object in a sequence diagram, when you add a message the drop down list of available messages derived from the target object will come from the classifier - not the object selected. This allows you to associate sequence diagram objects with classes and use the defined behavior of the class to model actual behavior at run time.

You may also select a message for a state flow connector. The same rules apply as for sequence objects.

Note that in the message dialog you may also elect to include messages defined in the target classifiers inheritance hierarchy.

The *Set Element Classifier* dialog is shown below. Select the desired item in the list and press *OK* to set the instance classifier.



6.2.4.10 Boundary Element Settings

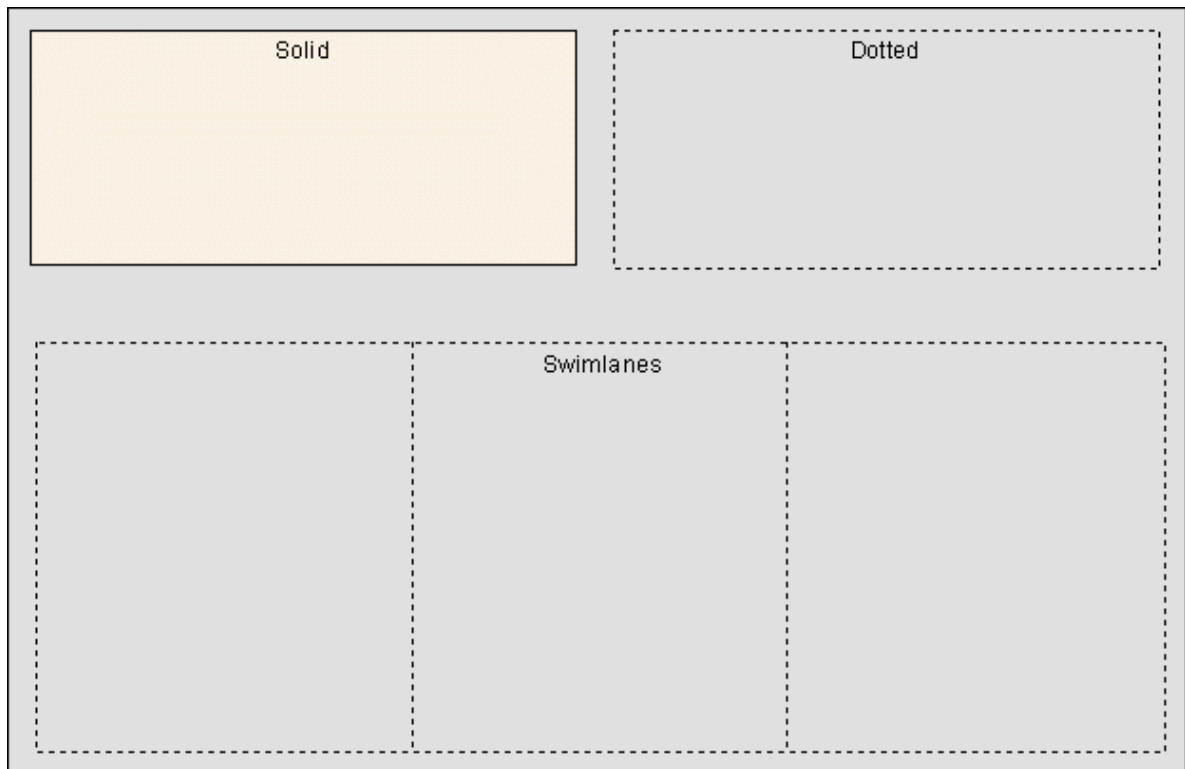
A system boundary element is a non-UML element used to define conceptual boundaries. You can use boundaries to help group logically related elements (from a visual perspective, not as part of the UML model)

Configuring Boundary Elements

Boundary elements may be configured to display in different ways. The main differences are:

- Solid border
- Dotted Border

With horizontal or vertical 'swim lanes' - swim lanes are used to group elements in a vertical or horizontal context (eg. Client, Application and Database tiers could be represented in swim lanes)



6.2.5 Compartments

In addition to the two main compartments shown in a class element (attributes and operations), EA also supports other compartments that may be optionally displayed.

To set the visibility of the various compartments, see [Set Feature Visibility](#).

See Also:

[Responsibility Compartment](#)

[Constraint Compartment](#)

[Tag Compartment](#)

Testing Compartment

Maintenance Compartment

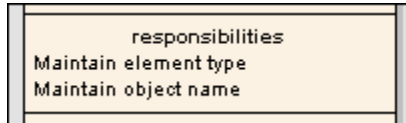
6.2.5.1 Tag Compartment

The Tag compartment lists all tagged values for an element as entered in the *Tagged Values* tab of the *Element Properties* dialog.



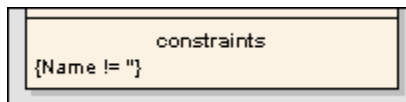
6.2.5.2 Responsibility Compartment

The responsibility compartment shows a list of responsibilities as entered on the *Responsibility* tab of the *Element Properties* dialog.



6.2.5.3 Constraint Compartment

The constraint compartment shows a list of element constraints as entered in the *Constraint* tab of the *Element Properties* dialog.



6.2.5.4 Testing Compartment

The testing compartment lists all of the tests associated with an element as listed in the Testing window (*View | Testing*).

6.2.5.5 Maintenance Compartment

The maintenance compartment lists all of the defects, changes, issues and tasks associated with an Element as listed in the Maintenance window (*View | Maintenance*).

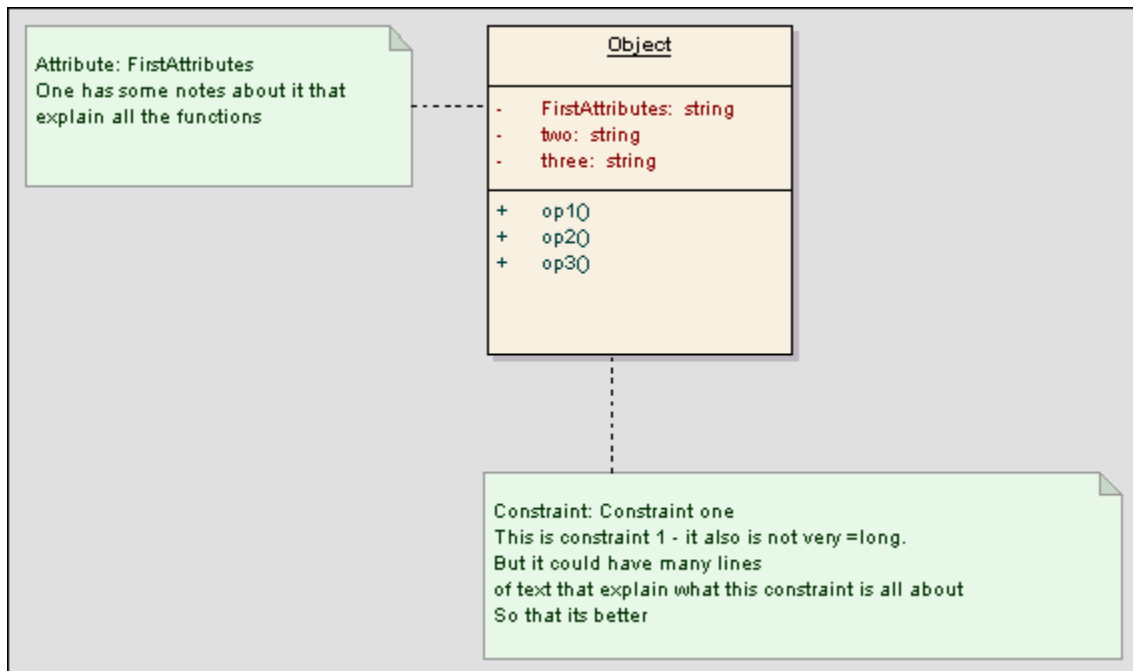
6.2.6 Linking Notes

In Enterprise Architect you can link notes to internal documentation or to an element.

6.2.6.1 Linking Notes to Internal Documentation

It is possible to link a note element to another element's internal documentation. This allows you to externalize model documentation to the diagram level, and as EA keeps the note and the internal structure in synch, you do not have to worry about updating the note contents - this is done automatically.

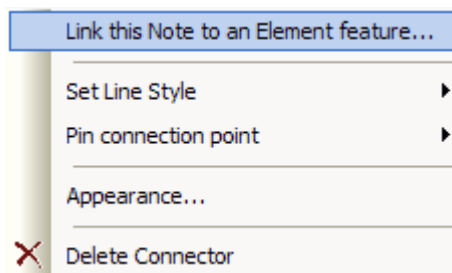
In the example below, two notes are linked into an element's internal structures. One is linked to an attribute, and displays the attribute name and notes. The other is linked to a constraint - showing the constraint name and documentation.



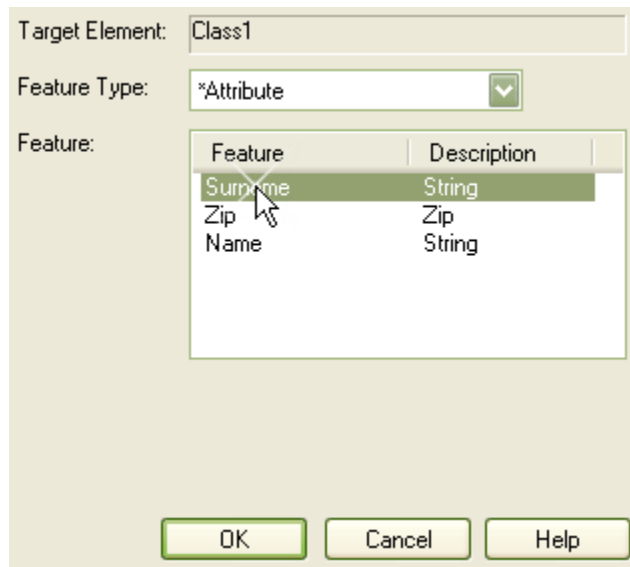
6.2.6.2 Link a Note to an Element

To link a note element (graphical text) to another design element, do the following:

1. Insert the target element into a diagram.
2. Add a note to the diagram, but do not add any text to it. To do this, right click somewhere on the diagram where you want the note to go. Select *Create Element or Connector | Common Elements | Note*.
3. Link the note and the element with a note link connector. The note link connector is located in the [UML Elements toolbar](#) under the *Link* drop down menu - it is the last item on the list.
4. Right click on the note link and select *Link this Note to an Element Feature*.



5. In the *Link note to element feature* dialog, select the *Feature Type* to link to.



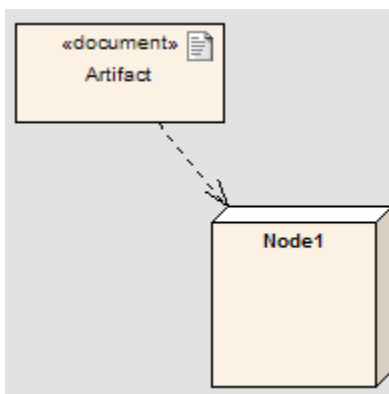
5. Select the exact *Feature* you want from the provided list.
6. Press *OK*.

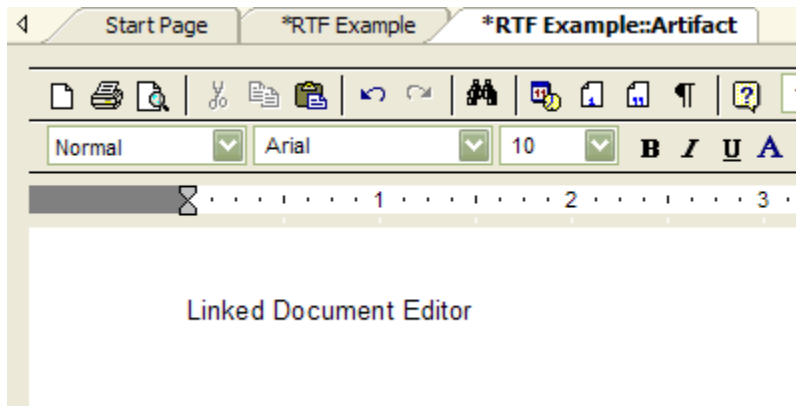
The note will now automatically derive its contents from the target element.

6.2.7 Linked Documents

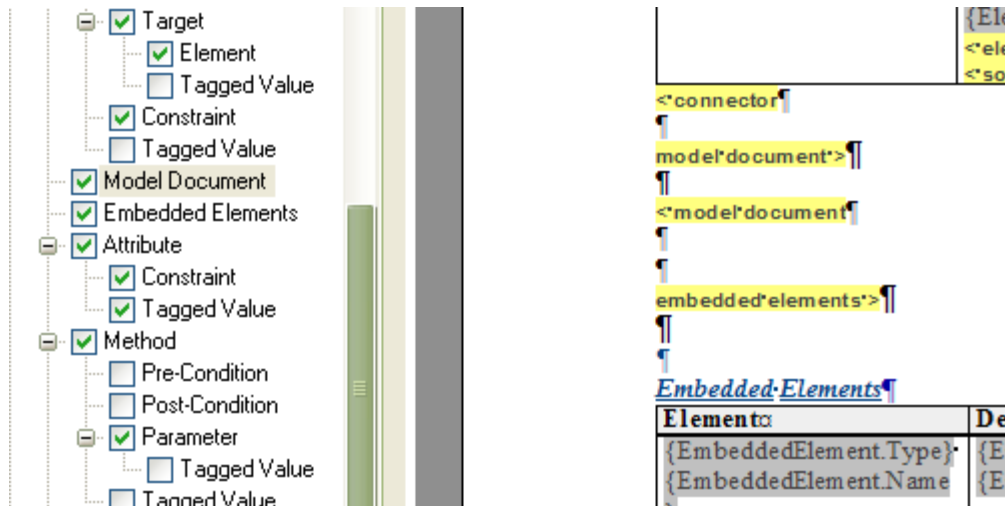
Enterprise Architect provides the ability to create documents via the new UML Artifact - "[Document Artifact](#)". The Corporate Edition of EA has the ability to link any UML Element to a document. Documents can be created from user defined *Linked Document Templates*. *Linked Document Templates* can be created with the Document Template Editor, see [Creating Linked Document Templates](#).

The new "[Document Artifact](#)" and the *Document Editor*.





Documents created via the *Document Artifact* will be rendered into the RTF Documentation by enabling the Model Document check box in the RTF Template Editor. See [New RTF Style Template Editor](#)



The document for the *Document Artifact* will be rendered into resultant RTF documentation at -

model document >
<model document

| | |
|----------------------------|-----------------|
| <u>Model Specification</u> | <u>Phase 01</u> |
|----------------------------|-----------------|

Connections

| Connector | Source | Target | Notes |
|----------------------------------|-----------------|--------------|-------|
| <u>Dependency</u> Link to Node 1 | Public Artifact | Public Node1 | |
| Source -> Destination | | | |

Linked Document Editor

See Also

- [Creating Document Artifacts](#)
- [Linking Document to UML Elements](#)
- [Editing Linked Documents](#)
- [Creating Linked Document Templates](#)
- [Editing Linked Document Templates](#)
- [RTF Style templates](#)
- [New RTF Style Template Editor](#)
- [Create a Rich Text Document](#)
- [RTF Style templates](#)
- [RTF Dialog Options](#)
- [RTF Template Editor](#)

6.2.7.1 Creating Linked Documents

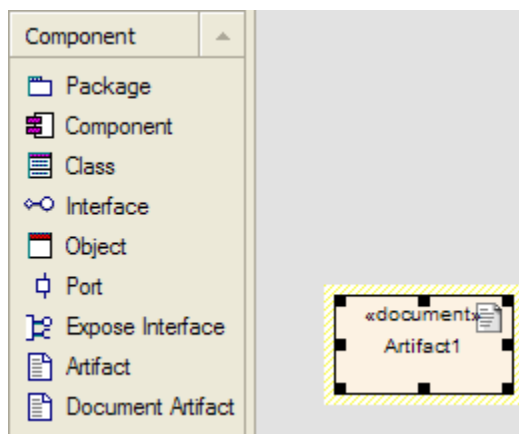
Enterprise Architect provides an additional UML Artifact - "[Document Artifact](#)", which allows the Artifact to contain an RTF document internally. The Corporate Edition of EA also allows for the linking of an internal RTF document to any UML element.

See Also

- [Creating Document Artifacts](#)
- [Linking Document to UML Elements](#)

6.2.7.1.1 Creating Document Artifacts

In a Component or Deployment diagram, drag and drop the *Document Artifact* element to your diagram.



Double-click on the *Document Artifact* element in order to access the Document Editor. The following dialog will appear. A Previously created *Linked Document Template* may be selected here.

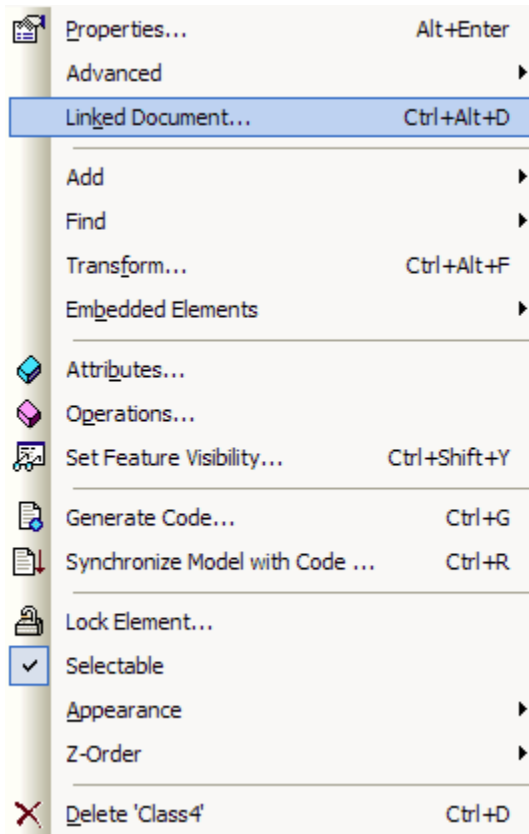
A screenshot of the Document Editor dialog box. It has a 'Name:' label followed by an empty text input field. Below it is a 'Copy template:' label followed by a dropdown menu showing 'None'. At the bottom, there are three buttons: 'Ok', 'Cancel', and 'Help'.

Press **Ok** .

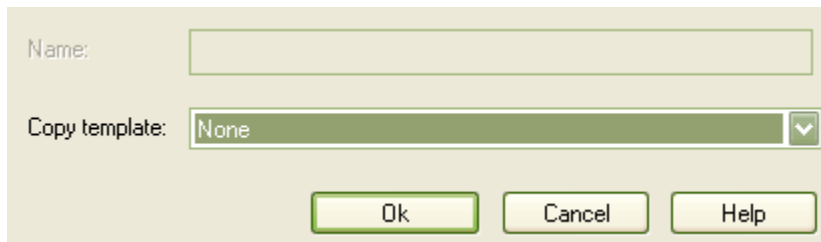
For more information on how to create *Linked Document Templates* refer to: [Creating Linked Document Templates](#)

6.2.7.1.2 Linking Document to UML Elements (Corporate Edition Only)

On any UML element, invoke the **Element | Linked Document** command from the main menu, or from the context menu.



The following dialog will appear. Previously created template may be selected.

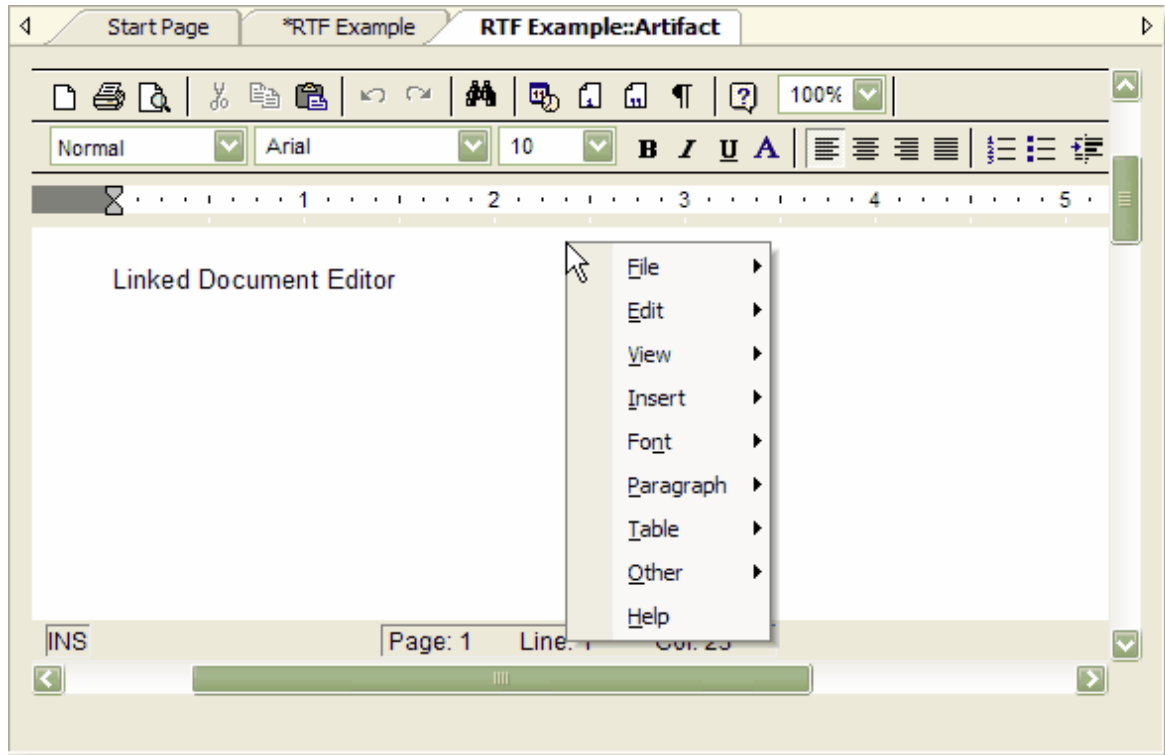


Press **Ok**.

For more information on how to create Linked Document Templates refer to: [Creating Linked Document Templates](#)

6.2.7.2 Editing Linked Documents

EA provides a windows like word processor in order to edit Linked Documents. The document word processor provides all the convenient features of a regular word processor. Documents can be saved and features can be accessed via the popup menu. To access the popup menu, just right click anywhere on the document.



The following topics provide assistance on using the Document Editor.

- [Scrolling through text](#)
- [File and Printing Options](#)
- [Line Editing](#)
- [Block Editing](#)
- [Clipboard](#)
- [Image and Object Imports](#)
- [Character Formatting](#)
- [Paragraph Formatting](#)
- [Tab Support](#)
- [Page Breaks and Repagination](#)
- [Headers, Footers and Bookmarks](#)
- [Table Commands](#)
- [Sections and Columns](#)
- [Stylesheets and Table of Contents](#)
- [Text/Picture Frame and Drawing Objects](#)
- [View Options](#)
- [Navigation Commands](#)
- [Search/Replace Commands](#)
- [Highlighting Commands](#)

6.2.7.3 *Linked Document Templates*

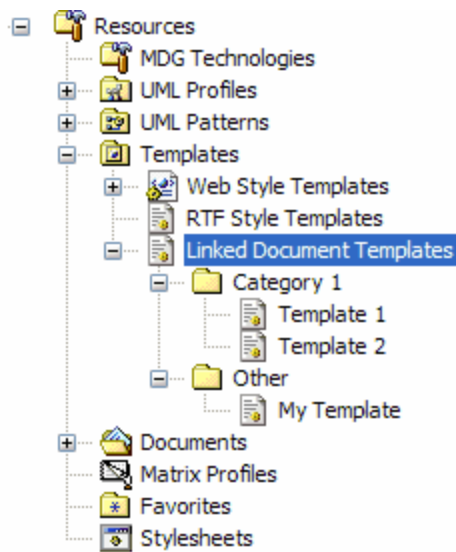
Enterprise Architect has the ability to create user defined *Linked Document Templates* that may be selected on creating a new *Linked Document*.

See Also

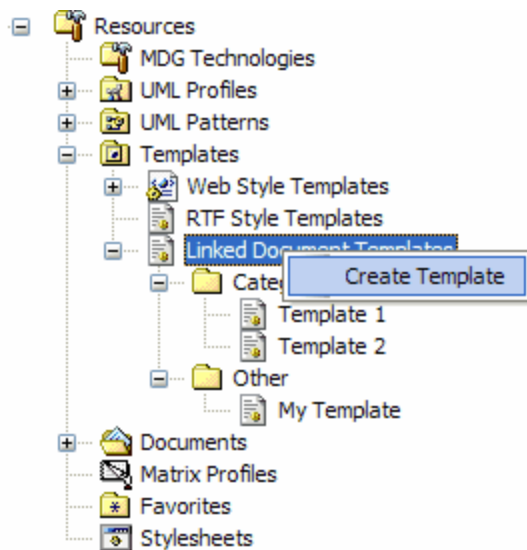
- [Creating Linked Document Templates](#)
- [Editing Linked Document Templates](#)

6.2.7.3.1 *Creating Linked Document Templates*

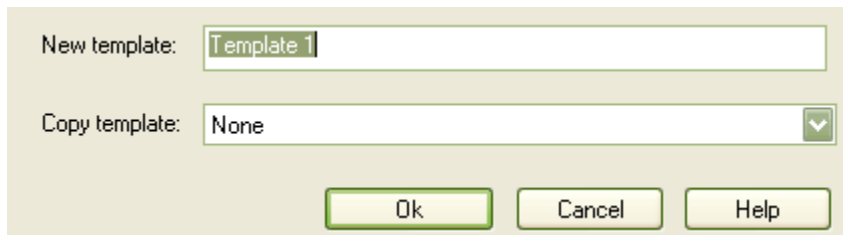
Linked Document Template can be created via the *Resources* Window.



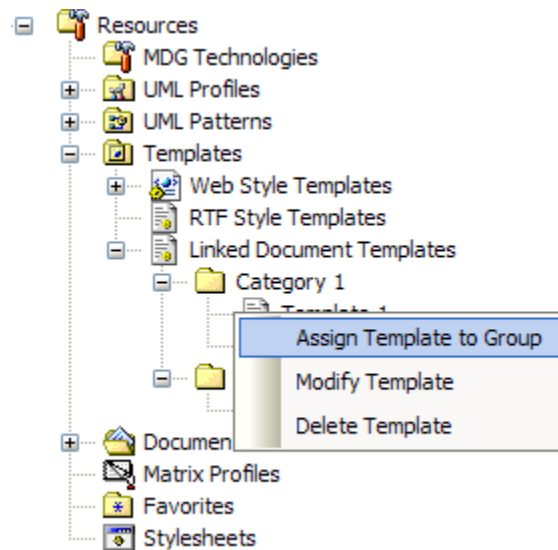
Under the *Templates* folder, right click on the *Linked Document Templates* icon.



The following dialog will appear. Enter a name for your template. Previously created template may also be selected, click **Ok**.



Templates can be grouped into folders. Right click on your newly created template and click **Assign Template to Group** to **Group**, enter a category name and press **ok**. Templates can also be modified and deleted.

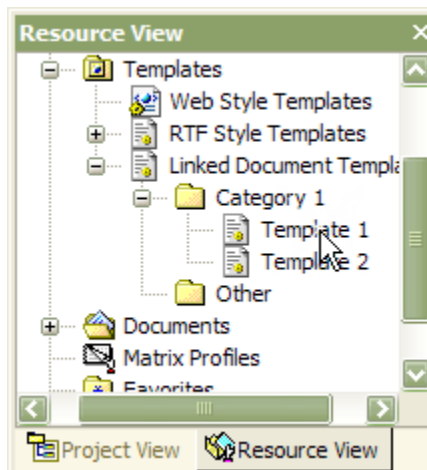


See Also

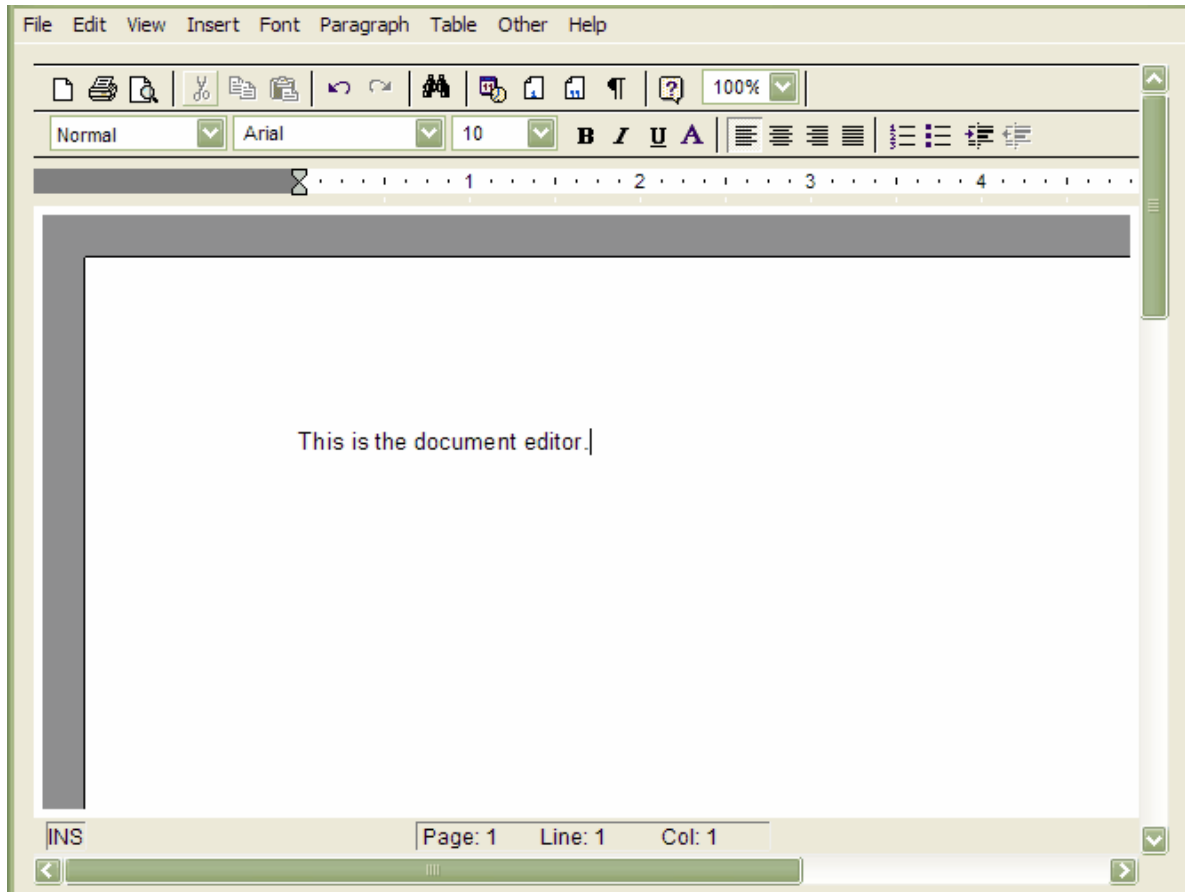
- [Editing Linked Document Template](#)

6.2.7.3.2 Editing Linked Document Templates

Double clicking on a previously created template in the *Resource View* to invoke the *Linked Document Template* Editor.



The *Document Template Editor* is built into EA.



The following topics provide assistance on using the Document Editor.

- [Scrolling through text](#)
- [File and Printing Options](#)
- [Line Editing](#)
- [Block Editing](#)
- [Clipboard](#)
- [Image and Object Imports](#)
- [Character Formatting](#)
- [Paragraph Formatting](#)
- [Tab Support](#)
- [Page Breaks and Repagination](#)
- [Headers, Footers and Bookmarks](#)
- [Table Commands](#)
- [Sections and Columns](#)
- [Stylesheets and Table of Contents](#)
- [Text/Picture Frame and Drawing Objects](#)
- [View Options](#)
- [Navigation Commands](#)
- [Search/Replace Commands](#)
- [Highlighting Commands](#)

6.3 Working with UML Connectors

UML connectors, along with elements, form the basis of a UML model. Connectors link elements together to denote some kind of logical or functional relationship between them. Each connector has its own purpose, meaning and notation and is used in specific kinds of UML diagrams.

For more information on using connectors, see:

- [Connector Context Menu](#)
- [Common Connector Tasks](#)
- [Connector Properties](#)

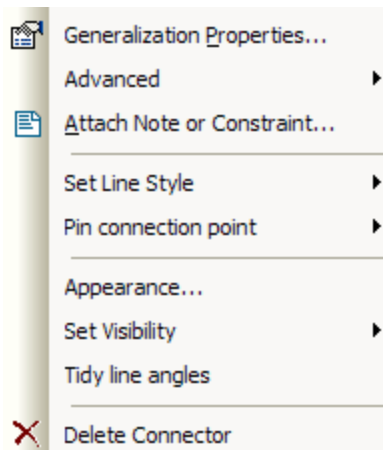
6.3.1 Connector Context Menu

If you right click on a connector in a diagram, the connector context menu opens. This provides quick access to some important functions. The menu is split into seven distinct sections:

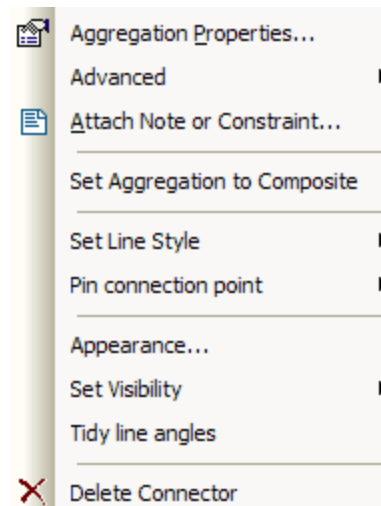
- [Zicom Mentor](#) - this will only show in the first section if you have Zicom Mentor installed and registered.
- [Properties](#)
- [Type Specific](#)
- [Common Actions](#)
- [Style](#)
- [Appearance](#)
- Delete -delete the connector with this option.

Note: Not all menu options will be present on all connector context menus. Context menus vary slightly between connector types. The type specific menu option is not always included, for example.

Example Context Menu for a Generalization:



Example Context Menu for an Aggregation:



6.3.1.1 Properties Menu Section

The Properties menu section on the Connectors context menu contains the following options:

| Menu Option | Description |
|-----------------------------|---|
| <Connector type> Properties | Opens the Properties window for the selected connector. |
| Advanced Properties | Opens the Advanced Properties dialog. |

Note: Not all menu options will be present on all connector context menus. Context menus vary slightly between connector types. The type specific menu option is not always included, for example.

6.3.1.2 Type Specific Menu Section

The Type Specific menu content is specific to the object, and only appears on a few different connectors. Some examples are below:

| Connector | Menu Option | Description |
|--------------|------------------------------|--|
| StateFlow | Message | Set the value of the message. |
| Aggregation | Set Aggregation to Composite | Change the aggregation to composite. |
| Aggregation | Set Aggregation to Shared | Appears after Set Aggregation to Composite has been selected. Sets the aggregation to shared. |
| Association | Specialize Associations | Provides a dialog that allows you to specify how the properties of this association specialize the properties of other associations. |
| Dependency | Dependency Stereotypes | Provides a sub-menu to select a stereotype for the dependency. |
| Trace | Dependency Stereotypes | Provides a sub-menu to select a stereotype for the dependency. |
| Role Binding | Dependency Stereotypes | Provides a sub-menu to select a stereotype for the dependency. |
| Represents | Dependency Stereotypes | Provides a sub-menu to select a stereotype for the dependency. |
| Occurrence | Dependency Stereotypes | Provides a sub-menu to select a stereotype for the dependency. |

Note: Not all menu options will be present on all connector context menus. Context menus vary slightly between connector types. The type specific menu option is not always included, for example.

6.3.1.3 Common Actions Menu Section

The Common Actions menu section on the Connectors context menu contains the following options:

| Menu Option | Description |
|---------------------------|--|
| Attach Note or Constraint | Attach a note or attach a constraint to the connector. |
| Connector Detail | See table below for sub-menu options. |

Element Features Sub-Menu

| Menu Option | Description |
|-----------------------|--|
| Set Source and Target | Change the source and/or target of the connector. |
| Change Type | Change the connector type . |
| Reverse Direction | Reverse the direction of the connector - eg. if the connector is an arrow, the arrowhead will swap to the other end. |

Note: Not all menu options will be present on all connector context menus. Context menus vary slightly between connector types. The type specific menu option is not always included, for example.

6.3.1.4 Style Menu Section

The Style menu section on the Connectors context menu contains the following options:

| Menu Option | Description |
|---|--|
| Set Line Style | Set the connector line style - options are Direct, Auto Routing, Custom, Bezier, Tree (Horizontal) or Tree (Vertical). |
| Pin Connection Point | Pin the start and/or connector ends to a position on the target element. |
| Bend Line at Cursor | Inserts an anchor point on the line at the point of the cursor so you can change the shape of the line. |
| Straighten Line at Cursor | Removes an anchor point on the line at the point of the cursor. |

Note: Not all menu options will be present on all connector context menus. Context menus vary slightly between connector types. The type specific menu option is not always included, for example.

6.3.1.5 Appearance Menu Section

The Appearance menu section on the Connectors context menu contains the following options:

| Menu Option | Description |
|------------------|---|
| Appearance | Set the line color and line thickness of the connector. |
| Set Visibility | See table below for sub-menu options. |
| Tidy Line Angles | Tidy the line angles of a custom connector. |

Set Visibility Sub-Menu

| Menu Option | Description |
|----------------------------------|---|
| Hide Connector | Hide the connector. To show the connector again, follow the instructions on the Hide/Show Connector page. |
| Hide Connector in Other Diagrams | Hide or show the connector in other diagrams. |
| Hide All Labels | Hide or show all labels attached to the connector. |
| Set Label Visibility | Hide or show labels attached to the connector on an individual basis. |

Note: Not all menu options will be present on all connector context menus. Context menus vary slightly between connector types. The type specific menu option is not always included, for example.

6.3.2 Common Connector Tasks

This section details some of the common tasks associated with managing model connectors.

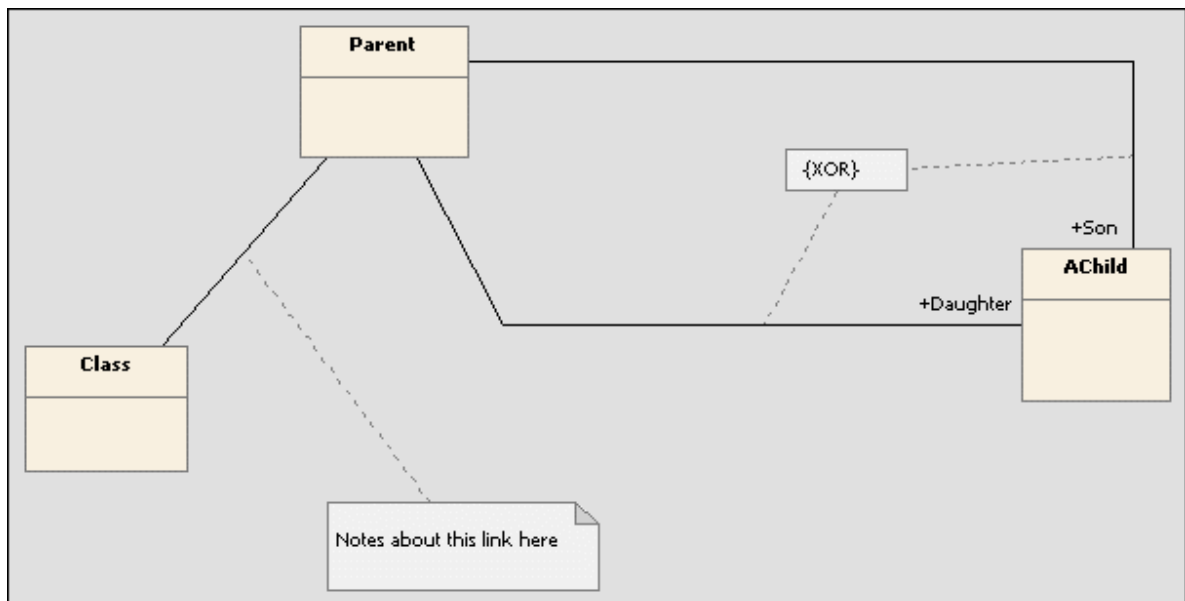
See Also

- [Connect Elements](#)
- [Change Connector Styles](#)
- [Arrange Connectors](#)

- [Change Connector Type](#)
- [Reverse Connector](#)
- [Delete Connectors](#)
- [Hide/Show Connectors](#)
- [Hide/Show Labels](#)
- [Change the Source or Target Element](#)
- [Set Relation Visibility](#)
- [Add a Note to a Link](#)
- [Use Tree Style Hierarchy](#)
- [Create a Link](#)
- [Show Uses Arrow Head](#)
- [Set Association Specializations](#)

6.3.2.1 Add a Note to a Link

You can link notes and constraints to graphical relationships. Notes let you provide explanation and further detail for one or more connectors on a diagram, with a visible note element, as in the example below.

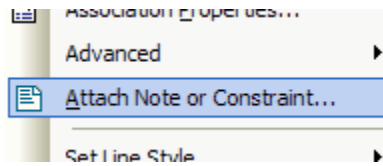


Constraints let you specify a logical or informal constraint against a set of links - for example the {XOR} constraint in the image above indicates that only one of the links in the specified set can be true at any one time (exclusivity).

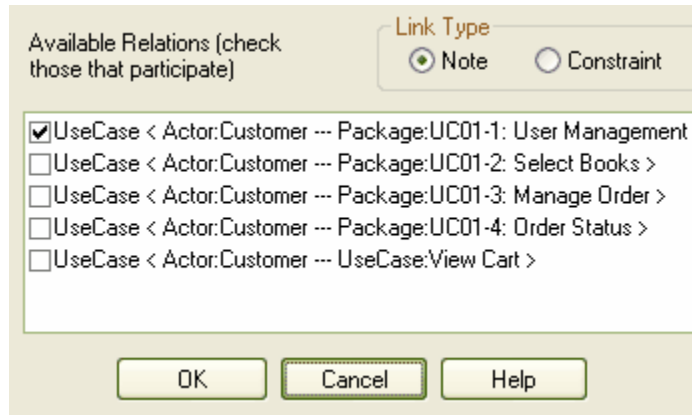
Attach a Note or Constraint to a Link

To attach a note or constraint to one or more links, do the following:

1. Right click on one of the links to attach a note to.
2. From the context menu, select *Attach Note or Constraint*.



3. In the *Link Relations* dialog, check all those links which participate in the set. In the example below, two links have been checked to participate in a logical constraint.



4. Press *OK* to complete the note or constraint creation.
5. You may then use the normal *Note dialog* to enter the appropriate text for the note or constraint.

Note: The constraint note is drawn slightly differently to a regular note, and has a { and } automatically added to visually indicate the constraint form.

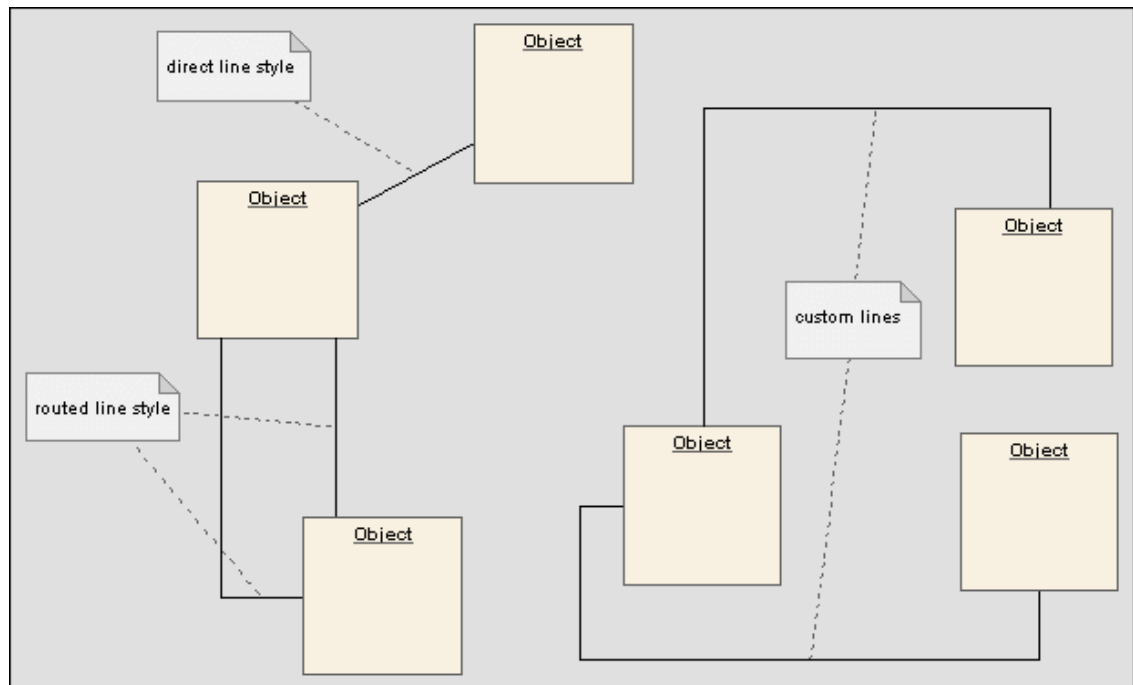
6.3.2.2 Arranging Connectors

Connectors between elements may be moved around so as to facilitate a good layout. There is a limit to how much a connector can be moved around - but generally it is very easy to find an acceptable layout. For the best layouts, use the Custom style - this allows you to add as many line points and bends as you desire to create a clean and readable diagram.

Move a Connector

To move a connector, follow the steps below:

1. Left click once on the connector to select it.
2. Holding the left mouse button down, move the mouse in the direction you wish to move the connector.
3. To refine the movement, click and hold very near to one end of the connector - this will produce a slightly different movement range.
4. To further refine the movement and range, select either a routed, direct or custom line style - each will behave slightly differently (see also: [Connector Styles](#)).



6.3.2.3 Change Connector Type

To change a connector type, follow the steps below:

1. In the diagram view, right click on the connector you wish to change, to open the context menu
2. From the *Connection Detail* submenu, select *Change Type*.
3. From the drop down list, select the *Connector Type* you require.
4. Press *OK* to apply changes.



6.3.2.4 Change the Source or Target Element

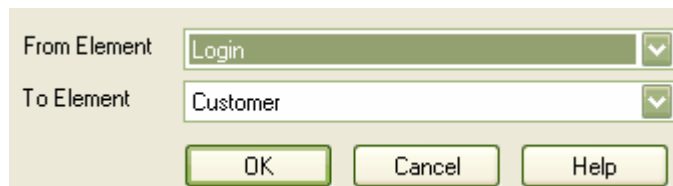
Changing the Source or Target Element for a Connector

Once you have created a link between two elements, there may come a time when you want to change either the source or target. Instead of deleting and re-creating the link, EA lets you change the source or target. There are two ways of doing this: using the mouse or using the *Set Message Scope* dialog.

Using the Set Message Scope dialog

To change the source or target element of a connector using the *Set Message Scope* dialog, follow the steps below:

1. Right click on the connector to open the context menu.
2. From the *Connection Detail* submenu, select *Set Source and Target* to open the *Set Message Scope* dialog.
3. Set the source and target elements using the *From Element* and *To Element* drop down lists.
4. Press *OK* to apply changes.



Using the Mouse

To change the source or target element of a connector using the mouse, follow the steps below:

1. Holding the *Shift* key, click with the mouse on or near the end of the connector that you wish to change.
2. Click on the element that you wish to attach to with the connector.

6.3.2.5 Connecting Elements

Connecting Elements on a diagram

The fastest and simplest ways to create connectors are via the *Quick Linker* and the *UML toolbox*. The following sections describe these and other approaches to creating connectors on a diagram:

- [Creating Connectors in place using the Quick Linker](#)
- [Creating Connectors using the UML Toolbox](#)
- [Creating a group of Elements using UML Patterns](#)
- [Creating Domain Specific Connectors from UML Profiles](#)

Note: You may reposition a connector by selecting and dragging the links as required.

Tip: To repeat the last connector you used, press the *F3* key .

Tip: To reattach one connector end to a different element, use Shift+Left-Click on a connector end, then select the desired element.

Selecting Connectors

To select a connector, simply left click it with the mouse. Drag handles will appear, indicating visually that the connector is selected. This gives the connector focus for keyboard commands such as *Delete*, and will display connector properties in docked windows such as the *Tagged Values* window. If there is more than one connector on a diagram, you can cycle between them using the arrow keys.

Dragging Connectors

You may drag a connector to position it. Left click on the connector, then while holding the mouse button down, drag the link to where you want it to appear. Note that there are some limitations on how far or to where you can drag a connector.

Connector Properties and Commands

You can double-click on a connector to [change properties](#), or right click to bring up the context menu containing commands to [change connector type](#) and [direction](#).

6.3.2.6 Connector Styles

Connectors come in five different routing styles:

Direct

The line is drawn from element A to element B in a straight line. You may move the line back and forth, up and down etc. to a limited degree.

Auto Routing

The line makes an attempt to route from A to B in a vertical and horizontal manner with 90 degree bends. Line can be pulled around a bit to give good results, but location of bends and number of bends is not configurable.

Bezier

Bezier lines allow connectors to be modeled using a smooth curved line. Bezier style is available for State Flows, State Transitions, Object Flows, and Control Flows.

Custom

Most flexible option. You can add one or more line points and bend and push the line into virtually any shape by using the *Toggle Line Point at Cursor* option..

Tree (Horizontal, Vertical)

The line is drawn from element A to B in a Tree Style line with two bends and the end points are fixed to the selected locations on the elements (Horizontal or Vertical).

Set the Connector Style

How to set the connector style, follow the steps below:

1. Right click on the connector you wish to change, to open the context menu.
2. Select *Set Line Style*.
3. From the submenu, select the style you require.

-OR-

1. Select the connector you wish to change.
2. Use the following keyboard shortcuts to change the style: *Ctrl+Shift+D* for Direct, *Ctrl+Shift+A* for Auto Routing, and *Ctrl+Shift+C* for Custom.

Bend Connectors

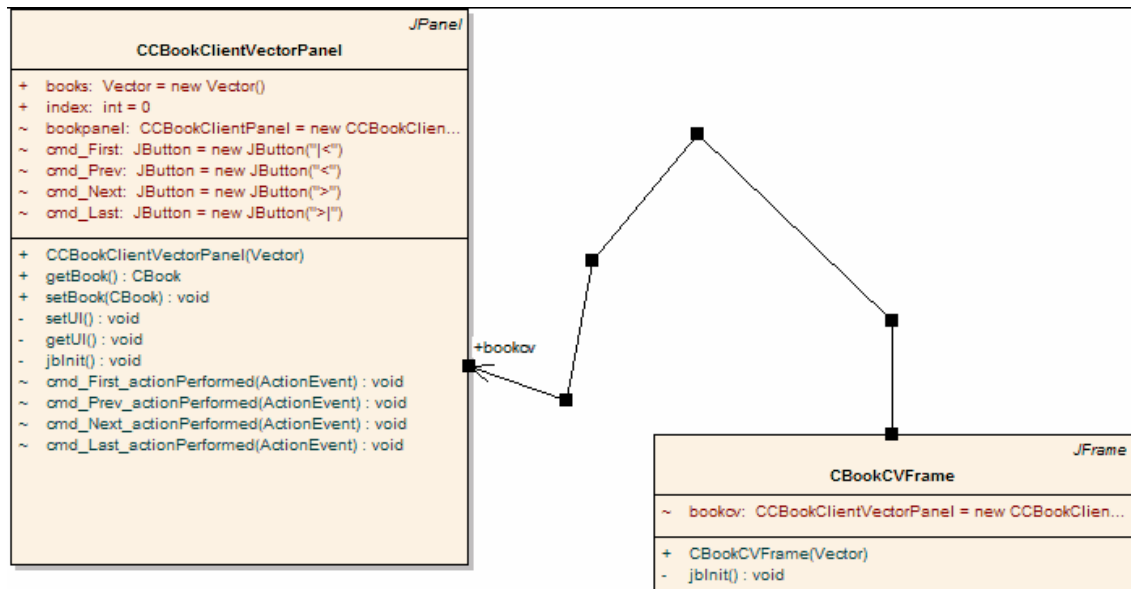
To bend a connector in order to have the ability to quickly and easily route connectors in the layout that you desire, follow the steps below:

1. Right click on the connector to open the context menu.
2. Set the line style to Custom Line (*Ctrl+Shift+C*), this will enable the *Bend Line at Cursor* command in the context menu.
3. Select the *Bend Line at Cursor* command to add a line point. (*Note: right-clicking a line point will show the command *Straighten Line at Cursor* which removes the line point*).

4. Using the mouse, drag the line point to the desired position.

-OR-

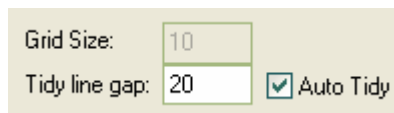
1. Hold down the **Ctrl** key and click a point on the connector to create a line point. (*Note: Ctrl+Click will also remove a line point*).
2. Using the mouse, drag the line point to the desired position.



Tidy Line Angles

To tidy line angles (custom connector), follow the steps below:

1. Right click on the connector to open the context menu
2. Select the **Tidy Line Angles** option - this will 'nudge' the custom line into good horizontal and vertical increments, saving you the time of trying to get good layouts by hand.



Tip: The tidy line function may also be set to operate by default on the **Diagram Behavior** page of the **Local Options** dialog (**Tools | Options**).

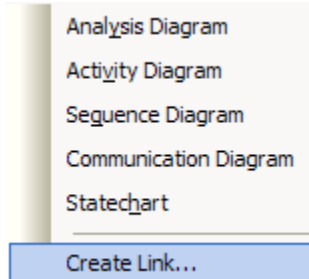
6.3.2.7 Create Link

You can create a link from one element to another directly in the Project Browser.

Link Elements from the Project Browser

To link elements from the Project Browser, follow the steps below:

1. In the Project Browser, right click on the element you wish to create a link for.
2. From the context menu, select *Add | Create Link* to open the *Create Link* dialog.



3. Select the *Direction* of the new link (Outgoing means this element is the source).
4. Select the *Link Type*.
5. Select the target in the *Choose Target* list .
6. To filter the list, use the drop down list *Select Target Type*.
7. Press *OK* to create the link.

From Element

Name: Login

Type: UseCase

Direction: Outgoing

Link Type: Aggregation

OK

Cancel

Help

To Element(s)

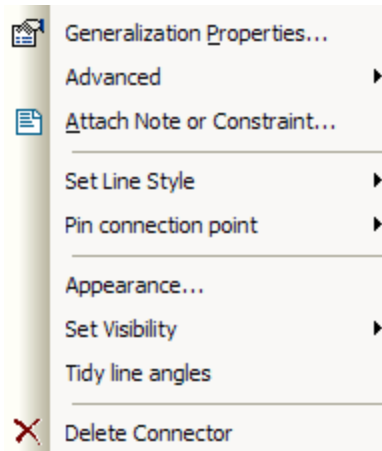
Choose target(s) Select Target Type: UseCase

| Package | Name |
|-------------------------|--------------------------------|
| UC01-1: User Management | Login |
| UC01-1: User Management | Register with Book Shop |
| UC01-2: Select Books | Browse Book Catalogue |
| UC01-2: Select Books | Locate Book by Title or Author |
| UC01-2: Select Books | Request UnListed Book |
| UC01-3: Manage Order | Add title to Cart |
| UC01-3: Manage Order | Pay for Order |
| UC01-3: Manage Order | Remove Item from Cart |
| UC01-3: Manage Order | View Cart |
| UC01-4: Order Status | Cancel Order |
| UC01-4: Order Status | Enquire on Order Status |
| XMIClassifier | Use Case |

6.3.2.8 Deleting Connectors

To delete a connector, follow the steps below:

1. Right click on the connector to open the context menu.
2. Select *Delete Connector*.



6.3.2.9 Generalization Sets

A Generalization set allows you to specify the relationship of a group of subtypes.

To create a generalization set, *Right Click* on the connector and select *Advanced | Generalization Set | New*.

Name:

Base Type:

Power Type: [v] [...]

Is Covering

Is Disjoint

Generalizations:

| Is Member | Name | Subtype/Instance |
|-------------------------------------|------|------------------|
| <input checked="" type="checkbox"/> | | Male |
| <input checked="" type="checkbox"/> | | Female |

[OK] [Cancel]

1. Give a *Name* to the Generalization set. eg. Gender
2. Select a *Power Type* from the drop down, the selector button or by typing a new one.
3. Check the Children that are part of this Generalization set in the *IsMember* column.

The OMG UML specification (*UML 2.0 Superstructure*, p. 94) states:

"Each Generalization is a binary relationship that relates a specific Classifier to a more general Classifier (i.e., from a class to its superclasses). Each GeneralizationSet defines a particular set of Generalization relationships that describe the way in which a general Classifier (or superclass) may be divided using specific subtypes."

6.3.2.10 Hide/Show Connectors

Connectors/relations which appear in multiple diagrams may be selectively shown or hidden. This allows for easier to read diagrams where elements may have many connectors, but not all are relevant in the context of the current diagram.

Hide or Show a Connector in the Current Diagram

To hide or show a connector in the current diagram, follow the steps below:

1. Select a diagram element in the diagram view.
2. Open the *Element Properties* dialog and select the *Links* tab
3. Right click on the connector you wish to hide or show. The context menu will allow you to *Show* hidden connectors or *Hide* visible ones.

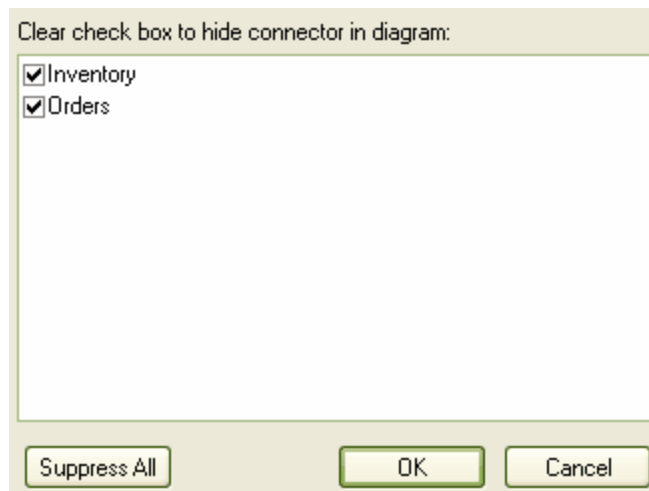
| Object | Type | Connection |
|---------------------|-------|----------------|
| CustomerAccount | Class | Generalization |
| CustomerAccount | Class | Association |
| CustomerPreferences | Class | Collaboration |
| CustomerPreferences | Class | Collaboration |
| CustomerPreferences | Class | Collaboration |
| CustomerPreferences | Class | Collaboration |
| CustomerPreferences | Class | Collaboration |
| CustomerPreferences | Class | Collaboration |

Tip: Alternatively, hide a connector by right clicking on it and selecting *Set Visibility | Hide Connector* from the context menu.

Hide or Show a Connector in Other Diagrams

To hide or show a connector in the other diagrams, follow the steps below:

1. Right click on the connector in the diagram to open the context menu.
2. From the *Set Visibility* submenu, select *Hide Connector in Other Diagrams*. This will open the *Set Connector Visibility* window.



3. If the two connected elements have been included in other diagrams, these diagrams will be listed here. All diagrams in the list which are checked will show the connector. Uncheck any diagrams in which you want the connector hidden.

Tip: If you want the connector hidden in all of the diagrams listed, press *Suppress All*.

4. When you are happy with your list, press *OK*.

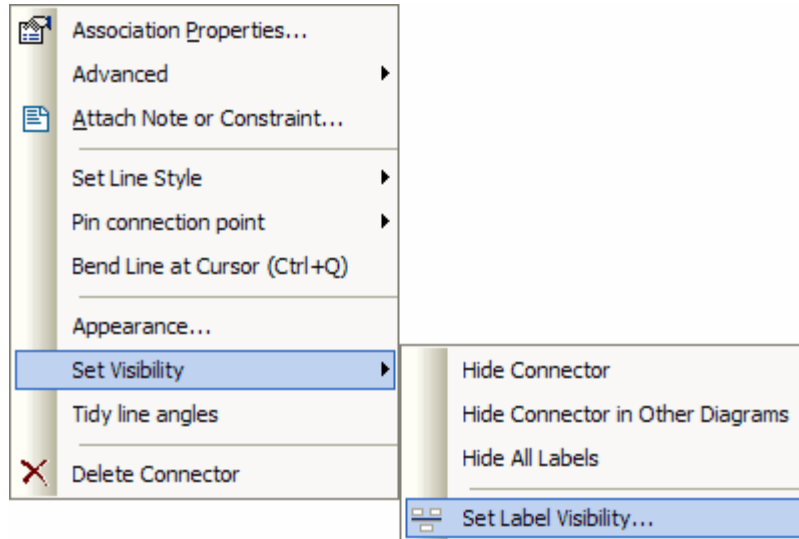
6.3.2.11 Hide/Show Labels

You can elect to hide or display one or more labels on a connector.

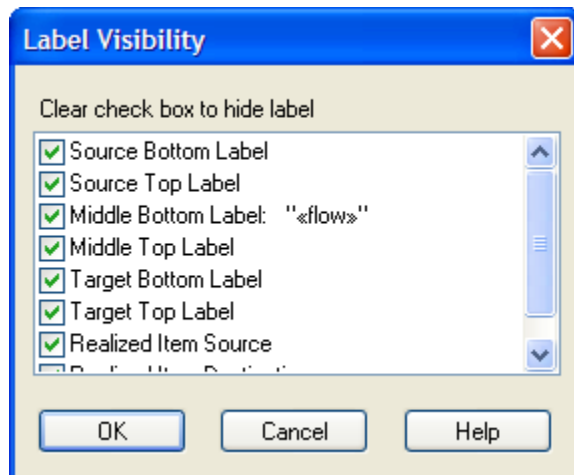
Hide/Show Labels

To hide/show labels, follow the steps below:

1. Right click on the connector to open the context menu.
2. From the *Set Visibility* submenu, select *Set Label Visibility*.



3. In the *Label Visibility* dialog that appears, check the labels to display and clear those you want to hide.
4. Press *OK* when done.



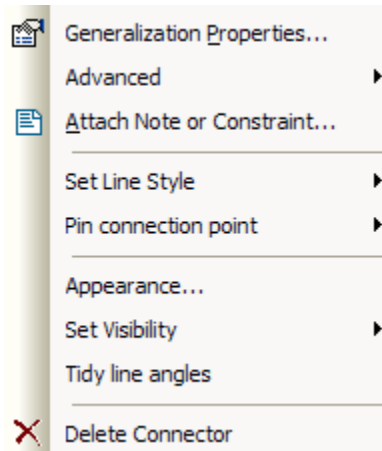
6.3.2.12 Reverse Connector

You can reverse the direction of a connector without having to delete and re-create it. This is helpful if your design changes or you add the connector wrongly to begin with.

Reverse a Connector

To reverse a connector, follow the steps below:

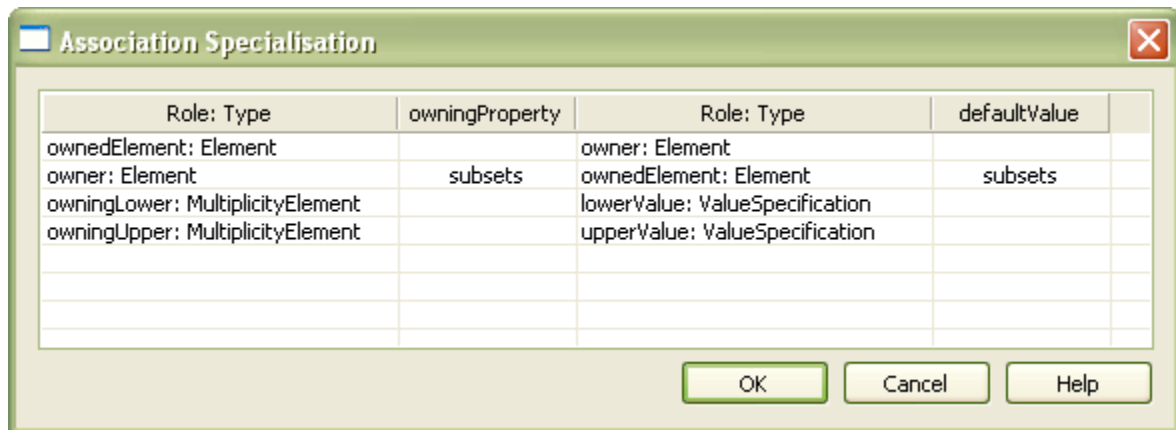
1. Right click on the incorrect connector to open the context menu.
2. From the *Connection Detail* submenu, select *Reverse Direction*.



6.3.2.13 Set Association Specializations

UML allows for the specialization of properties defined by associations. EA allows for this through the *Specialize Associations* option in the advanced section of the context menu for an association.

That brings up the following dialog that shows all associations between the two classes connected by the current association and their parents.



The left two columns refer to the source role of the current association, while the right two refer to the target role. With this you are able to select the relationships of each end to the properties listed. When a relationship is set then this will be drawn at the corresponding end of the connector on any diagram it appears on.

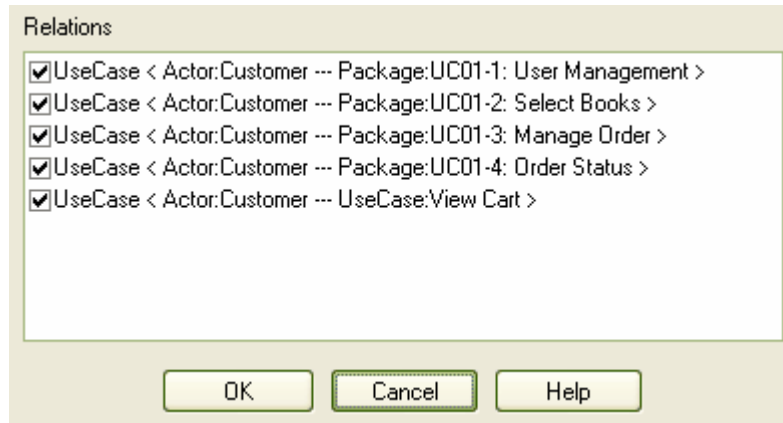
6.3.2.14 Set Relation Visibility

The visibility of individual links/connectors/relations may be changed on a per diagram basis.

Set Relation Visibility

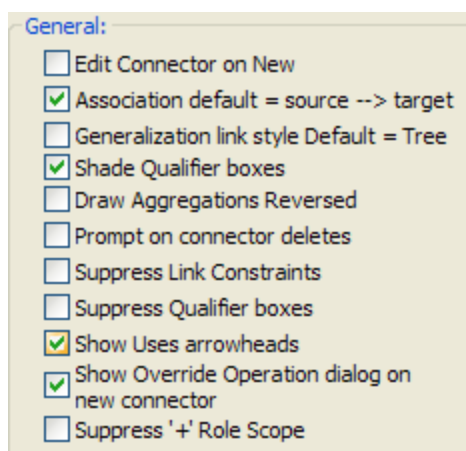
To set relation visibility, follow the steps below:

1. Open the diagram you wish to change
2. From the *Diagram* menu, select *Set Visible Relations*. Alternatively, press *Ctrl+Shift+I*.
3. Check the list items you want to show and clear those you wish to hide.
4. Press *OK* to apply changes.

**6.3.2.15 Show Uses Arrow Head**

By default EA sets the Use connector in use cases to have no arrow head, to change this behavior use the following steps.

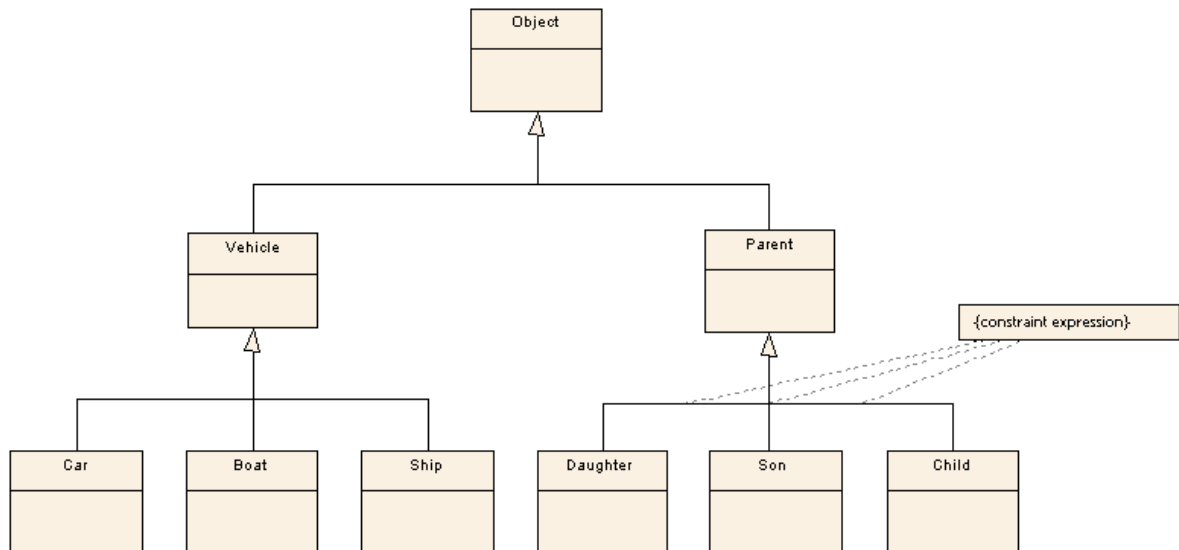
1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu .
2. Select the *Links* section.
3. Check the *Show Uses arrowheads* check box.



4. When the Use Case diagram is saved the Use connectors will change to display arrowheads.

6.3.2.16 Tree Style Hierarchy

It is possible in EA to create a tree style inheritance diagram, using a special form of the Generalization link. The example below illustrates this idea.



This style of diagram provides a clearer layout for inheritance hierarchies and is easy to work with.

Create a Tree Style Link

To create a tree style link, follow the steps below:

1. Create a normal Generalization between two elements, and then right click on the link to open the context menu.
2. Select the *Set Line Style | Tree Style - Vertical* or the *Set Line Style | Tree Style - Horizontal* option.
3. EA will automatically make the Generalization layout conform to a specific shape. By adding more Generalization links, and checking their Tree Style option, the appearance as in the diagram above is possible. You can slide the root and child classes left and right to achieve a desirable result - EA will maintain the conformity of the branch links.

Setting the Default Link Style

To set this style of link as default, follow the steps below:

1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu .
2. Select the *Links* section.
3. Check the *Generalization link style Default = Tree* check box to make this branching style the default style for inheritance links.

General:

- Edit Connector on New
- Association default = source --> target
- Generalization link style Default = Tree
- Shade Qualifier boxes
- Draw Aggregations Reversed
- Prompt on connector deletes
- Suppress Link Constraints
- Suppress Qualifier boxes
- Show Uses arrowheads
- Show Override Operation dialog on new connector
- Suppress '+' Role Scope

6.3.3 Connector Properties

To access the *Connector Properties* dialog, double click with the mouse on a connector in a diagram. Several characteristics of connectors may be changed from this dialog.

The Connector properties dialog has several tabs. The *General* tab allows you to configure the name of the connector (Link Name), the direction, the line style (routed or auto), the optional stereotype and a comment.

The screenshot shows the 'Connector Properties' dialog box with the 'General' tab selected. The dialog has four tabs: 'General', 'Constraints', 'Source Role', and 'Target Role'. The 'General' tab contains the following fields and controls:

- Source:** A text box containing 'Login'.
- Target:** A text box containing 'Register with Book Shop'.
- Link Name:** An empty text box.
- Direction:** A dropdown menu set to 'Source -> Destination'.
- Style:** A dropdown menu set to 'Custom'.
- Stereotype:** A dropdown menu.
- Object Flow:** An unchecked checkbox.
- Notes:** A large text area with a vertical scrollbar, currently empty.
- Buttons:** 'OK', 'Cancel', and 'Help' buttons are located at the bottom of the dialog.

| Control | Description |
|---------------------|---|
| Link Name | An optional name for the link. The name will be displayed on the diagram if you enter one. |
| Direction | Direction details - from source to destination, reverse or bi-directional. Some links have arrow heads which depend on this setting. Some links are logically dependent on this (eg. inheritance) |
| Style | Connection style - Choose from Direct, Auto-Routing, Bezier, Custom, Tree (Vertical) or Tree(Horizontal) |
| Stereotype | An optional stereotype for the link(will be displayed on the diagram if entered) |
| Object Flow | Available only for flow connectors. Indicates that objects or data can flow along this link. |
| Virtual Inheritance | Available only for Generalizations connectors. Indicates if inheritance is virtual. |
| Note | An optional note about the link. The note will be displayed in documentation if desired |

Tip: If you set a stereotype, this will display in a diagram and over-ride the link type in the RTF documentation.

6.3.3.1 Connector Constraints

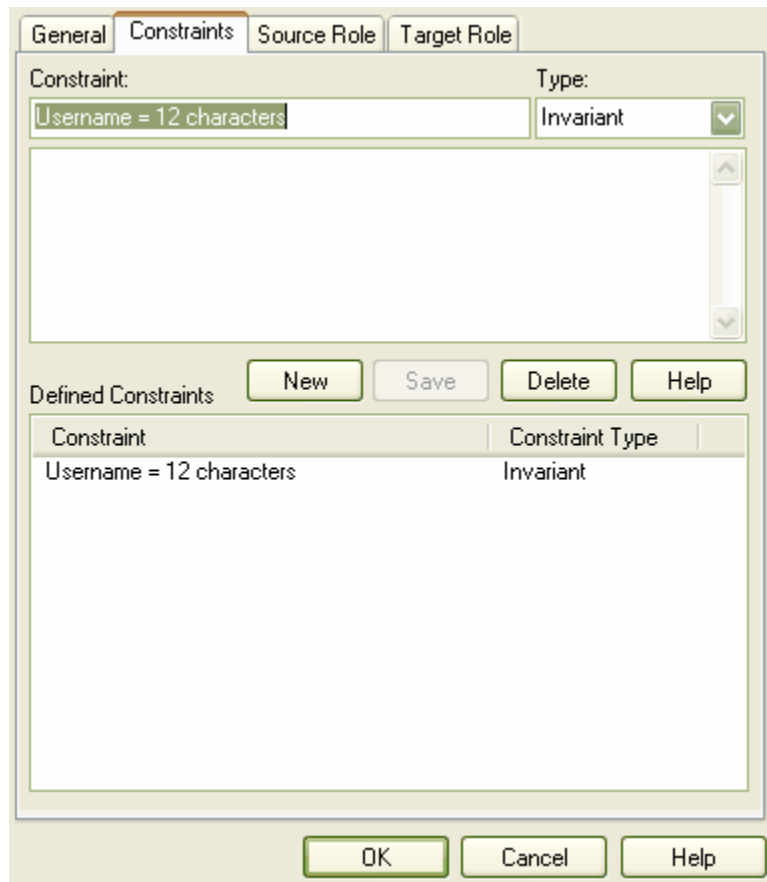
A UML connector may also have associated constraints placed on it. Constraints tell us something about the rules and conditions under which a relation operates. For example, it may be a pre-condition that the a customer is of a certain type before an association link to an Account is allowed.

Tip: Constraints about an association (connector) may be added to further refine the model. Constraints detail the business and operational rules for the model.

Set Constraints on a Link

To set constraints on a link, follow the steps below:

1. Double click on a connector to open the **Connection Properties** dialog.
2. Select the **Constraints** tab.
3. Fill in details about the constraint(s) that apply and press **Save**.



| Control | Description |
|-----------------|--|
| Constraint | Name of constraint |
| Type | The type of constraint (eg. pre-condition) |
| Notes | Notes about the link |
| Constraint List | A list of constraints for this list |
| New/Save/Delete | Add, Modify and Delete buttons |

6.3.3.2 Source Role

A link may have certain properties assigned to one end, and associated with a particular role that element will play in the relationship.

You may enter details about this role to further develop your model.

Set Source Role Details

To set the source role details, follow the steps below:

1. Double click on a connector to open the *Connection Properties* dialog.
2. Select the *Source Role* tab.
3. Enter the required details and press *OK*.

General Constraints **Source Role** Target Role

Login Role:
m_PersonList

Role Notes:
A list of staff in order

Derived Derived Union Owned
 Containment: Unspecified
 Access: Protected
 Aggregation: none
 Target Scope: instance
 Navigability: Unspecified
 Changeable: none

Multiplicity:
1..*

Ordered
 Allow Duplicates

Constraint(s):
Qualifier(s):
Stereotype: ...
Member Type: PersonList

OK Cancel Help

| Control | Description |
|------------------|--|
| Role | The name of the role to be played |
| Role Note | A note about the role |
| Derived | Check if the role value or values can be computed from other information |
| Owned | Check if the role is owned by the opposite class as opposed to the association |
| Derived Union | Check if the role is derived from the properties that subset it |
| Multiplicity | The role multiplicity (eg. 1..n) |
| Ordered | Check if the role is a list and the list is ordered |
| Allow Duplicates | Check if the role can contain duplicate elements (Relevant only if multiplicity > 1) |
| Containment | The nature of the containment at the Destination (reference, value...) |
| Access | Access level for role |
| Aggregation | The type of aggregation that this role uses |
| Target Scope | At what level does this role apply (instance or classifier) |
| Navigability | Specifies if this role is navigable (non-navigable ends are shown depending on diagram properties) |
| Changeable | Specifies how this role is subject to change |
| Role Constraints | A Free text constraint on the role |
| Qualifier | A qualifier or restriction on the role. Separate multiple qualifiers with a semi-colon. |
| Stereotype | A stereotype that applies to this end of the association |

| | |
|-------------|---|
| Member Type | A type that can be used when generating collection classes for multiplicity > 1 |
|-------------|---|

Note: Source role details are displayed at the start end of a connector. If you have drawn the link the wrong way, you can always use the Reverse Direction feature from the connector context menu.

6.3.3.3 Destination Role

A link may have certain properties assigned to one end, and associated with a particular role that element will play in the relationship.

You may enter details about this role to further develop your model.

Set Destination Role Details

To set the destination role details, follow the steps below:

1. Double click on a connector to open the *Connection Properties* dialog.
2. Select the *Destination Role* tab.
3. Details and appearance of this tab are identical to the *Source Role* tab. See [Source Role](#)

Note: Destination Role details will be displayed at the terminating end of a connector on the diagram.

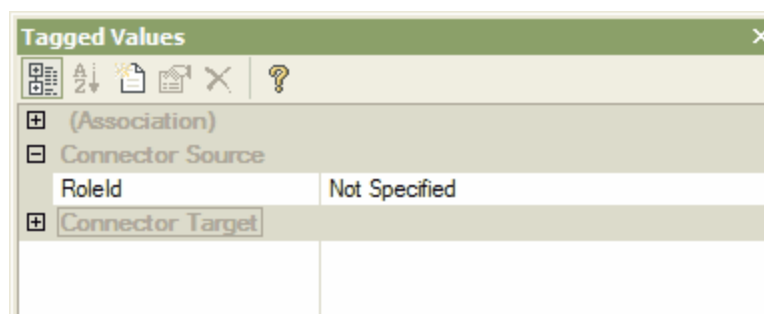
6.3.3.4 Role Tagged Values

For Association and Aggregation connector types you can set additional tagged values on the Source and/or Target Role.

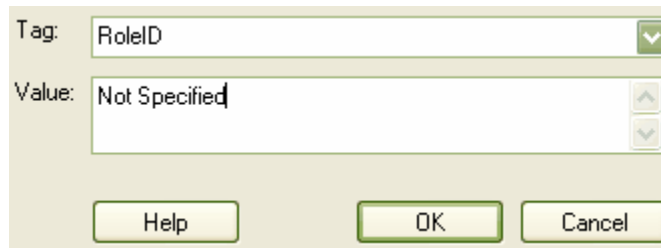
Setting Tagged Values

To set tagged values for the connector, follow the steps below:

1. Ensure that the Tagged Values window is open by pressing the *Ctrl + Shift + 6* hotkey combination, or from the *View | Other Windows | Tagged Values*.
2. Click on the connector in the diagram to display the tagged values information for the connector in the *Tagged Values* window.
3. Select *Connector Source* or *Connector Target* in the *Tagged Values* window as required.



4. In the *Tagged Values* window press either the *New Tags* button or the *Ctrl + N* hotkey combination. In the *Tagged Value* dialog enter the tag name and values or select a [Predefined Tagged Value](#) type from the dropdown list.

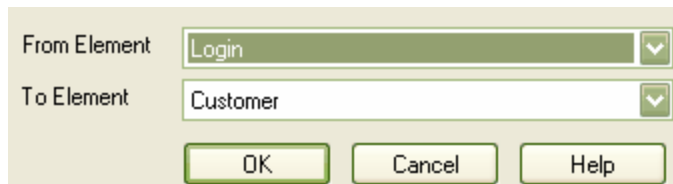


The image shows a dialog box with a light beige background. At the top, there is a label 'Tag:' followed by a text input field containing 'RoleID' and a small downward-pointing arrow on the right. Below this is a label 'Value:' followed by a larger text input field containing 'Not Specified' and two small vertical arrows on the right. At the bottom of the dialog, there are three buttons: 'Help', 'OK', and 'Cancel', each with a thin green border.

5. Press the *OK* button to save changes.

6.3.3.5 Message Scope

A message in a sequence diagram represents a dynamic interaction from one element to another. Sometimes when you are designing your model you may need to change either the start or end point of a message as the responsibilities of elements change during design. For this reason, EA lets you change the message scope by setting a new start or end element.



The image shows a dialog box with a light beige background. It has two labels: 'From Element' and 'To Element'. The 'From Element' label is followed by a dropdown menu showing 'Login'. The 'To Element' label is followed by a dropdown menu showing 'Customer'. At the bottom, there are three buttons: 'OK', 'Cancel', and 'Help', each with a thin green border.

Change Message Scope

To change message scope, follow the steps below:

1. Select the message in the sequence diagram.
2. Right click on the message to open the context menu.
3. Select *Set Message Scope*.
4. In the pop up dialog, select the required *From* and *To* elements from the drop down lists.
5. Press *OK* to save changes.

The message has now been re-routed to meet your changed requirements.

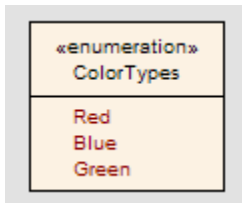
6.4 UML Stereotypes

UML supports a large number of stereotypes, which are an inbuilt mechanism for logically extending or altering the meaning, display and syntax of a model element. Different model elements have different stereotypes associated with them.

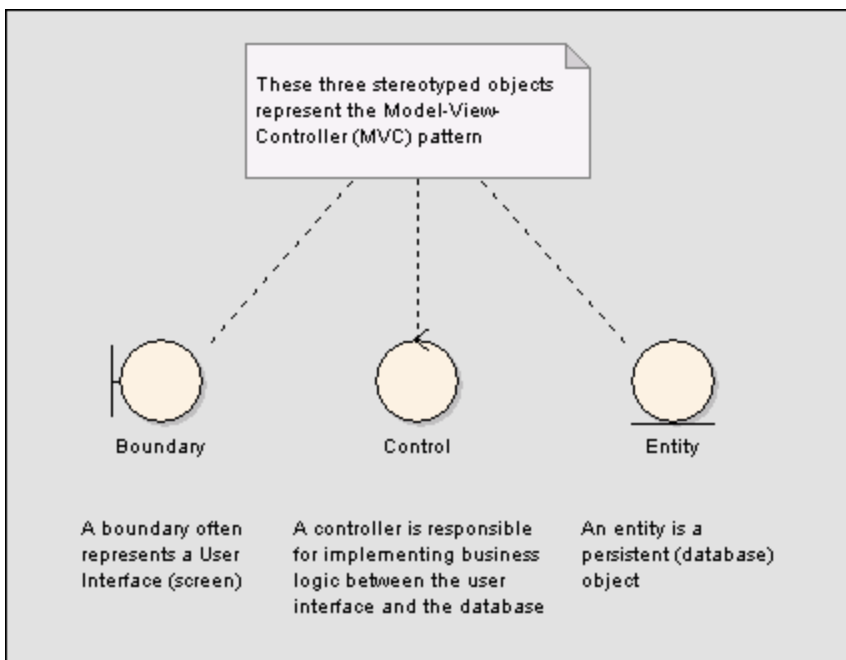
The OMG UML specification states:

"A stereotype is, in effect, a new class of metamodel element that is introduced at modeling time. It represents a subclass of an existing metamodel element with the same form (attributes and relationships) but with a different intent. Generally a stereotype represents a usage distinction. A stereotyped element may have additional constraints on it from the base metamodel class. It may also have required tagged values that add information needed by elements with the stereotype. It is expected that code generators and other tools will treat stereotyped elements specially. Stereotypes represent one of the built-in extensibility mechanisms of UML."

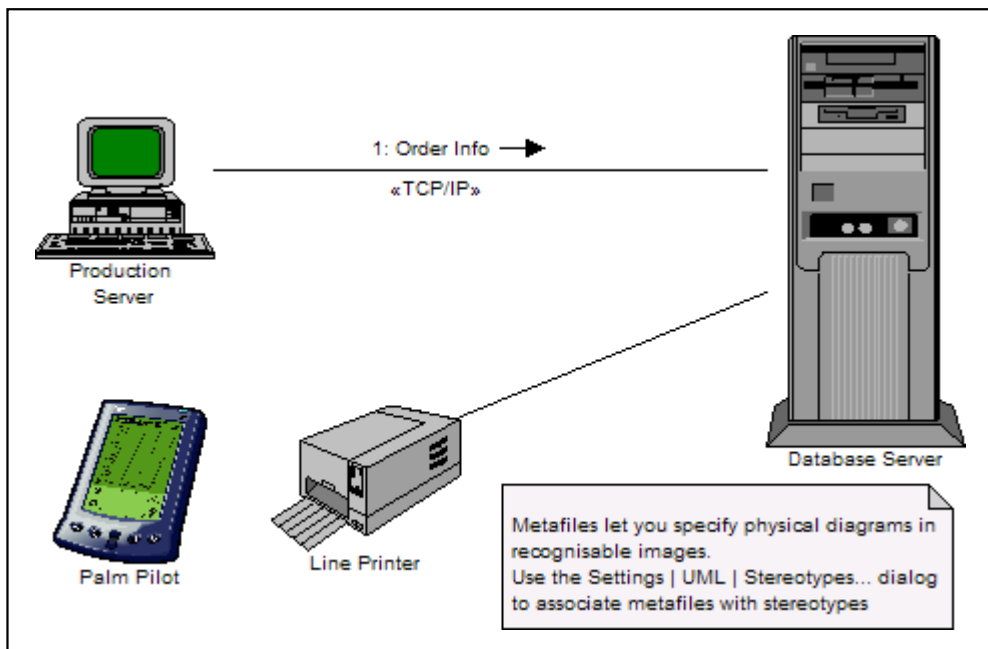
The stereotype is generally displayed as in the example below (where <<enumeration>> is the stereotype).



In some cases, the stereotype will cause the element to be drawn differently - as below:



A metafile may be associated with the applied stereotype, as in the example below:

**See Also:**

- [Custom Stereotypes](#)
- [Standard Stereotypes](#)
- [Stereotypes with Alternate Images](#)
- [Shape Scripts](#)



6.4.1 Applying Stereotypes

EA enables you to apply one or more stereotypes to any UML construct, including:

- Elements (such as Classes and Objects)
- Relationships (such as Dependencies and Associations)
- Association Ends
- Attributes, and Operations
- Operation Parameters

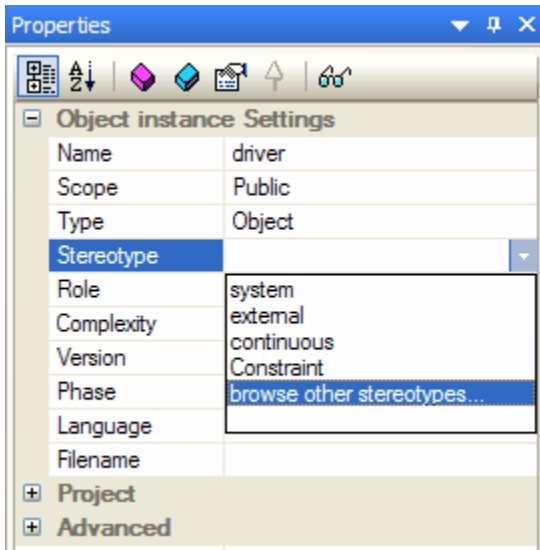
To apply a stereotype to any UML construct, using the *Properties Dialog*, select any one of the following steps:

1. Select the desired stereotype from the combo box.
2. Click on the "..." button to use the [Stereotype Selector](#) Dialog.
3. Enter the desired stereotype(s) to apply by typing them into the field as a comma-separated list.

Stereotype:  

To apply a stereotype to an Element using the *Properties Window*, select any of the following steps:

1. Select the desired stereotype from the combo box.
2. Select the "*browse other stereotypes...*" entry in the combo box to use the [Stereotype Selector](#) Dialog.
3. Enter the desired stereotype(s) to apply by typing them into the field as a comma-separated list.



See Also

- [Custom Stereotypes](#)
- [Stereotype Selector](#)
- [Stereotype Visibility](#)

6.4.2 Custom Stereotypes

With EA you can create your own stereotypes with its own custom appearance. The stereotype can be altered to make use of metafiles (image files), customized colors or make use of the EA Shape Script to make new element shapes to determine the shape and dimensions of the element.

To add your own custom stereotypes, follow the steps below:

1. From the main menu, select **Settings | UML | Stereotypes**.
2. Enter a **Stereotype** name.
3. Select a **Base Class** from the drop down list.
4. If you wish to associate a **Metafile** with this stereotype, press the **Browse [...]** button and locate the required .emf or .wmf file.
5. Enter optional **Notes** and select **Default Colors** for this stereotype
6. Press **Save** to save the stereotype

The table below describes the functionality of the Stereotypes dialog.

| Control | Description |
|----------------------------|---|
| Stereotype | Name of the Stereotype |
| Group name | Allows for the grouping of stereotype features by a plural name, for attributes and operations and is shown on diagrams in the attribute and operations compartments. |
| Base Class | Allows the stereotyped element to inherit the base characteristics from a pre-existing element type. |
| Notes | Use this section to add any stereotype notes. |
| Override Appearance | |
| None | Use this option to use the default element appearance. |
| Metafile | Metafile allows an image file to be used for the appearance of the stereotype. |

| | |
|-----------------------|--|
| Shape Script | The EA Shape Script allows the user to specify custom shapes for the stereotype using the EA Shape Scripting language. For more information see the Shape Scripting Topic. |
| Assign | Adds the associated metafile or shape script from the stereotyped element. |
| Remove | Removes the associated metafile or shape script from the stereotyped element. |
| Default Colors | |
| Fill | Fill determine the default background color of the element. |
| Border | Border is used to control the border color. |
| Font | Font is used to control the color of the stereotype font. |
| Reset | Reset the appearance of the element to the default element appearance. |

| Stereotype | Applies To | Notes |
|-----------------|----------------|---|
| access | dependency | Public contents of target are access... |
| analysis system | model | Contains analysis classes - entities, ... |
| asp page | class | A Microsoft active server page |
| become | message | Target is same as source but later in... |
| bind | dependency | Source instantiates target template ... |
| boundary | class | Specifies an element that is at the s... |
| button | GUIElement | A button GUI element |
| call | dependency | Source invokes the target |
| case worker | business class | A worker who directly interacts with ... |
| cdrom | node | A class that represents a CDROM dri... |
| cd-rom | node | A class that represents a CDROM dri... |
| check | OpTable | A Check constraint to enforce doma... |
| check | Operation | check |
| checkbox | GUIElement | A checkbox GUI element |
| client page | class | A class that represents a client base... |
| clientscript | class | A collection of client-side scripts |
| column | Attribute | column |
| column | AttribTable | A column attribute for a table |
| combobox | GUIElement | A combo box GUI element |
| communicate | uses | Communication between actor and ... |
| computer | node | A class that represents a computer |
| control | class | Specifies an element that controls th... |
| copy | message | Target is exact but independent cop... |
| create | message | Target is created by event or messa... |
| date | GUIElement | A GUI element for date entry |
| derive | dependency | Source may be computed from target |
| design system | model | Contains design elements |
| destroy | message | Target is destroyed by event or mes... |
| dialog | GUIElement | A GUI screen |
| disk array | node | A class that represents a disk array |
| document | component | The component represents a docum... |
| dropdown | GUIElement | A GUI element that forces user sele... |
| entity | class | Specifies a persistent element mainl... |
| entity | business class | Passive class accessed and manipu... |
| enumeration | class | Specifies an enumerated type |

See Also

- [Applying Stereotypes](#)
- [Stereotype Selector](#)
- [Stereotype Visibility](#)

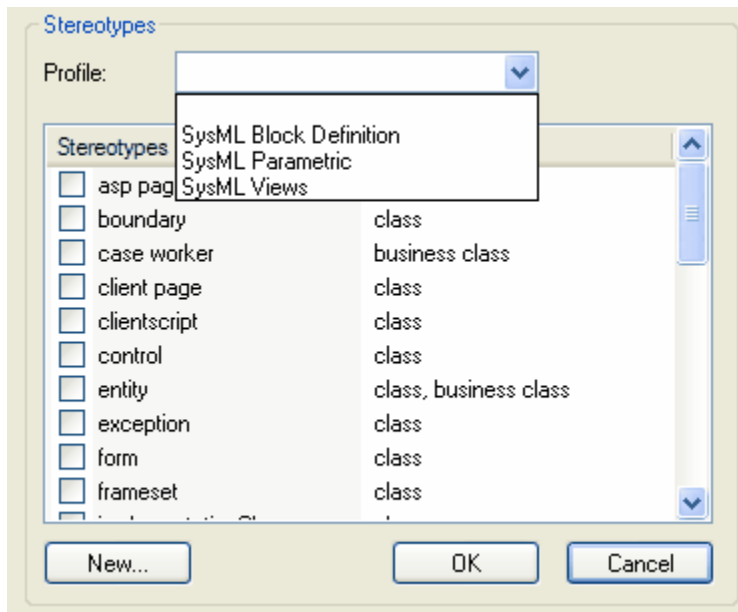
6.4.3 Stereotype Selector

The stereotype selector dialog enables you to apply one or more stereotypes to a UML construct, from multiple stereotype sources, such as profiles or the custom stereotypes list.

To Select Stereotypes to Apply/Remove

1. Invoke the Stereotype Selector dialog by clicking on the "..." button near the stereotype field.

2. Choose the desired stereotype source from the *Profile* combo.
3. Enable or Disable the desired stereotype by turning its corresponding checkbox on or off in the *Stereotypes* list.
3. Click *OK* to apply the selection.



It is also possible to define a new stereotype to apply to the desired construct by clicking on the *New...* button and entering the name of the new stereotype when prompted.

See Also

- [Applying Stereotypes](#)
- [Custom Stereotypes](#)
- [Stereotype Visibility](#)

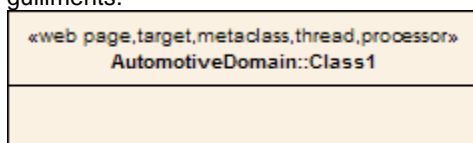
6.4.4 Stereotype Visibility

Visibility of applied stereotypes is controlled using three options in the [Diagram Properties](#) dialog:

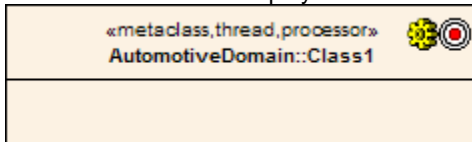
1. The *Show Element Stereotypes* checkbox to show/hide all element stereotypes in the current diagram
2. The *Show Feature Stereotypes* checkbox to show/hide all attribute/operation stereotypes in the current diagram
3. The *Use Stereotype Icons* checkbox to display icons, instead of strings, for those stereotypes that have icons defined.

The example below shows how a class would appear having multiple stereotypes applied to it:

Use Stereotype Icons disabled - displays all the applied stereotypes in a comma-separated string within gulliments.



Use Stereotype Icons enabled - displays icons for those stereotypes with icons defined. Stereotypes without icons defined are still displayed in the comma-separated string



See Also

- [Diagram Properties](#)
- [Applying Stereotypes](#)
- [Custom Stereotypes](#)
- [Stereotype Selector](#)

6.4.5 Standard Stereotypes

Below is a list of standard element stereotypes:

| Stereotype | Base Class | Type |
|------------------|-------------------|------------|
| «access» | Permission | Stereotype |
| «appliedProfile» | Package | Stereotype |
| association | Association | Constraint |
| «association» | AssociationEnd | Stereotype |
| «auxiliary» | Class | Stereotype |
| «become» | Flow | Stereotype |
| «call» | Usage | Stereotype |
| complete | Generalization | Constraint |
| «copy» | Flow | Stereotype |
| «create» | BehavioralFeature | Stereotype |
| «create» | CallEvent | Stereotype |
| «create» | Usage | Stereotype |
| «derive» | Abstraction | Stereotype |
| derived | ModelElement | Tag |
| «destroy» | BehavioralFeature | Stereotype |
| «destroy» | CallEvent | Stereotype |
| destroyed | Association | Constraint |
| destroyed | Association | Constraint |
| disjoint | Generalization | Constraint |
| «document» | Abstraction | Stereotype |
| documentation | Element | Tag |
| «executable» | Abstraction | Stereotype |
| «facade» | Package | Stereotype |
| «file» | Abstraction | Stereotype |
| «focus» | Class | Stereotype |
| «framework» | Package | Stereotype |
| «friend» | Permission | Stereotype |
| global | Association | Constraint |
| «global» | AssociationEnd | Stereotype |
| «implementation» | Class | Stereotype |
| «implementation» | Generalization | Stereotype |
| implicit | Association | Stereotype |
| «import» | Permission | Stereotype |

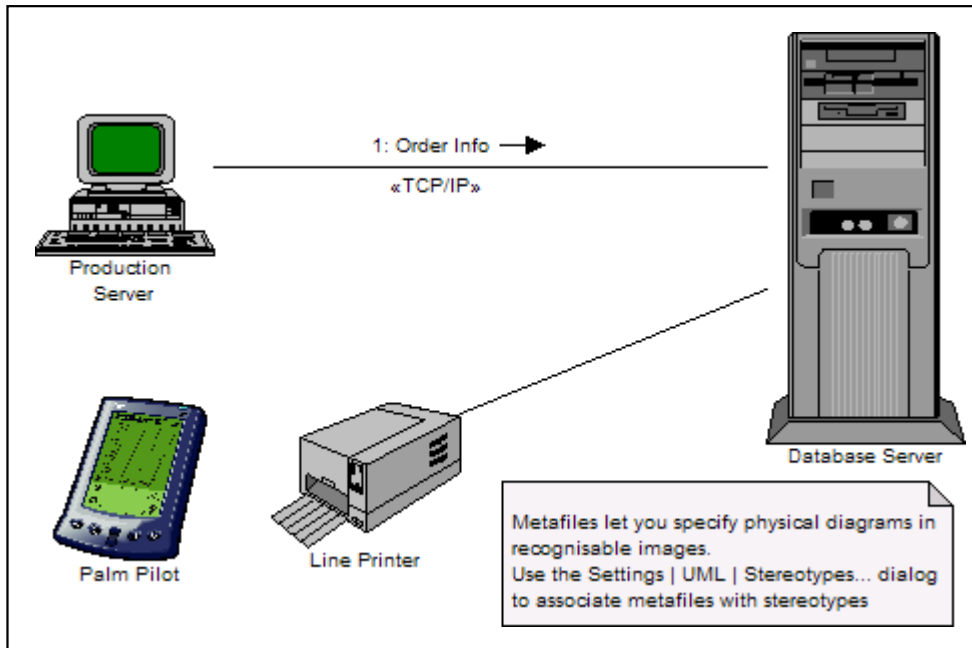
| | | |
|---------------|----------------|------------|
| incomplete | Generalization | Constraint |
| «instantiate» | Usage | Stereotype |
| «invariant» | Constraint | Stereotype |

| | | |
|------------------|-----------------|------------|
| «library» | Abstraction | Stereotype |
| local | Association | Constraint |
| «local» | AssociationEnd | Stereotype |
| «metaclass» | Class | Stereotype |
| «metamodel» | Package | Stereotype |
| «modelLibrary» | Package | Stereotype |
| «modelLibrary» | Package | Stereotype |
| new | Association | Constraint |
| new | Association | Constraint |
| overlapping | Generalization | Constraint |
| parameter | Association | Constraint |
| «parameter» | AssociationEnd | Stereotype |
| persistence | Association | Tag |
| persistence | Attribute | Tag |
| persistence | Classifier | Tag |
| persistent | Association | Tag |
| «postcondition» | Constraint | Stereotype |
| «powertype» | Class | Stereotype |
| «precondition» | Constraint | Stereotype |
| «process» | Classifier | Stereotype |
| «profile» | Package | Stereotype |
| «refine» | Abstraction | Stereotype |
| «requirement» | Comment | Stereotype |
| «responsibility» | Comment | Stereotype |
| self | Association | Constraint |
| «self» | AssociationEnd | Stereotype |
| semantics | Classifier | Tag |
| semantics | Operation | Tag |
| «send» | Usage | Stereotype |
| «signalflow» | ObjectFlowState | Stereotype |
| «source» | Abstraction | Stereotype |
| «stateInvariant» | Constraint | Stereotype |
| «stub» | Package | Stereotype |
| «systemModel» | Package | Stereotype |
| «table» | Abstraction | Stereotype |
| «thread» | Classifier | Stereotype |
| «topLevel» | Package | Stereotype |

| | | |
|-----------|-------------|------------|
| «trace» | Abstraction | Stereotype |
| transient | Association | Constraint |
| transient | Association | Constraint |
| «type» | Class | Stereotype |
| usage | Association | Tag |
| «utility» | Classifier | Stereotype |
| xor | Association | Constraint |

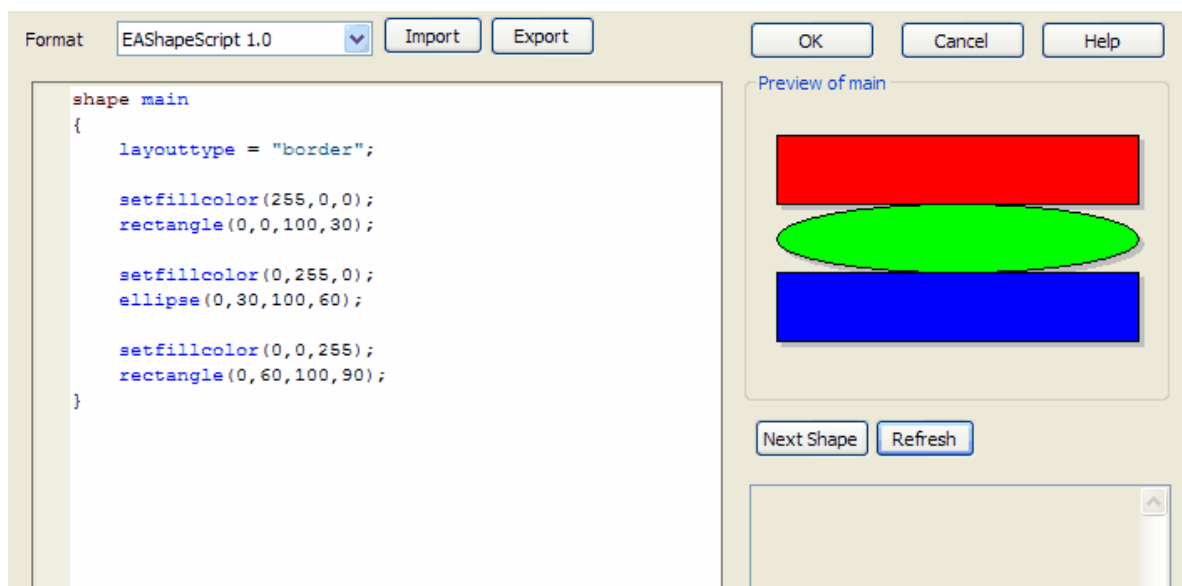
6.4.6 Stereotypes with Alternate Images

You can alter the appearance of elements using stereotypes. If the stereotype has an associated metafile specified, then when the stereotype is applied to a class or other element which supports alternate graphical format, then EA will draw the alternate image instead of the standard one.



6.4.7 Shape Scripts

EA *Shape Scripts* allows you to specify custom shapes via a scripting language. These custom shapes are drawn instead of the standard UML notation. Each script is associated with a particular *Stereotype*, and will be drawn for every element of that stereotype.

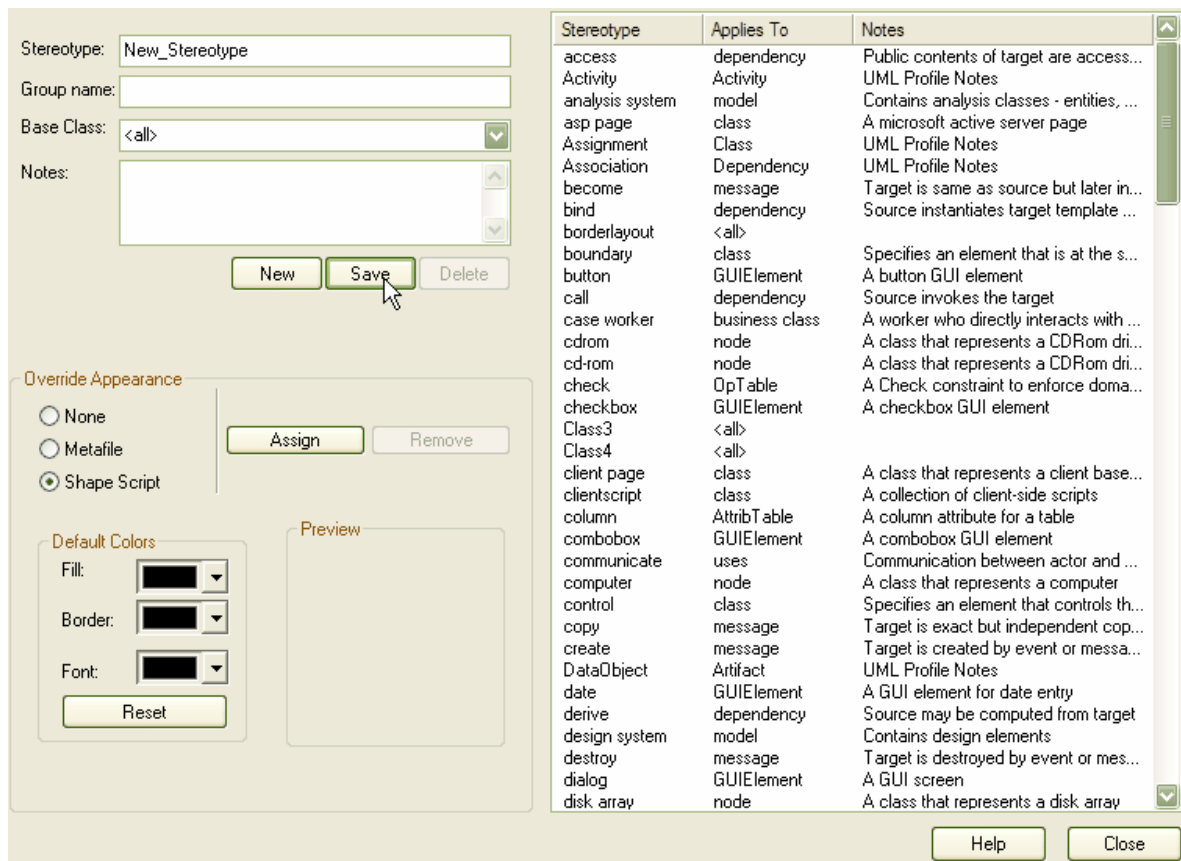


See Also

- [Getting Started](#)
- [Writing Script](#)
- [Example Scripts](#)
- [Shape Editor](#)

6.4.7.1 Getting Started

Shape scripts are associated with stereotypes. Each stereotype defined can have a *Shape Script*. *Shape scripts* are defined via the *Stereotypes* dialog. In order to access this dialog, go to the main menu and select **Settings | UML | Stereotypes**.



Creating a *Shape Script* for an existing *Stereotype* may be achieved by selecting a *Stereotype* from the list. Alternatively, new *Stereotypes* may be created by pressing the **New** button and giving the *Stereotype* a name. Select a base class and press **Save**. Once the *Stereotype* is saved, it will appear in the list.

In order to override the appearance, select *Shape Script* radio button and then press the **Assign** button. The following dialog will be displayed.



Enter the example shape scripts in the edit window, you can press the Refresh button in order to view it in the preview window. Once you have finished writing your shape scripts, press **OK**. In order for your shape script to be saved, you need to press **Save** from the **Stereotypes** dialog.

Once you have created your Shape Script for a particular stereotype, you can now assign that stereotype to an element. The appearance will now reflect the shape script you created. To do this, drag and drop a *Class* element to your diagram. Right click on the Class element and press **Properties**. From the **Stereotype** pull down, select the stereotype you created, press **OK**. The objects shape will now reflect the *shape script* you created.

The screenshot shows the 'General' tab of a dialog box for a class element named 'Class1'. The 'Stereotype' dropdown is set to 'New Stereotype'. The 'Abstract' checkbox is unchecked. The 'Status' is 'Proposed', 'Complexity' is '*Easy', and 'Language' is 'Java'. The 'Phase' and 'Version' are both set to '1.0'. There is an 'Advanced' button and standard 'Apply', 'OK', 'Cancel', and 'Help' buttons at the bottom.

See Also

- [Shape Scripts](#)
- [Writing Scripts](#)
- [Example Scripts](#)
- [Shape Editor](#)

6.4.7.2 Writing Scripts

The following section is a detailed reference for writing shape scripts. For an introduction to writing shape scripts, see the [Getting Started](#) and [Example Scripts](#) pages.

See the following reference sections for more detailed information on shape scripting:

- [Syntax Grammar](#)
- [Shape attributes](#)
- [Drawing methods](#)
- [Color queries](#)
- [Conditional branching](#)
- [Query methods](#)
- [Displaying Element properties](#)
- [Sub-shapes](#)
- [Reserved Names](#)
- [Miscellaneous](#)

See Also

- [Shape Scripts](#)

- [Getting Started](#)
- [Example Scripts](#)
- [Shape Editor](#)

6.4.7.2.1 Syntax Grammar

grammar symbols:

* = zero or more

+ = one or more

| = or

; = terminator

| | | |
|-----------------------------------|-----|---|
| ShapeScript | ::= | <Shape>*; |
| Shape | ::= | <ShapeDeclaration> <ShapeBody>; |
| ShapeDeclaration | ::= | <ShapeType> <ShapeName>; |
| ShapeType | ::= | "shape" "decoration"; |
| ShapeName | ::= | <ReservedShapeName> <stringliteral>; |
| ReservedShapeName | ::= | See Reserved Names for full reserved shape listing |
| ShapeBody | ::= | "{" <InitialisationAttributeAssignment>* <DrawingStatement>* <SubShape>* "}"; |
| InitialisationAttributeAssignment | ::= | <Attribute> "=" <Value> ";"; |
| Attribute | ::= | See Shape Attributes for full listing of attribute names |
| DrawingStatement | ::= | <IfElseSection> <Method>; |
| IfElseSection | ::= | "if" "(" <QueryExpression> ")" <TrueSection> [<ElseSection>]; |
| QueryExpression | ::= | <QueryName> "(" <ParameterList> ")"; |
| QueryName | ::= | See Query Methods for a full listing of Query names |
| TrueSection | ::= | "{" <DrawingStatement>* "}" |
| ElseSection | ::= | "else" "{" <DrawingStatement>* "}" |
| Method | ::= | <MethodName> "(" <ParameterList> ")" ";"; |
| MethodName | ::= | See Drawing Methods for a full listing of method names |

6.4.7.2.2 Shape Attributes

syntax: attribute "=" value ";"

example:

```
shape main
{
    //Initialisation attributes - must be before drawing commands
    noshadow = "true";
    h_align = "center";

    //drawing commands
    rectangle(0,0,100,100);
    println("foo bar");
}
```

| Attribute Name | Type | Description |
|----------------|--------|---|
| v_align | string | Affects vertical placement of printed text and subshapes depending on the layoutType attribute. Valid values: "top", "center", or "bottom"; |
| h_align | string | Affects horizontal placement of printed text and subshapes depending on the layoutType attribute. Valid values: "left", "center", or "right" |

| | | |
|-------------------------|---------|---|
| endpointy, endpointx | integer | Only used for the reserved "target" and "source" shapes for connectors, this point determines where main connector line connects to the end shapes. Default: 0 and 0 |
| editableField | string | Adding this attribute to a shape will define it as an editable region of the element. This field only impacts element shapes only, line glyphs are not supported. Valid Values: "alias", "name", "note", "stereotype" |
| noshadow | string | Assign to "true" to suppress the shapes shadow from being rendered. Default: "false" Valid values: "true" or "false" |
| orientation | string | Applies to "decoration" shapes only. Determines where the decoration is positioned within the containing element glyph. Valid values: "NW", "N", "NE", "E", "SE", "S", "SW", "W" |
| layoutType | string | Determines subshapes are sized and positioned. See layout types for further details. Valid values: "leftright", "topdown", "border" |
| preferredWidth | | used by border layout - east and west used by leftright layout, shapes where scalable is false to determine how much space they occupy for layout purposes |
| preferredHeight | | used by border layout - north and south used in the drawing of the source and target shapes for connectors to determine how wide the line is. |
| scalable | string | Setting scalable to false will stop the shape from being relatively sized to the associated diagram glyph, Valid values: "true" or "false" Default: "true" |
| rotatable | string | Set to "false" to prevent rotation of the shape. This attribute is only applicable to the "source" and "target" shapes for lines glyphs. Default: "true" Valid values: "true" or "false" |

6.4.7.2.3 Drawing Methods

| Method Name | Description |
|--|--|
| moveto (int x, int y) | moves the pen cursor to the point specified by x and y. |
| lineto (int x, int y) | moves the pen cursor to the point specified by x and y. |
| rectangle (int left, int top, int right, int bottom) | draws a rectangle with extents at left top, right, bottom. |

| | |
|--|--|
| roundrect (int left, int top, int right, int bottom, int abs_cornerwidth, int abs_cornerheight) | Draws a rectangle with rounded corners with extents defined by left, top, right and bottom. The size for the corners is defined by abs_cornerwidth and abs_cornerheight, these values do not scale with the shape. |
| ellipse (int left, int top, int right, int bottom) | Draws an ellipse with extents defined by left, top, right and bottom. |
| arc (int left, int top, int right, int bottom, int startingpointx, int startingpointy, int endingpointx, int endingpointy) | draws an ellipical arc with the ellipse having extents at left, top, right and bottom. The start point of the arc is defined by the intersection of the ellipse and the line from the center of the ellipse and the point (startingpointx, startingpointy). The end of the arc is similarly defined by (endingpointx, endingpointy). |
| bezierto (int controlpoint1x, int controlpoint1y, int controlpoint2x, int controlpoint2y, int endpointx, int endpointy) | Draws a bezier curve and updates the pen position. |
| polygon (int centerx, int centery, int numberofsides, int radius, float rotation) | Draws a regular polygon with center at the point (centerx, centery), numberofsides number of sides |
| image (string imageld, int left, int top, int right, int bottom) | Draws the image has the name imageld in the Image Manager . |
| startpath () | |
| startcloudpath (puffWidth, puffHeight, noise) | Parameters: float puffWidth (default = 30), the distance between puffs float puffHeight (default = 15), the distance float noise (default = 1.0), the randomisation of the puffs positions. Similar to StartPath, except that it draws the path with cloud like curved segments ("puffs"). |
| endpath () | Ends the sequence of drawing commands that define a path. |

| | |
|---|---|
| fillpath() | Fills the previously defined path with the current fill color. |
| strokepath() | Draws the outline of the previously defined path with the current pen |
| fillandstrokepath() | Fills the previously defined path with the current fill color, then draws its outline with the current pen. |
| drawnativeshape() | <p>DrawNativeShape will cause EA to render the shape using it's usual, non-shapescript notation. Subsuqent drawing commands will be super-imposed over the native notation.</p> <p>This method is only enabled for element shape scripts, line shape scripts are not supported.</p> |
| setpen(int red, int green, int blue, [int penwidth]) | sets the pen to the color and optionally the pen width. |
| setlinestyle(string linestyle) | <p>Parameters: string linestyle, one of the following styles are valid: "solid", "dash", "dot", "dashdot", "dashdot", "dashdot", "dashdot", "dashdotdot", "dashdotdotdot", "dashgap", "dotgap" "dashdotgap", "dashdotdotgap", "dashdotdotdotgap"</p> <p>Changes the stroke pattern for commands that use the pen.</p> |
| setpenwidth(int penwidth) | Sets the width of the pen. Pen with should be between 1 and 5. |
| setfillcolor(int red, int green, int blue) | Sets the fill color. |
| setpencolor(int red, int green, int blue) | Sets the pen color. |
| setdefaultcolors() | Returns the brush and pen color to the default settings, or to the user defined colors if available. See Color Queries . |
| addsubshape(string shapename, [int width, int height]); | Adds a sub shape with the name shapename that must be defined with in the current shape definition. |
| hidelabel(string labelname) | |
| showlabel(string labelname) | |
| println(string text) | Appends a line of text to the shape and a line break. |
| print(string text) | |

6.4.7.2.4 Color Queries

Color queries may only be used to retrieve arguments for the SetPenColor and SetFillColor commands. These queries may be used in place of the arguments.

```

getUserFillColor()
getUserBorderColor()
getUserFontColor()
getUserPenSize()

```

```

shape main
{
    setfillcolor(getuserbordercolor());
    setpencolor(getuserfillcolor());

    rectangle(0,0,100,100);
}

```

6.4.7.2.5 Conditional Branching

Shape scripts provides condition branching with the if else statement, and query methods that evaluate to either true or false.

See [Syntax Grammar](#) for if statement syntax.

See [Query Methods](#) for methods that can be used as the conditional expression for if statements.

See [Example Scripts](#) for an example.

6.4.7.2.6 Query Methods

Two query methods are available for seeing if the associated element has certain tags or properties, these methods may be used as the conditional expression for an if else statement.

| | |
|--|--|
| boolean HasTag(string tagname, [string tagvalue]) | HasTag returns true if the associated element has a tag value with the name tagname. If the second parameter tagvalue is provided, the tag tagname must be present, and the value of the tag has to be equal to tagvalue for the method to return true. |
| boolean HasProperty(string propertyname, [string propertyvalue]) | HasProperty returns true if the associated element has a property with the name propertyname. If the second parameter propertyvalue is provided, the property must be present, and the value of the property has to be equal to propertyvalue for the method to return true. |
| See Displaying Element Properties for a list of valid values for propertyname. | |

6.4.7.2.7 Displaying Element Properties

The commands print, println, and printwrapped all take a string parameter representing the text to be printed. Element properties may be added to the text using the substitution macro #propertyname#.

For example: println("name: #NAME#");

Element properties visible to shape scripts: Properties for Element Shape Scripts

name
 scope
 type
 classifier
 propertytype
 stereotype
 alias
 language
 complexity
 version
 phase
 language
 packagename
 author
 status
 keywords
 istagged
 isabstract

multiplicity
isroot
isleaf
isspec
isactive
persistence
notes
cardinality
visibility
concurrency
isembedded
islocked
parentedge
datecreated
datemodified

Properties for Connector Shape Scripts

name
direction
type
stereotype
source.multiplicity
target.multiplicity
source.aggregation
target.aggregation
source.multiplicityisordered
target.multiplicityisordered
source.changable
target.changable
source.constraints
target.constraints
source.qualifiers
target.qualifiers
source.targetscope
target.targetscope
source.stereotype
target.stereotype
source.metatype
target.metatype
isroot
isleaf
notes
subtype

6.4.7.2.8 Sub-Shapes

Shapes can contain and be composed of other shapes.

Subshape Layout:

To set the layout type, the layouttype attribute must be set in the initialisation attributes section of the script, in other words, before any of the methods are called.

Valid values for this attribute are:

"LeftRight"

Shapes with leftright layout will position subshapes side by side, with the first added on the left, and subsequent subshape to the right.

"TopDown"

TopDown places subshapes in a vertical arrangement, with the first shape added to the top and subsequent shape added below.

"Border"

Border layout requires an additional argument to the addsubshape method to specify which region of the containing shape the subshape will occupy, "N", "E", "S", "W" or "CENTER". Each region may only be occupied by one subshape.

A subshape that assigned to the "E" or "W" region, must have its **preferredwidth** attribute specified in its declaration. Similarly, subshapes added to "N" or "S" must have there **preferredheight** attribute set. In this case, the values for these attribute are treated as static lengths and do not scale glyph.

6.4.7.2.9 Reserved Names

Elements

Elements (class, state, event, etc) have 2 reserved names for parts of the shape.

| | |
|--------------|---|
| main | The main shape is the whole shape. |
| label | Define a shape for label and the shape will have a detached label. |

Connectors

Connectors (association, dependency, generalization, etc) have 3 reserved names for parts of the shape.

| | |
|---------------|---|
| main | The main shape is the whole shape. |
| source | The source shape is an extra shape at the source end of the connector. |
| target | The target shape is an extra shape at the source end of the connector. |

6.4.7.2.10 Miscellaneous

Return command:

Execution of the script may be terminated by using the return command. Please see [Example Scripts](#) for an example.

Looping:

The Shape Script feature does not support looping constructs.

Comments:

C-style comments are supported. Examples:

```
// C Style Single Line comment
/* Multi Line
comment supported */
```

String Manipulation:

Not Supported.

Arithmetical Operations:

Not Supported.

Variables Declaration:

Not Supported.

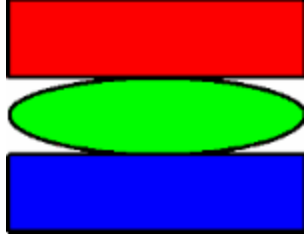
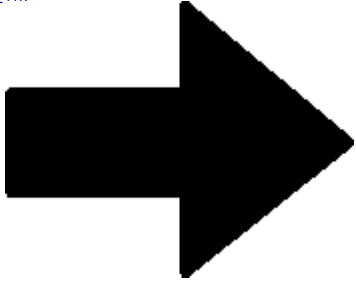
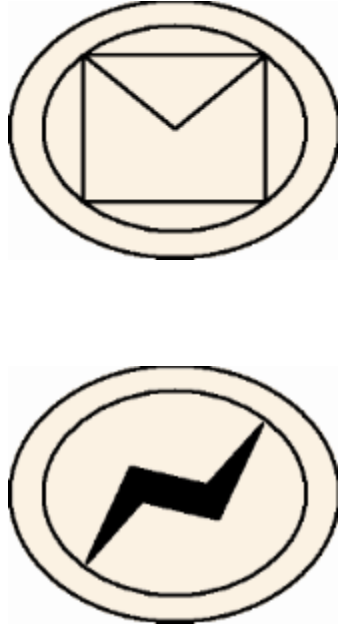
Can I apply a Shapascript without using Stereotypes?

No.

6.4.7.3 Example Scripts

Below is a selection of example scripts.

| Code | Result |
|------|--------|
|------|--------|

| | |
|--|---|
| <pre>//BASIC SHAPES shape main { setfillcolor(255,0,0); // (R,G,B) rectangle(0,0,90,30); // (x1,y1,x2,y2) setfillcolor(0,255,0); // (R,G,B) ellipse(0,30,90,60); // (x1,y1,x2,y2) setfillcolor(0,0,255); // (R,G,B) rectangle(0,60,90,90); // (x1,y1,x2,y2) } </pre> |  |
| <pre>//SINGLE CONDITIONAL SHAPE shape main { if (HasTag("Trigger","Link")) { // Only draw if the object has a tagged value Trigger=Link // Set the fill color for the path setfillcolor(0,0,0); startpath(); // Start to trace out a path moveto(23,40); lineto(23,60); lineto(50,60); lineto(50,76); lineto(76,50); lineto(50,23); lineto(50,40); endpath(); // End tracing out a path // Fill the traced path with the fill color fillandstrokepath(); return; } } </pre> |  |
| <pre>//MULTI CONDITIONAL SHAPE shape main { startpath(); ellipse(0,0,100,100); endpath(); fillandstrokepath(); ellipse(3,3,27,27); if (HasTag("Trigger","None")) { return; } if (HasTag("Trigger","Error")) { setfillcolor(0,0,0); startpath(); moveto(23,77); lineto(37,40); lineto(60,47); lineto(77,23); lineto(63,60); lineto(40,53); lineto(23,77); endpath(); fillandstrokepath(); return; } if (HasTag("Trigger","Message")) { rectangle(22,22,78,78); moveto(22,22); lineto(50,50); lineto(78,22); return; } } </pre> |  |

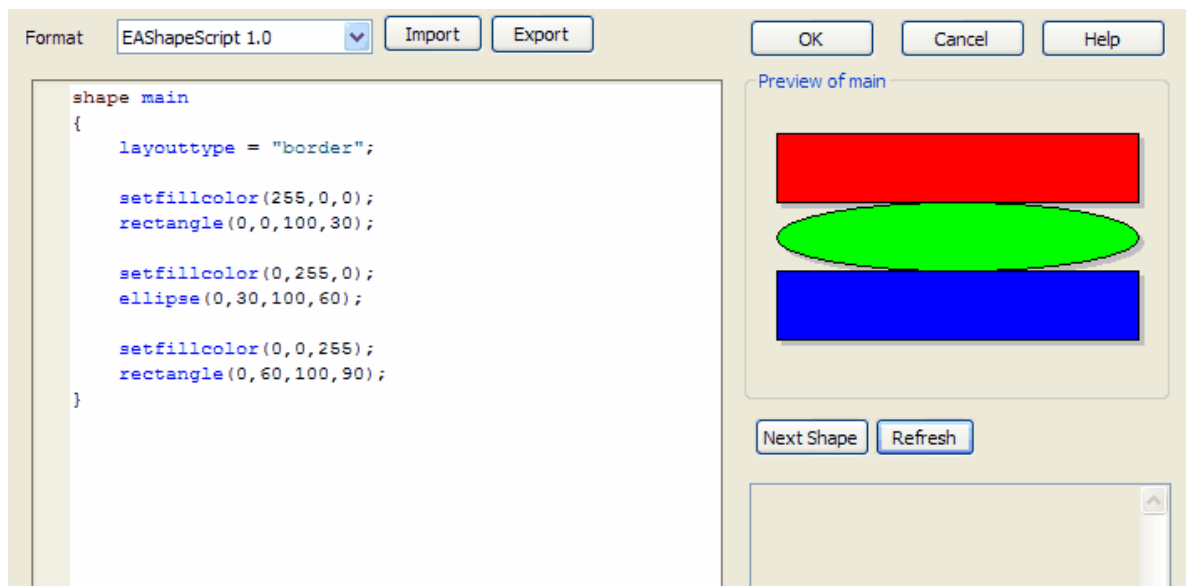
| | |
|--|--|
| <pre>//SUB SHAPES shape main { rectangle(0,0,100,100); addsubshape("red", 10, 20); addsubshape("blue", 30, 40); addsubshape("green", 50, 20); addsubshape("red", 100, 20); shape red { setfillcolor(200, 50, 100); rectangle(0,0,100,100); } shape blue { setfillcolor(100, 50, 200); rectangle(0,0,100,100); } shape green { setfillcolor(50, 200, 100); rectangle(0,0,100,100); } } </pre> |  |
| <pre>//Editable Field Shape shape main { rectangle(0,0,100,100); addsubshape("namecompartment", 100, 20); addsubshape("stereotypecompartment", 100, 40); shape namecompartment { h_align = "center"; editablefield = "name"; rectangle(0,0,100,100); println("name: #name#"); } shape stereotypecompartment { h_align = "center"; editablefield = "stereotype"; rectangle(0,0,100,100); println("stereotype: #stereotype#"); } } </pre> |  |
| <pre>//Return Statement Shape shape main { if(hasTag("alternatenotation", "false")) { //draw ea's inbuild glyph drawnativeshape(); //exit script with the return statement return; } //alternate notation commands //... rectangle(0,0,100,100); } </pre> | |

**See Also**

- [Shape Scripts](#)
- [Getting Started](#)
- [Writing Scripts](#)
- [Shape Editor](#)

6.4.7.4 Shape Editor

The *Shape Editor* allows the creation of *Shape Scripts* and is accessed via the *Stereotypes* dialog. In order to access the *Stereotypes* dialog, go to the main menu and select **Settings | UML | Stereotypes**. From the **O** *verride Appearance* group, select *Shape Script*, then press the **Assign** (or **Edit**) button.



| Control | Description |
|------------|--|
| Format | Selects the version |
| Import | Import a shape script from a text file. |
| Export | Export shape script to a text file. |
| OK | Exit out of the Shape Editor, don't forget to save your script from the Stereotypes dialog. See Getting Started |
| Cancel | Exit |
| Next Shape | Rotate though the multiple shape definitions. |
| Refresh | Parses your script and displays the result in the Preview window. |

See Also

- [Shape Scripts](#)

- [Getting Started](#)
- [Writing Scripts](#)
- [Example Scripts](#)

6.5 MDG Technologies

The Model Driven Generator (MDG) Technologies allow for a logical collection of resources pertaining to a specific technology to be bundled into one centralized location in Enterprise Architect. With MDG Technologies the user has the option of granular importation of UML Profiles, UML Patterns, Code templates and Language types to be contained in a single, easy to access area contained in the Enterprise Architect Resource View. To get you started with MDG Technologies Sparx Systems offer MDG Technologies for download from www.sparxsystems.com.au/resources/mdg_tech/.

Typical Tasks

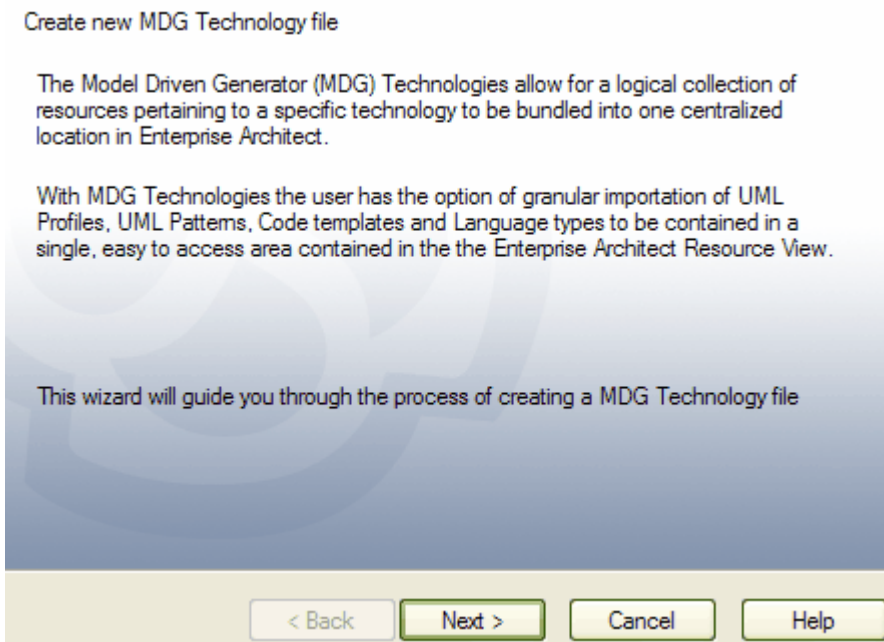
Typical tasks you might want to perform with MDG Technologies include:

- [Creating MDG Technologies](#)
- [Working with MDG Technologies](#)
- [Adding MDG Technologies to the UML Toolbox](#)
- [Import MDG Technologies](#)

6.5.1 Creating MDG Technologies

MDG Technology files may be created using the MDG Technology wizard. By using the Technology wizard it is possible to create and MDG Technology files which may include [UML Profiles](#), Code Modules, [Patterns](#), Images and [Tagged Value Types](#). To create an MDG Technology file use the following instructions:

1. To start the wizard go to the *Tools* menu and select *Generate MDG Technology File*.
2. This will open the *MDG Technology Creation Wizard*, press the *Next* button to proceed.



- The wizard will then allow the user to specify the creation of a new MDG Technology Selection (MTS) file, or use an existing MTS file or the option of not using an MTS file. An MTS file stores the selected options that a user defines during the creation of an MDG technology file and if created will allow the user to perform modifications to the MTS file, allow the user to add or remove specific items to the MDG technology file. Select the appropriate option and then press the *Next* button to continue the Wizard.

MDG Technology Wizard - Use a MTS file

Specify the name and path of the MDG Technology Selections (MTS file) to use. You can open an existing MTS file and modify your previous customizations, or you can choose to create a new MTS file.

You can also choose not to use a MTS file at all for the creation of this technology.

Create a new MTS file

Open an existing MTS file

Name and path of MTS to open:

Don't use a MTS file for this technology

< Back Next > Cancel Help

- If a MTS file is selected the user will be prompted to save the changes in the existing MTS file or given the opportunity to save the changes into a new MTS file, this allows the user to preserve the existing MTS file and create a modification based on the existing MTS file. Select the appropriate option and press the *Next* button to proceed with the Wizard.

MDG Technology Wizard - Save MTS file

You can save your changes in the MTS file that you opened, or enter the name and path of a new MTS file.

Name and path of MTS to save:

< Back Next > Cancel Help

- From the Create section of the MDG Technology Wizard the options detailed in the table below may be selected by the user, choose the appropriate options and then press the *Next* button to continue.

MDG Technology Wizard - Create
Please specify the technology to be created

Technology:

Filename:

ID: Version:

Notes:

Select items to be included in this Technology:

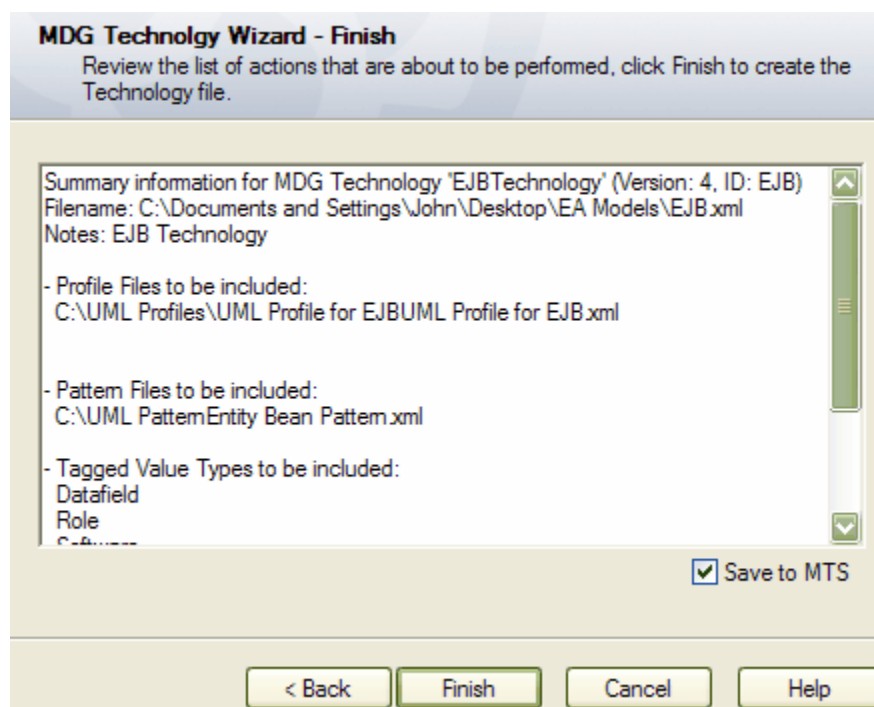
- Profiles
- Patterns
- Tagged Value Types
- Code Modules
- MDA Transforms
- Images

< Back Next > Cancel Help

| Field | Description |
|-------|-------------|
|-------|-------------|

| | |
|--|--|
| Technology | The Technology field is used to define the name of the Technology. |
| Filename | The Filename is the path to the MDG Technology file (the file extension for the MDG Technology File is .xml). |
| ID | The ID fields is a reference to the MDG Technology file, this field is restricted to a length of 12 characters. |
| Version | Specify the version of the MDG Technology file. |
| Notes | Add an explanation regarding the functionality of the MDG Technology. |
| Select Items to be included in this Technology | This section of the Wizard gives the user control over the items to be included in the MDG Technology, ensure the checkbox is selected next to the items that are to be included in the Technology file. |

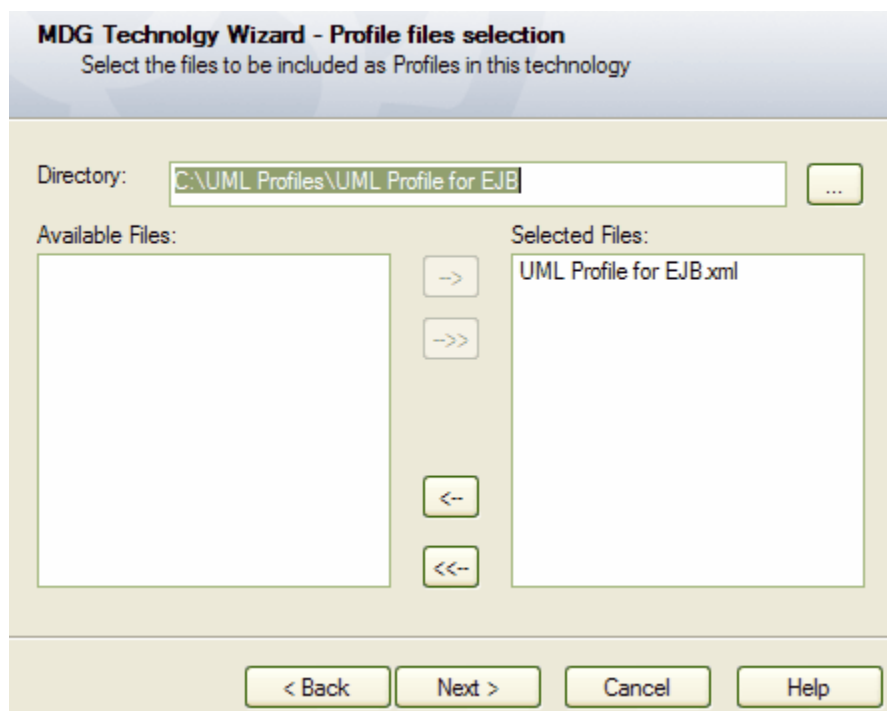
6. The items selected in the Select Items to be included in this Technology will run the specific dialogs to enable the selection of the specific items that are to be included in the MDG Technology. The method used for selection of specific items are found in the following topics:
- [Profiles](#)
 - [Patterns](#)
 - [Tagged Values Types](#)
 - [Code Modules](#)
 - [MDA Transforms](#)
 - [Images](#)
7. The final Wizard screen will then provide information about the items that are included in the MDG Technology file, if the user is satisfied with the selection of items press the *Finish* button. To update the MTS file ensure the Save to MTS checkbox is selected. To import the Technology file into an EA model see the [Import MDG Technology](#) topic.



6.5.1.1 Adding Profile in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including UML 2 compliant profiles. To use the Profiles section of the wizard use the following instructions:

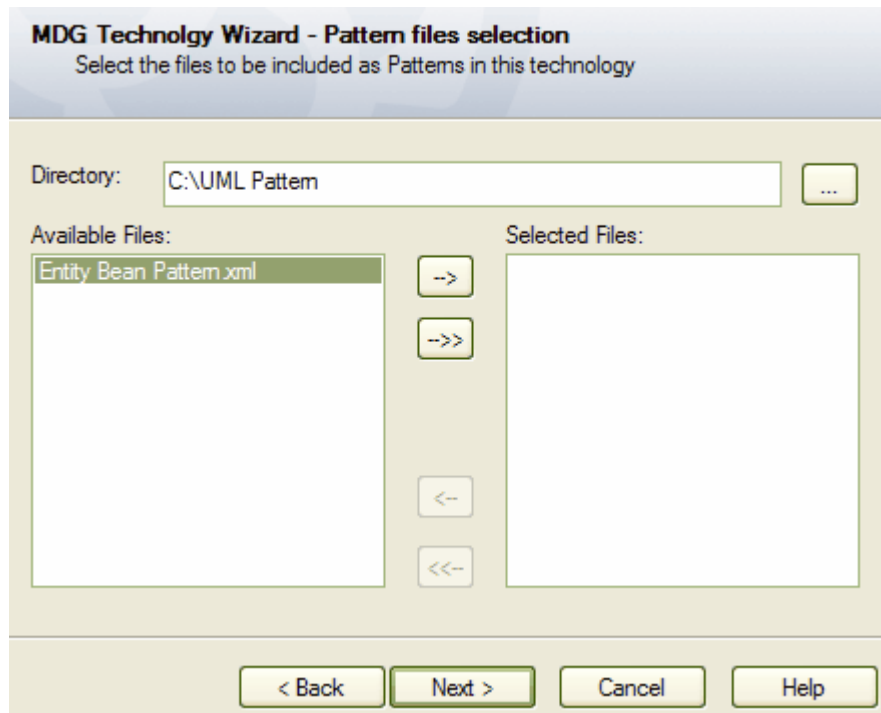
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Profiles* item selected.
2. The *Profile files selection* wizard dialog will then allow the user to navigate to a the directory containing the Profile/s. Once the profile/s have been located select the desired profile/s by highlighting them in the *Available Files* section and pressing the --> button to select the Profile individually or the -->> button to select all of the available Profiles. After the appropriate selections have been made press the *Next* button to proceed.



6.5.1.2 Adding Pattern in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including Patterns. To use the Patterns section of the wizard use the following instructions:

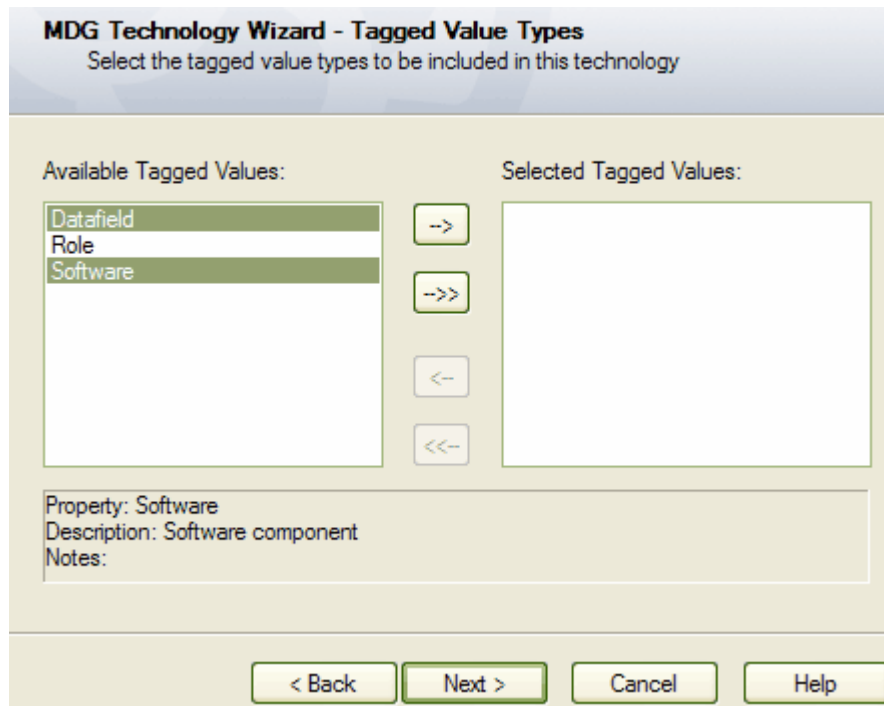
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Pattern* item selected.
2. The *Pattern files selection* wizard dialog will then allow the user to navigate to a the directory containing the Pattern/s. Once the Pattern/s have been located select the desired Pattern highlighting them in the *Available Files* section and pressing the --> button to select the Pattern individually or the -->> button to select all of the available Patterns. After the appropriate selections have been made press the *Next* button to proceed.



6.5.1.3 Adding Tagged Values in MDG Technology Wizard

During the process of creating an MDG Technology file the user may select the option of including Tagged Value Types. To use the Tagged Value Types section of the wizard use the following instructions:

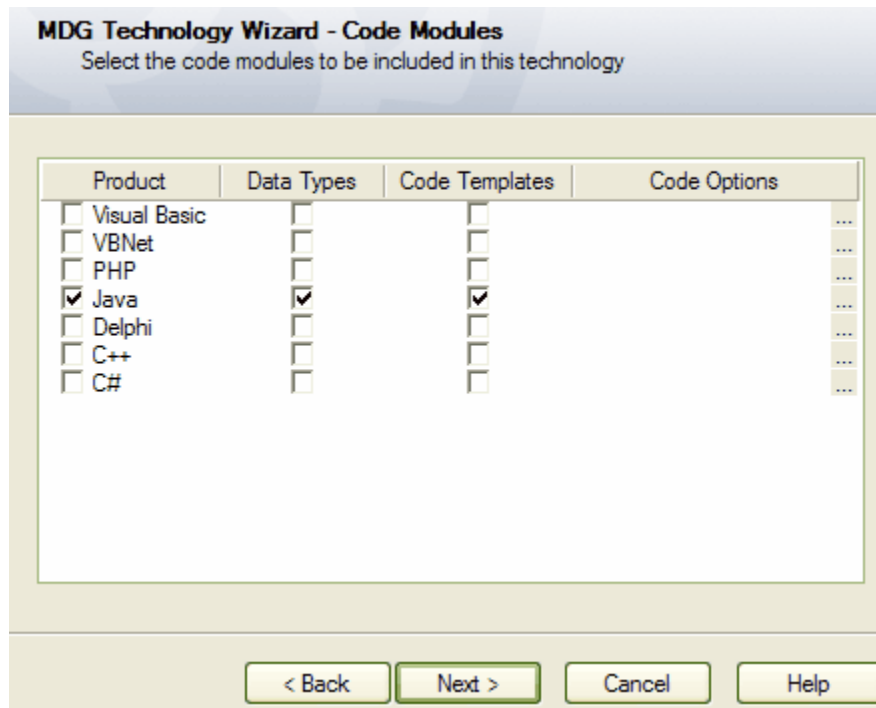
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Tagged Values Type* item selected.
2. The *Tagged Value Type* wizard dialog will then allow the user select the Tagged Values Types that have been defined in the model. Select the appropriate Tagged Value Types by pressing the --> button to select the Tagged Values Type individually or the -->> button to select all of the available Tagged Values Type. After the appropriate selections have been made press the *Next* button to proceed.



6.5.1.4 Adding Code Modules in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including Code Modules. To use the Code Modules section of the wizard use the following instructions:

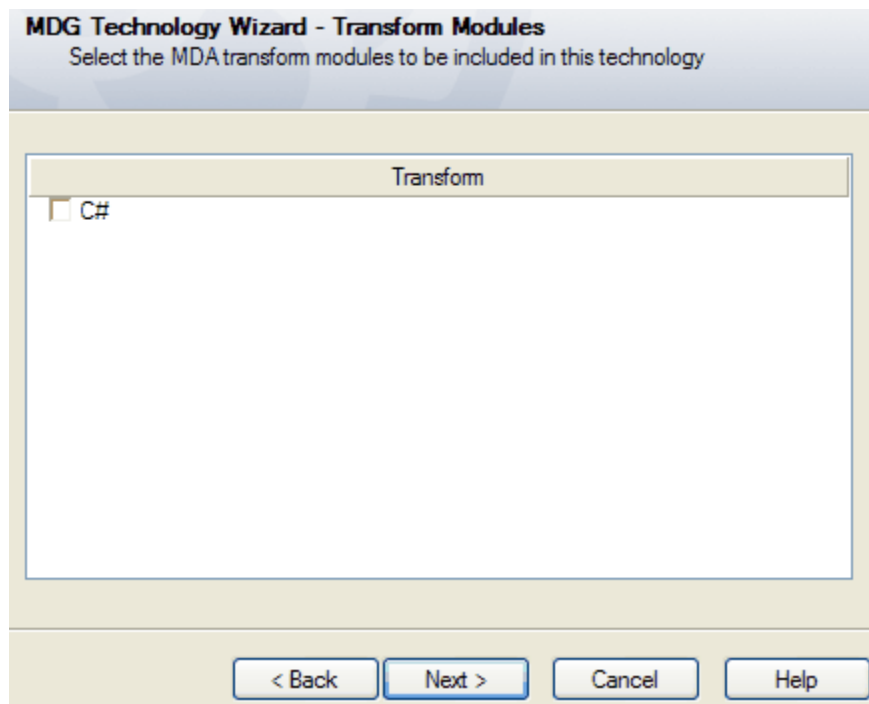
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Code Modules* item selected.
2. The *Code Modules* wizard dialog will then allow the user select the from the available Code Models that are present in the model. Select the appropriate Code Module/s by checking the required Product, Data Types, Code Templates checkboxes. To select the Code Options for each product press the ... button to select the appropriate code options. After the selections have been made press the *Next* button to proceed.



6.5.1.5 Adding MDA Transforms in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including the MDA Transforms that have been modified in the model. To use the Transform Modules section of the wizard use the following instructions:

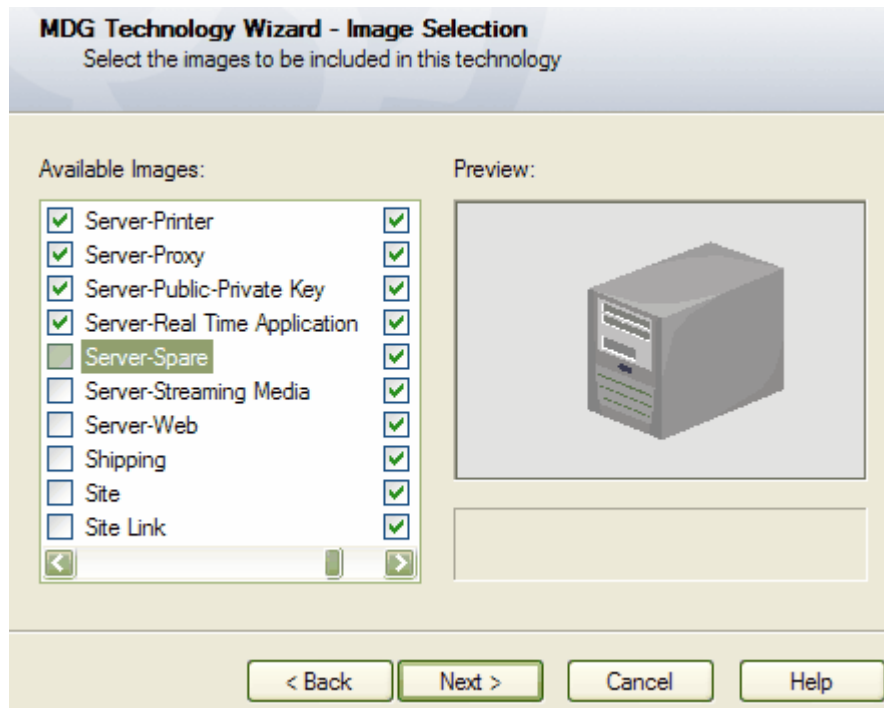
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *MDA Transforms* item selected.
2. The *Transform Modules* wizard dialog will then allow the user select the available model transform templates that are present in the current model. Select the appropriate templates by selecting the checkbox next to the template name. After the selections have been made press the *Next* button to proceed.



6.5.1.6 Adding Images in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including the images that have been imported into the model. To use the Image Selection section of the wizard use the following instructions:

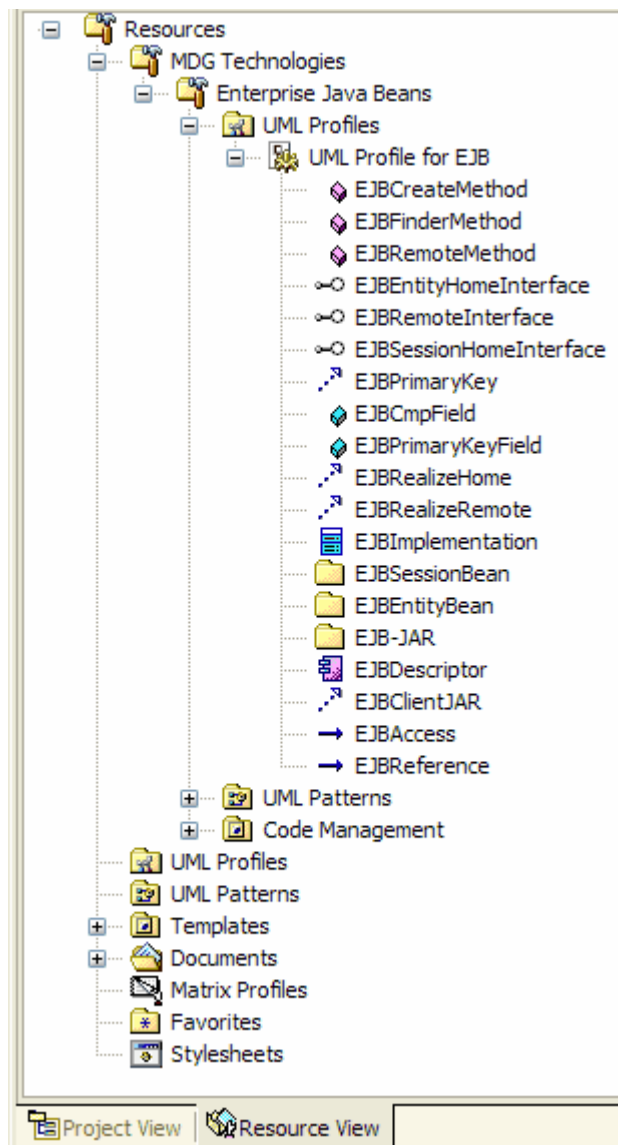
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Images* item selected.
2. The *Image Selection* wizard dialog will then allow the user select the available model images that are present in the current model. Select the appropriate images by selecting the checkbox next to the image name. After the selections have been made press the *Next* button to proceed.



6.5.2 Working with MDG Technologies

The Resource Window

The Resource window is the second tab of the docked Project Browser window. It contains a tree structure with entries for items such as MDG Technologies, Documents, Stylesheets, Matrix profiles and UML Profiles. The MDG Technologies node initially contains no entries - to activate a MDG Technology you must import them into EA from supplied XML files. Once they have been imported, you can open the MDG Technology and work with its elements.



MDG Technologies can bundle up the functionality provided by UML Profiles, UML Patterns, Model Types and Code Templates

Profiles contained in the MDG Technologies can be applied to:

- Elements such as classes and interfaces can be dragged directly from the resource window to the current diagram.
- Attributes can be dragged over a host element (eg. Class) - they will automatically be added to the element feature list.
- Operations are like Attributes - drag over a host element to add the operation.
- Links such as Association, Generalization, and Dependency are added by selecting them in the browser, then clicking the on start element in a diagram and dragging to the end element (in the same manner as adding normal links). The link will be added with the new stereotype and tagged value information.
- Association Ends can be added by dragging the link end element over the end of an Association in the diagram.

Patterns contained in the MDG Technologies can be used to:

- Enable reuse in a model.
- Build in robustness.

Code Templates can be used to:

- Specify the transformation from UML elements into various parts of a given programming language.

Model Types can be used to:

- define the data types for the model.

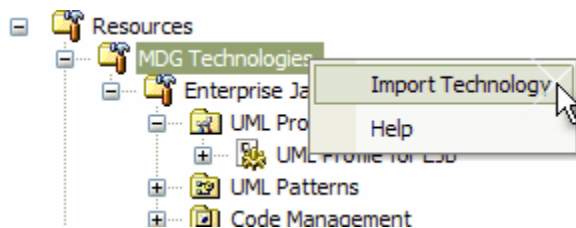
6.5.3 Import MDG Technologies

To import a MDG Technology you will need a suitable MDG Technology XML file. If the MDG Technology includes references to any Metafiles, they should be in the same directory as the XML MDG Technology.

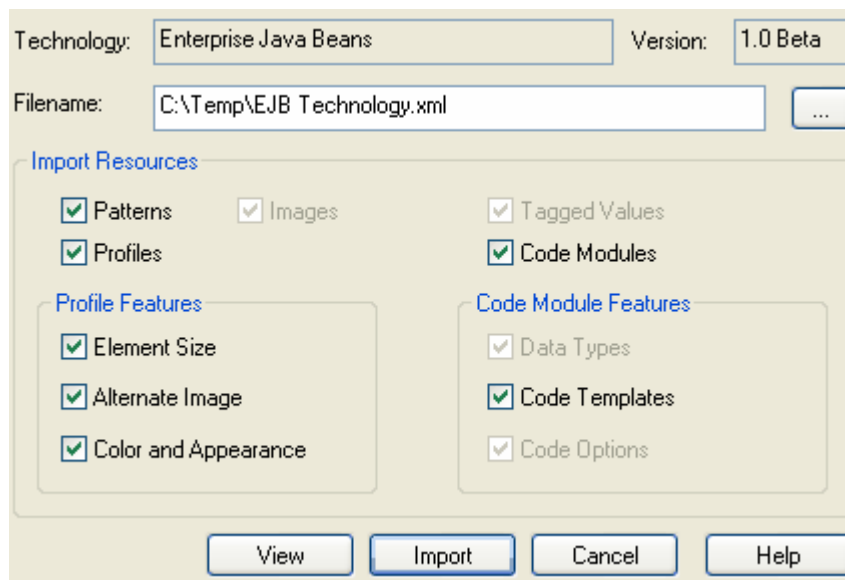
Import a MDG Technology

To import a MDG Technology, follow the steps below:

1. Right click on the MDG Technologies tree node and select *Import Technology* from the context menu in the resource view - as in the example below.



2. The *Import Technology* dialog will open.



Note: Options that are greyed out (such as the Model Types in the image above) indicate that no examples of that type exist in the MDG Technology XML file.

3. Locate the MDG Technology file to import using the *Browse [...]* button.

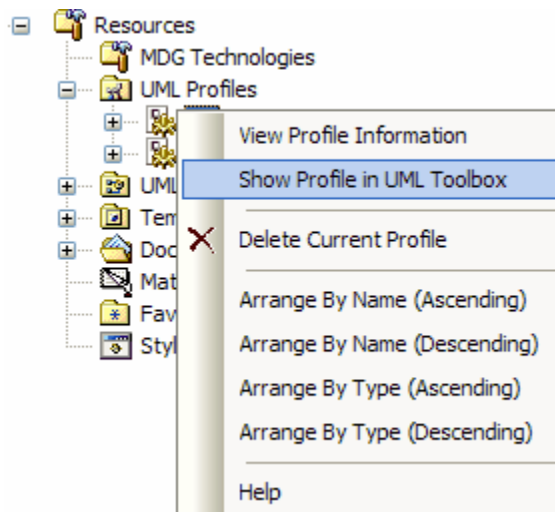
4. Set the required import options for all of the resources defined in the MDG Technology - you can select to import:
 - Element Size yes/no - check this to import the element size attributes.
 - Color and Appearance yes/no - check this to import the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image yes/no - check this to import the metafile image.
 - Patterns yes/no - check this to import patterns if they exist.
 - Profiles yes/no - check this to import profiles if they exist.
 - Code Modules yes/no - check this to import the various languages that are associated with the technology if they exist.
 - Data Types yes/no - check this import the data types.
 - Code Templates yes/no - check this to import the code templates if they exist.
 - Code Options yes/no - check this to import the options which include items such as default file extensions and default file paths.
5. Press *Import*.

If the MDG Technology already exists, EA will offer to overwrite the existing version and import the new one - or cancel. Once the import is complete, the MDG Technology is ready to use.

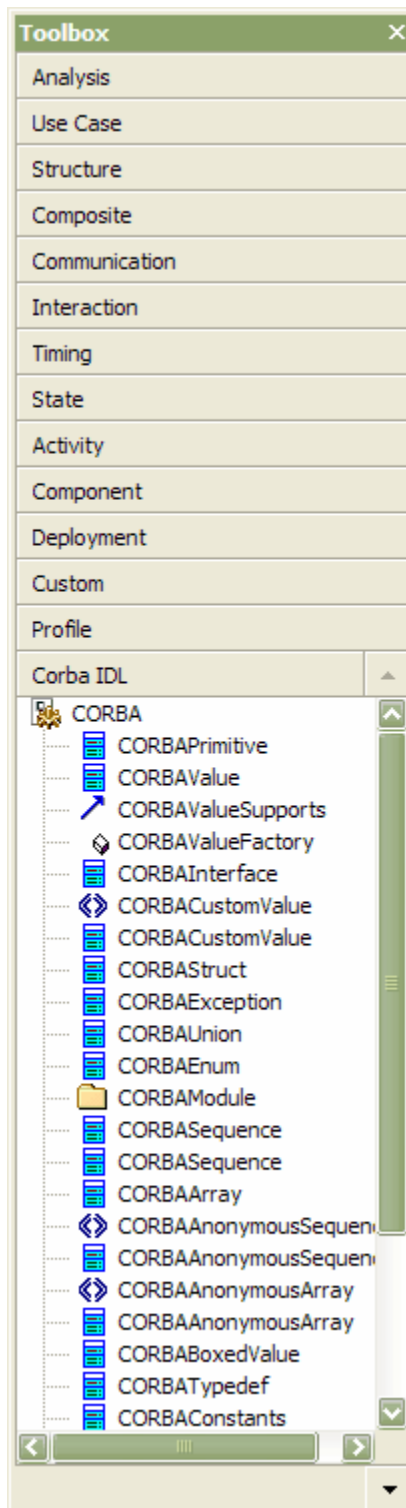
6.5.4 Add MDG Technologies to UML Toolbox

MDG technologies and UML profiles can be added to the UML tool box to enable the quick use of the items contained in the profile or technology file. To add a technology to the UML toolbox use the following instructions:

1. Locate the MDG Technology file in the Resource View.
2. Right click on the technology file in the resource window to bring up its context menu and select the *Show Technology in UML Toolbox* option.



3. This will add the Technology to the UML Toolbox, from here the user can drag the items contained in the Technology file directly into a diagram.



6.6 UML Profiles

What are UML Profiles?

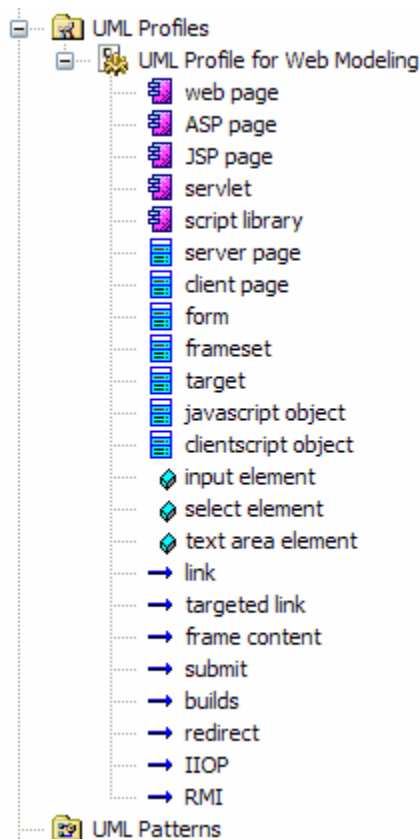
UML Profiles provide a means for extending the UML Language, which allows for building UML models in particular domains. They are based on additional Stereotypes and Tagged values that are applied to Elements, Attributes, Methods, Links, Link Ends, etc. A profile is a collection of such extensions that together

describe some particular modeling problem and facilitate modeling constructs in that domain. For example the UML Profile for XML as defined by David Carlson in the book "Modeling XML Applications with XML" pp. 310, describes a set of extensions to basic UML model elements to enable accurate modeling of XSD Schemas.

Enterprise Architect has a generic UML Profile mechanism for loading and working with different Profiles. UML Profiles for Enterprise Architect are specified in XML files, with a specific format - see the examples in this section. These XML files can be imported into EA in the Resource page of the Project Browser. Once imported, you can drag and drop Profile elements onto the current diagram. EA will attach the stereotype, tagged values and default values, notes and even metafile if one is specified, to the new element. You can also drag and drop attributes and operations onto existing classes and have them immediately added with the specified stereotype, values etc.

Profiles in the Resource View

The Resource window is the second tab of the docked Project Browser window. It contains a tree structure with entries for items such as MDG Technologies, Documents, Stylesheets, Matrix profiles and UML Profiles. The UML Profiles node initially contains no entries - to be able to use profiles you must import them into EA from supplied XML files.



Items in the Profile represent stereotypes. These can be applied to UML elements in the following ways:

- Stereotypes that apply to elements such as **classes** and **interfaces** can be dragged directly from the resource window to the current diagram, automatically creating a stereotyped element. Alternatively, they can be dragged onto existing elements, automatically applying them to the element.
- Stereotypes that apply to **attributes** can be drag-and-dropped onto a host element (eg. class) - a stereotyped attribute will automatically be added to the element's feature list.
- Stereotypes that apply to **operations** are like those that apply to attributes - drag-and-drop onto a host element to add the stereotyped operation.
- Stereotypes that apply to **connectors** such as **associations**, **generalizations** and **dependencies** are added by selecting them in the browser, then clicking on the start element in a diagram and dragging to

the end element (in the same manner as adding normal links). A stereotyped link will be added.

- Stereotypes that apply to **association ends** can be added by dragging the link end element over the end of an association in the diagram.

To get you started, some profiles are supplied on the Sparx Systems website at www.sparxsystems.com.au/uml_profiles.htm. You can download these and import them into EA. Over time we will expand the range of Profiles, the content of each profile and the degree of customization possible in each profile. Remember, you can always create your own profiles to describe modeling scenarios specific to your development environment.

See Also

- [Creating Profiles](#)
- [Using Profiles](#)
- [Profile References](#)

6.6.1 Creating Profiles

This section describes how to create profiles and profile items. These creation tasks include creating the profile stereotypes, defining the metaclasses they apply to, as well as defining tagged values and constraints. This section also describes how to export a profile for use in UML modeling.

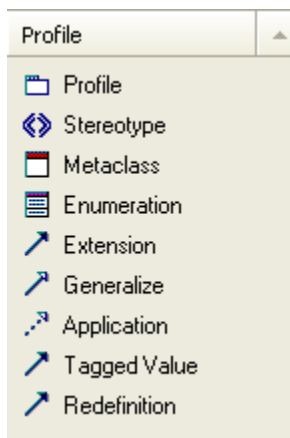
Perform the following steps to create a UML Profile:

1. [Create a Profile Package](#)
2. [Add Stereotypes and Metaclasses](#)
3. [Define Tagged Values for Stereotypes](#)
4. [Define Constraints for Stereotypes](#)
5. [Add Enumerations](#)
6. [Export the Profile](#)

6.6.1.1 Create a Profile Package

Create a Profile

To create a profile, drag the *Profile* item from the Profile section of the UML toolbox onto a diagram.



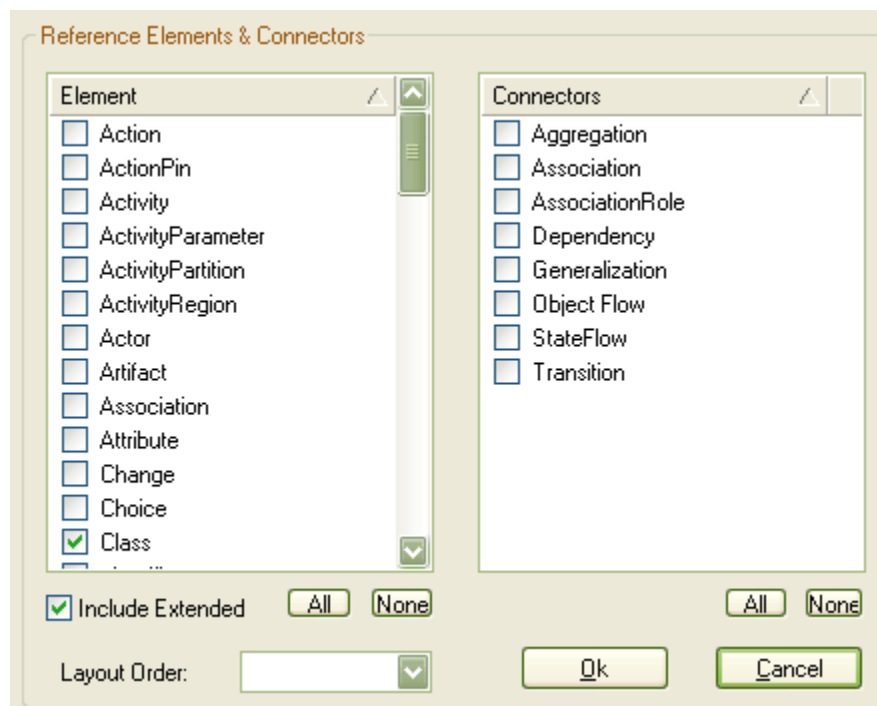
Once you have entered a name for the profile, a package will be created with the «profile» stereotype and a child diagram beneath it. This child diagram will be used to [add stereotypes](#) to the profile.

6.6.1.2 Add Stereotypes and Metaclasses to UML Profiles

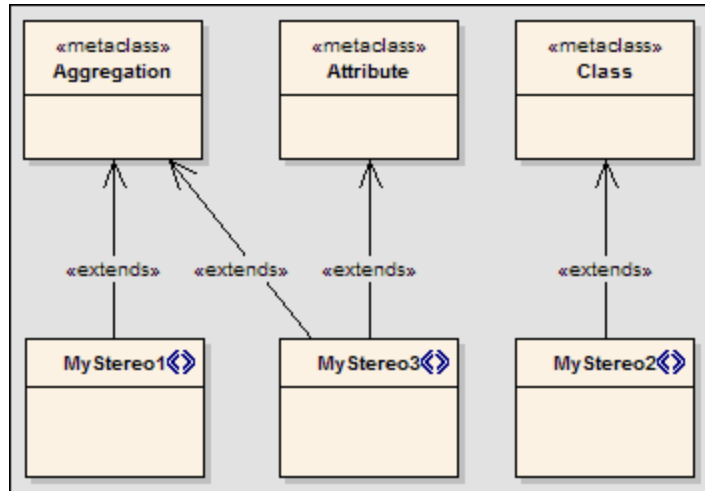
Adding Stereotypes and Metaclasses to a Profile

To add stereotypes to a profile, follow the steps shown below:

1. Open up the diagram contained in the profile package.
2. Create your *Stereotype* elements in the diagram belonging to the profile package by dragging the *Stereotype* tool from the *Profile Toolbox*.
3. Create the *Metaclass* elements which your profile stereotypes will extend by dragging the *Metaclass* tool from the *Profile Toolbox*. This will display the following dialog in which you can tick multiple metaclasses for dropping onto the diagram.



4. *Extend* your metaclasses with your profile stereotypes by creating *Extension* associations in your profile model. This will create a diagram which looks something like the following:



6.6.1.3 Define Stereotype Tags

Stereotypes within a UML Profile may have one or more associated tagged values. In creating a UML Profile, these tagged values are defined as attributes of the stereotype class.

Defining Stereotype Tags

To define Tagged Values for a stereotype, follow the steps shown below:

1. Open the [Attributes](#) dialog for the stereotype element.

| Name | Type | Scope |
|---------|---------|---------|
| keytrue | boolean | Private |

2. Press **New** to create a new attribute.
3. Enter the name of the stereotype tag in the **Name** field.
4. Set the **Type** and the **Initial** value of the tag. (See [Add Enumerations to UML Profiles](#) for instructions on creating enumerated types for tagged values).
5. Add a description of the tag into the **Notes** field.
6. Press **Save**

See Also

- [Define Stereotype Tags with Predefined Tag Types](#)
- [Define Stereotype Tags with Supported Attributes](#)
- [Using the Tagged Value Connector](#)

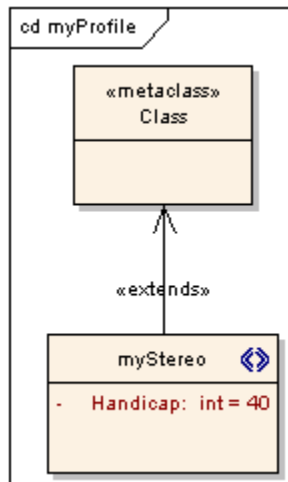
6.6.1.3.1 Define Stereotype Tags with Predefined Tag Types**Defining Predefined Tag Types**

To define a stereotype tag with a predefined tag type, you must first create the predefined tag type. Full instructions in how to do this are given in [The Tagged Values Window](#), and specifically in [Predefined Tagged Value Types](#).

Assigning Predefined Tag Types to Stereotypes

It is very simple to assign a predefined tag type to a stereotype. Simply create an attribute with the same

name. For example, to have the example Tagged Value "Handicap" from the [Predefined Tagged Value Types](#) page appear in a stereotype, create an attribute named "Handicap". You can set the default value for the tagged value by giving the attribute an "Initial" value.



6.6.1.3.2 Define Stereotype Tags with Supported Attributes

Defining Stereotype Tags with Supported Attributes

Supported stereotype attributes tags are special tags which set the default behavior of stereotyped elements such as the initial size of the element and the default location of any image files associated with the stereotype, etc. To define Tags for a stereotype with the supported attributes, follow the steps shown below (for a list of supported attributes go to the [supported attributes](#) section):

1. Open the [Attributes](#) dialog for the stereotype element.

General Detail Constraints

Name:

Type: Derived Static

Scope: Property Const

Stereotype:

Containment:

Alias:

Initial:

Notes:

Attributes

| Name | Type | Scope |
|---------------|------------|----------------|
| this is a tag | boolean | Private |
| _sizeX | int | Private |
| enumb | enum | Private |

OK Cancel Help

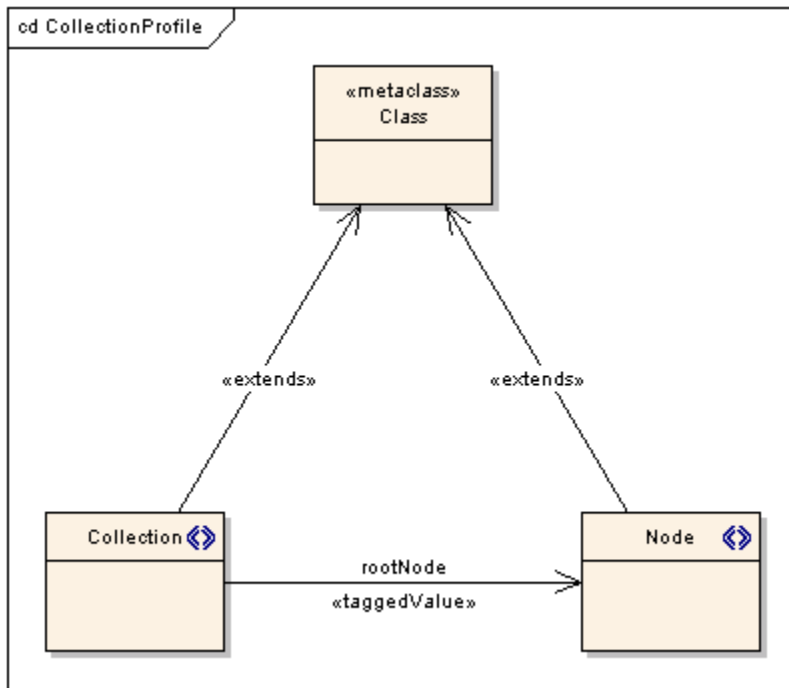
Note: For supported attributes set only the *Name* (which needs to match the attributes listed in the supported attributes section) and the *Initial* value, do not set the other values.

2. Enter the name of the stereotype Tag in the *Name* field.
3. Set the initial value of the tag.
4. Press *Save*

6.6.1.3.3 Using the Tagged Value Connector

The Tagged Value connector, found in the Profile toolbox, can be used to define a tagged value which will take as its value an element which has the stereotype pointed to.

The following example illustrates how this might be used. It shows a profile which defines two stereotypes, «Collection» and «Node». «Collection» will add a tag named "rootNode". Clicking in the value field for the tag rootNode in the Tagged Values docked window will allow selection from a list of all elements in the current model with the «Node» stereotype.



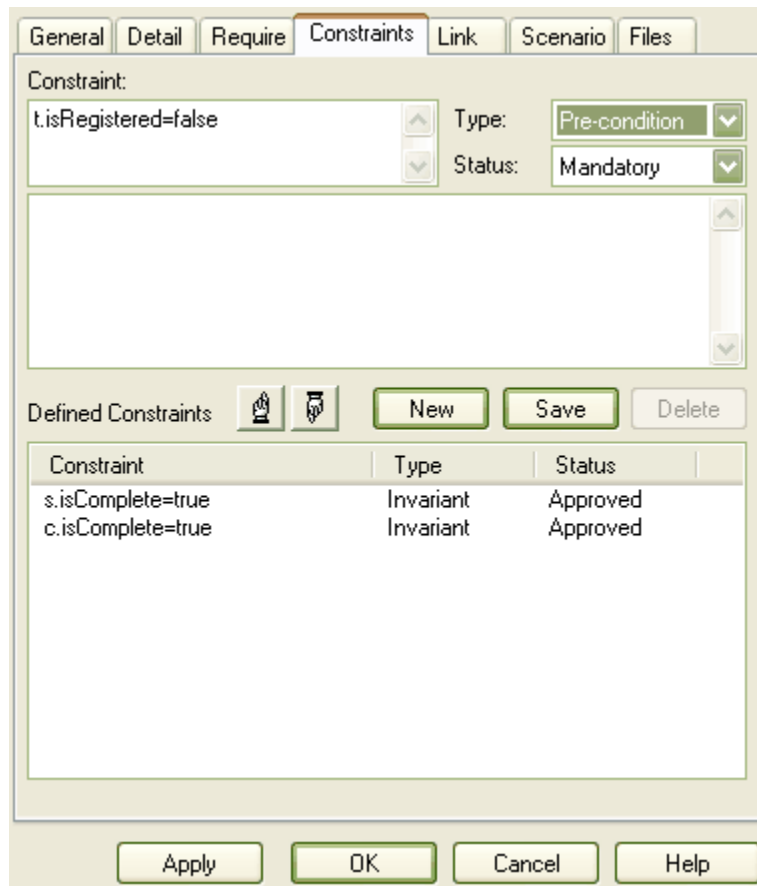
6.6.1.4 Define Stereotype Constraints

Defining constraints for stereotypes follows the same procedure as defining constraints for any class.

Defining Stereotype Constraints

To define constraints for a stereotype, follow the steps shown below:

1. Open the Class Properties dialog of the stereotype element in a diagram.
2. Open the *Constraints* tab and press *New* to create a new constraint.



3. Enter the value (in the *Constraint* field), the *Type* and the *Status* of the Constraint.
4. Enter any additional information in the Notes field.
5. Press *Save*.

6.6.1.5 Add Enumerations to UML Profiles

Enumerations may be used to restrict the values available to stereotype tags.

Note: Enumerations defined under a profile package do not appear as elements in the profile when imported.

Adding Enumerations Elements to UML Profiles

To add an enumeration element, follow the steps detailed below:

1. Drag an *Enumeration* item from the Profile section of the UML toolbox onto the diagram.
2. Set the name of the new Enumeration into the *Name* text field.
3. Open the *Detail* tab of the Enumeration and press the *Attributes* button.

| Name | Type | Scope |
|------------|------|---------|
| redValue | int | Private |
| blueValue | int | Private |
| greenValue | int | Private |

4. Enter the name of the enumeration attribute in the *Name* field.
5. Set the *Type* and the initial value of the enumeration attribute.
6. Add any relevant notes into the notes text field.
7. Press the *Save* button.
8. Open the attribute properties for a stereotype.
9. Use the enumeration defined above in the attribute *Type* field to provide a drop-down list for setting the value of the tag in the Tagged Values window.

6.6.1.6 Add Redefinitions to UML Profiles

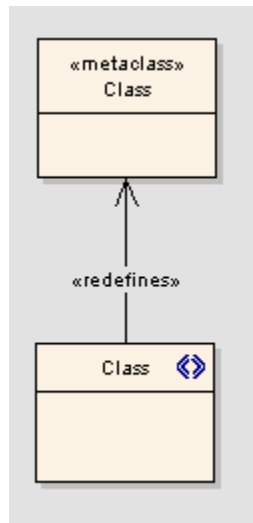
Redefinitions allow you to include standard EA elements and connectors in UML Profiles. This helps you to design a UML Profile as a complete modelling solution, providing a means to drag stereotyped elements and unadorned elements from the same toolbox.

Adding Redefinitions to UML Profiles

To add a redefinition, follow the steps detailed below:

1. Drag a *Metaclass* element from the Profile section of the UML toolbox onto the diagram.
2. Define the *Metaclass* element as described in the section [Add Stereotypes and Metaclasses to UML Profiles](#).

3. Drag a *Stereotype* element from the Profile section of the UML toolbox onto the diagram. The name currently isn't used by EA, so give it any name you like.
4. Draw a *Redefinition* connector from the *Stereotype* element to the *Metaclass* element.



Once you save this as a UML Profile and import it into the Resource Browser, you will be able to drag unadorned Class elements from the profile onto a diagram.

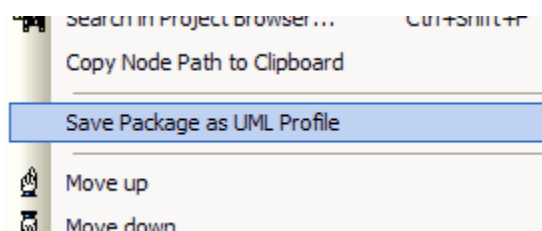
6.6.1.7 Export a UML Profile

Once a profile has been created and the elements and metaclasses have been defined it is then possible to save the profile for future UML models.

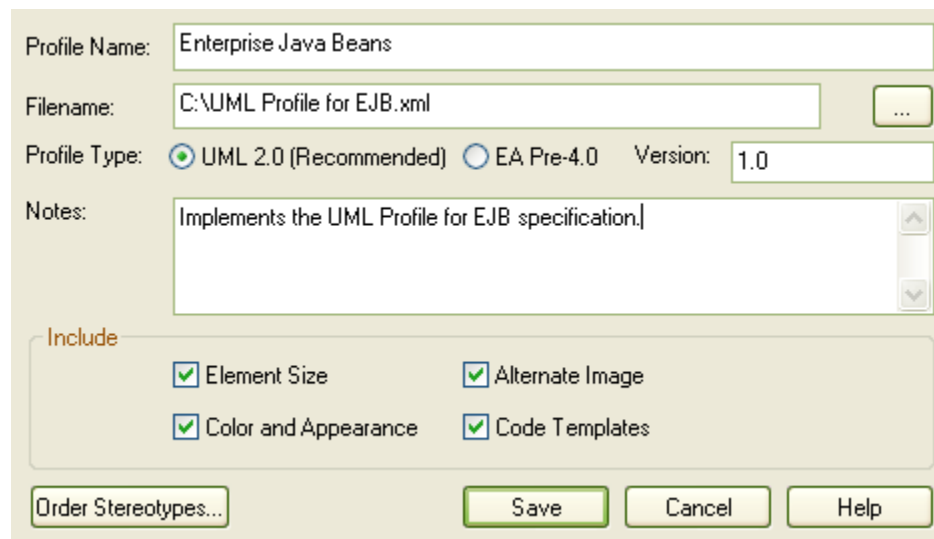
Create a Profile

To save a profile, follow the steps shown below:

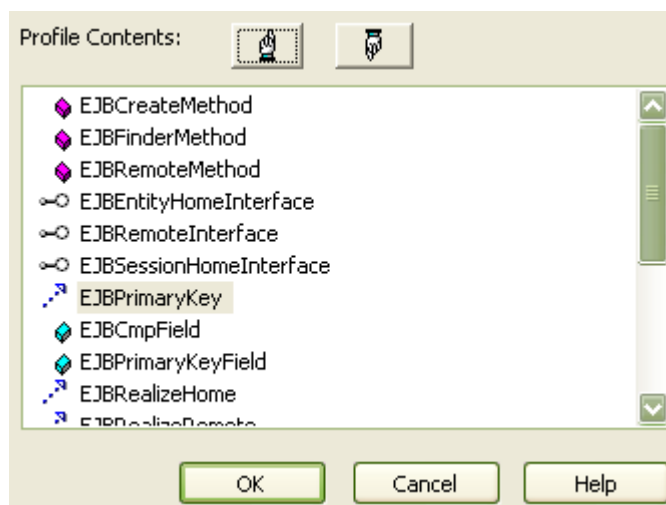
1. When you have completed defining your profile you can save your profile to disk by right-clicking anywhere in the profile diagram, or on the profile package in the *Project View* and select *Save Package as UML Profile*.



2. The *Save UML Profile* Dialog will then be opened



3. Determine the destination for the XML Profile file to be exported using the *Browse [...]* button.
4. Set the required export options for all stereotypes defined in the profile:
 - Element Size - check this to export the element size attributes.
 - Color and Appearance - check this to export the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image - check this to export the metafile images.
 - Code Templates - check this to export the code templates if they exist.
5. Set the Profile Type to UML 2.0 (*note: it is still possible to create stereotypes using the same format as for pre-4.0 versions of EA. This has a reduced feature set and is provided for legacy use*).
6. Click the *Order Stereotypes* button to define the order that you wish elements to appear in the generated profile (and therefore in the Resource Browser and UML Toolbox).



In this dialog, you can select individual items and use the Hands to move the items up or down in the list, or you can right-click in the work area and choose to order in ascending or descending alphabetical or type order.

7. Press **Save** to save the profile to disk.

Your profile, and its stereotypes, will now be available to use with other EA models by using the [Import Profile](#) option.

6.6.2 Using Profiles

This section describes the use of profiles for UML modeling. It describes tasks including how to import an available profile for use in a model, how to create elements and connectors using the stereotypes contained in the profile, applying and synchronizing values and constraints.

The following topics are discussed in this section:

- [Import a UML Profile](#)
- [Add Profiles to UML Toolbox](#)
- [Tagged Values in Profiles](#)
- [Add Profile Connector to Diagram](#)
- [Synchronize Tags and Constraints](#)

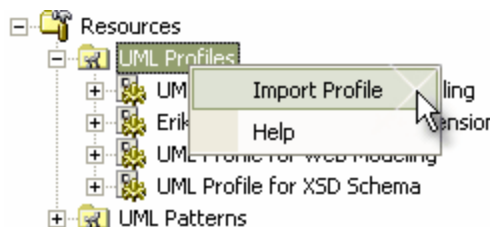
6.6.2.1 Import a UML Profile

To import a profile you will need a suitable Profile XML file - for example, like the profiles supplied on the Sparx Systems website at www.sparxsystems.com.au/uml_profiles.htm. If the Profile includes references to any Metafiles, they should be in the same directory as the XML profile.

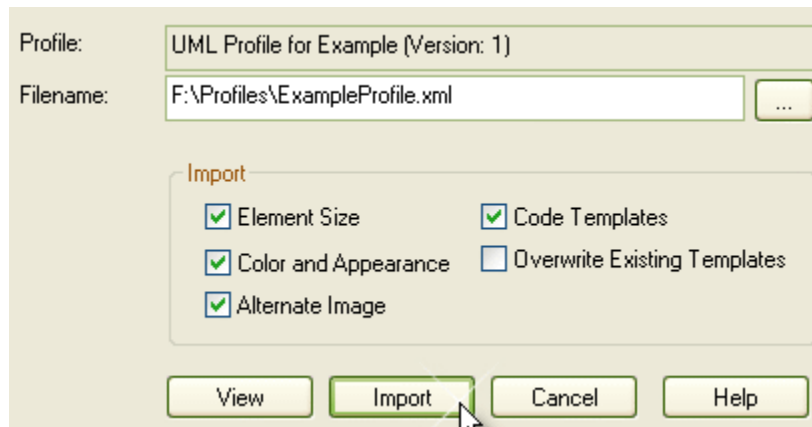
Import a Profile

To import a profile, follow the steps below:

1. Right click on the UML Profiles tree node and select **Import Profile** from the context menu.



2. The **Import UML Profile** dialog will open.



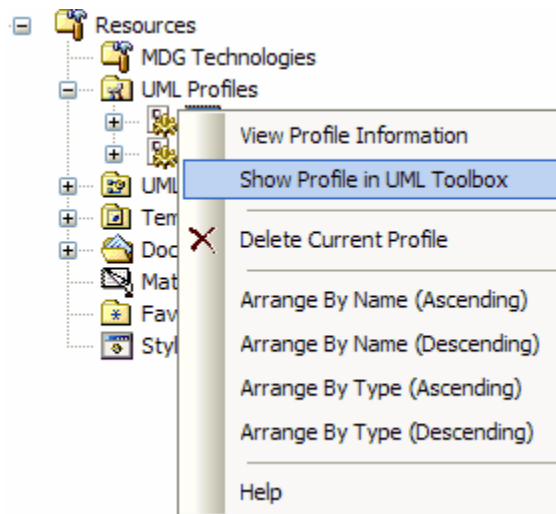
3. Locate the XML Profile file to import using the *Browse [...]* button.
4. Set the required import options for all stereotypes defined in the profile - you can select to import:
 - Element Size - check this to import the element size attributes.
 - Color and Appearance - check this to import the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image - check this to import the metafile image.
 - Code Templates - check this to import the code templates if they exist.
 - Overwrite Existing Templates - check this to overwrite any existing code templates defined in the current project.
5. Press *Import*.

If the profile already exists, EA will offer to overwrite the existing version and import the new one - or cancel. Once the import is complete, the profile is ready to use.

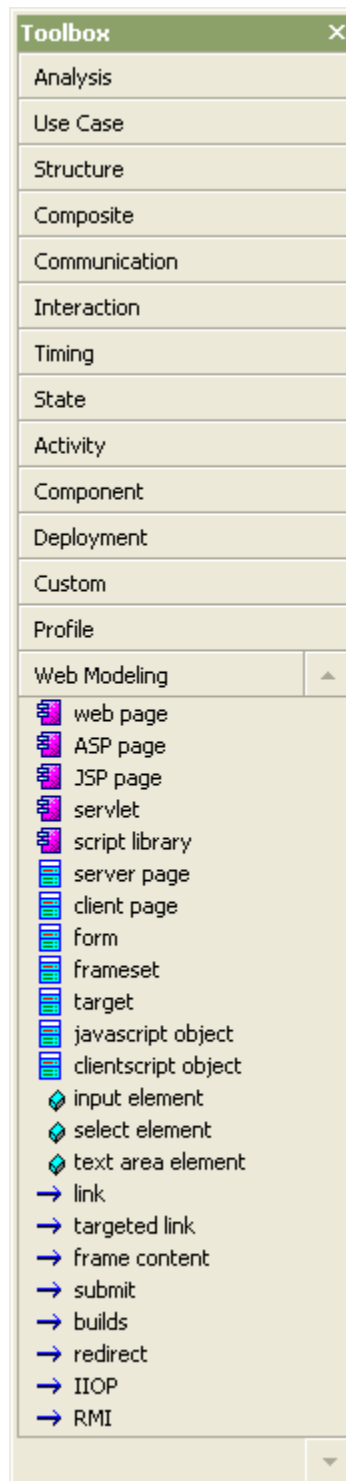
6.6.2.2 Add Profiles to UML Toolbox

MDG technologies and UML profiles can be added to the UML Toolbox to enable the quick use of the items contained in the profile or technology file. To add a profile to the UML Toolbox do the following:

1. Locate the Profile in the Resource View.
2. Right click on the Profile name in the resource window to bring up its context menu and select the *Show Profile in UML Toolbox* option.



3. From the UML Toolbox, you can drag the items contained in the Profile directly into a diagram.



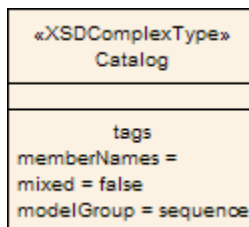
6.6.2.3 Tagged Values in Profiles

Stereotypes within a UML Profile may have one or more associated tagged values. When you create an element based on a UML Profile Stereotype by dragging from the Resource Tree to a diagram, any associated tagged values are added to the element as well. Tagged Values and Profiles are an excellent way to extend the usage of EA and the power of UML modeling.

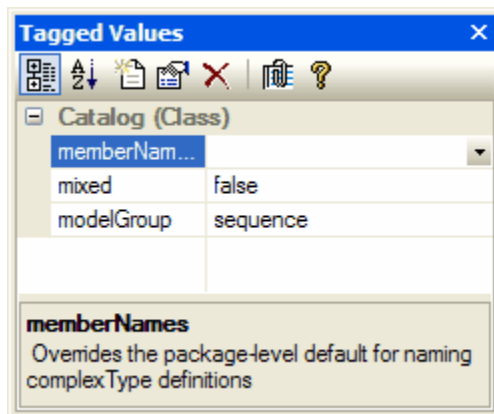
For example, in the UML Profile for XSD, there is an XSDComplexType stereotype, which has the following tagged value declaration

```
<TaggedValues>
  <Tag name="mixed" description="Determines whether this element may contain mixed element and character content. Refer to the W3C XML Schema recommendation" />
  <Tag name="modelGroup" description="Overrides the package-level default model group" values="all | sequence | choice" default="choice" />
  <Tag name="memberNames" description="Overrides the package-level default for naming complexType definitions" />
</TaggedValues>
```

When you create an element from the XSDComplexType stereotype (by dragging from the UML Profile section onto a diagram), the tagged values are added automatically.



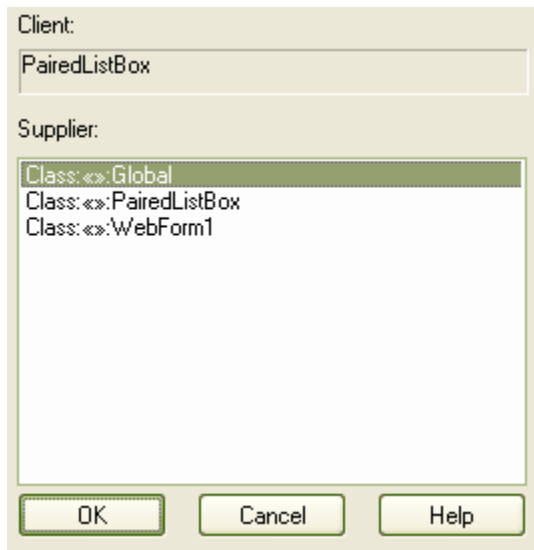
Tagged values which have default values are automatically set and displayed in the element tags section if applicable. When you select the element, the Tagged Values docked window displays all the associated tags - including ones that have no value set. Also note that tagged values in the Profile that have a 'Values' section (eg. values="element | attribute | both" default="both") will display in the Property Browser with a drop list of allowable values when selected (as in the example below). Where no Value list exists, the tag accepts free text.



6.6.2.4 Add Profile Connector to Diagram

To add a profile based connector to the current diagram, left click on the connector in the resource tree, then left click the source element in the diagram and drag it to the target.

You can also drag the connector from the Resource Window to the source and use the list box below to select the target.



6.6.2.5 Synchronize Tags and Constraints

When you first create an element, attribute, operation or link from a UML Profile item, tagged values and constraints may be created as well. Over time it may be that you modify the tagged values and constraints associated with a particular element - so the items already created may be missing additional tagged values or constraints.

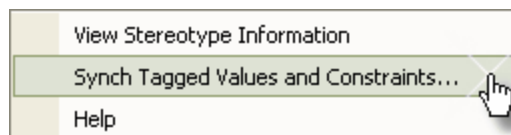
Likewise, you may have manually set the stereotype on a set of elements - and now want them to receive the tagged values and constraints normally associated with that stereotype.

To make sure you have all the related tagged values and stereotypes, use the *Synch Tagged Values and Constraints* function.

Synchronize Elements

To synchronize elements, follow the steps below:

1. Locate the required UML Profile in the Resource tree.
2. Locate the stereotyped profile element.
3. Right click and select the *Synch Tagged Values and Constraints* option.



4. When the *Synchronize tagged values and constraints dialog* appears - press *OK* to proceed - the list will be populated with the items that have been modified and the change that was made.

Synchronize all stereotyped elements with the current profile stereotype to have the default tags and constraints.

Stereotype:

Type:

Actions:

| Element Name | Tag or Constraint Added |
|--------------|-------------------------|
| | |

OK Cancel Help

6.6.3 Profile References

UML Profile XML File Format Information

Enterprise Architect provides a facility to import pre-defined elements, operations, attributes and connectors as a source of re-useable components that meet common modeling needs (eg. profiles for XML schema, for business process modeling etc.). This section provides a quick list of the types of things that can be pre-defined and the characteristics of each. There is an example file at the end which indicates how each type is specified.

This section gives the user a reference of the supported tags that are available for users, the structure of profiles, supported attributes and an example of the XML file that constitutes a profile.

UML Profile Reference Subjects:

- [Supported Types](#)
- [Profile Structure](#)
- [Supported Attributes](#)
- [Example Profile](#)

6.6.3.1 Supported Types

A UML profile is made up of one or more stereotypes that may have tagged values and constraints. The table below and the [Supported Attributes](#) table defines what can be stereotyped and what information needs to be supplied.

List of All Supported Types in AppliesTo/Apply Node

| AppliesTo/ Apply | Type | Tags | Constraint | Metafile |
|------------------|---------|------|------------|----------|
| "actor" | Element | Yes | Yes | Yes |
| "package" | Package | Yes | Yes | Yes |
| "usecase" | Element | Yes | Yes | Yes |
| "collaboration" | Element | Yes | Yes | Yes |
| "class" | Element | Yes | Yes | Yes |
| "table" | Element | Yes | Yes | Yes |
| "component" | Element | Yes | Yes | Yes |
| "node" | Element | Yes | Yes | Yes |
| "object" | Element | Yes | Yes | Yes |
| "sequence" | Element | Yes | Yes | Yes |
| "entity" | Element | Yes | Yes | Yes |
| "screen" | Element | Yes | Yes | Yes |

| | | | | |
|------------------|----------------|-----|-----|-----|
| "GUIElement" | Element | Yes | Yes | Yes |
| "requirement" | Element | Yes | Yes | |
| "state" | Element | Yes | Yes | |
| "activity" | Element | Yes | Yes | Yes |
| "interface" | Element | Yes | Yes | Yes |
| "event" | Element | Yes | Yes | |
| "issue" | Element | Yes | Yes | |
| "change" | Element | Yes | Yes | |
| "hyperlink" | Element | | Yes | |
| "attribute" | Attribute | Yes | Yes | |
| "operation" | Operation | Yes | Yes | |
| "association" | Connector | Yes | Yes | |
| "associationEnd" | AssociationEnd | | | |
| "generalization" | Connector | Yes | Yes | |
| "dependency" | Connector | Yes | Yes | |
| "stateflow" | Connector | Yes | Yes | |
| "objectflow" | Connector | Yes | Yes | |
| "startnode" | Element | Yes | Yes | |
| "stopnode" | Element | Yes | Yes | |
| "note" | Element | Yes | Yes | |
| "decision" | Element | Yes | Yes | |
| "aggregation" | Connector | Yes | Yes | |

6.6.3.2 Profile Structure

UML Profiles for Enterprise Architect are distributed in XML format. The file follows the following format:

General Header Details

The following is the general header details:

```
<?xml version="1.0" encoding="utf-8" ?>
<UMLProfile profiletype="uml2">
  <!--Profile name, version number and general notes -->
  <Documentation id="XSDSchema" name="UML Profile for XSD Schema" version="1.0" notes="Defines a set of
  stereotypes and tagged values for XSD Schemas" />
  <!--The profile content -->
  <Content>
    <!--List of stereotypes used in this profile. May also include tagged values, constraints, metafile and
    descriptive comments-->
    <Stereotypes>
```

Stereotype Definitions

This is followed by one or more Stereotype definitions - for example:

```
<!-- <XSDComplexType> -->
<Stereotype name="XSDComplexType" notes="ComplexType definition generated in XML Schema">
<AppliesTo>
  <Apply type="class" />
</AppliesTo>
<TaggedValues>
  <Tag name="mixed" description="URI to unique target namespace" />
  <Tag name="modelGroup" description="Default model group used when generating complexType definitions for this
  Schema" values="all | sequence | choice" default="choice" />
  <Tag name="attributeMapping" description="Default for generating UML attributes as elements, attributes or both
  within complexTypes" values="element | attribute | both" default="both" />
  <Tag name="roleMapping" description="Prefix associated with namespace" />
```

```

    <Tag name="memberNames" description="Schema version"/>
  </TaggedValues>

  <Constraints>
    <Constraint name="" type="" notes="" />
  </Constraints>
</Stereotype>

```

Note the specification of Stereotype name and notes. Also note the use of tagged values to set properties for the profile element. The tagged values may have a default value, may be empty and may specify allowable values. Tagged values are edited in the properties window of an element, method, attribute or link.

You can also specify default size, default comment and Metafile for drawing element - see the fragment below:

```
<Stereotype name="Router" cx="130" cy="100" notes="" metafile="router.emf">
```

In the above example, the metafile shape for this element is specified as 'router.emf' - when you load this profile, the .emf file MUST be in the same directory as the Profile - or the load will fail.

Also note how to specify a default comment for an element. All white space between lines will be ignored. To force a line break, use the '\n' character. To force tabs, use '\t'.

```

<Comment>
  Some text here about how this will work\n\t
  with comments being imported from the XML description
  in one long row.
</Comment>

```

The example above would import like this:

```

Some text here about how this will work
with comments being imported from the XML description in one long row.

```

6.6.3.3 Supported Attributes

Details of Attributes Supported by Main XML Element Nodes

The table below lists the three main types of element you can define in an XML Profile document. These are the "stereotype" (each stereotype will create a visible entry in the Resource Tab/UML Profile window in EA). The "tagged values" are additional properties that an element or link support and "constraints" that apply to the model element.

| Type | Attribute | Optional | Notes |
|------------|------------|----------|---|
| stereotype | name | No | Stereotype name |
| | notes | Yes | Notes visible in browser |
| | metafile | Yes | Filename of associated metafile. MUST be in same directory as Profile XML |
| | cx | Yes | Initial x coordinate of element (deprecated) |
| | cy | Yes | Initial y coordinate of element (deprecated) |
| | _sizeX | Yes | Initial x coordinate of element |
| | _sizeY | Yes | Initial y coordinate of element |
| | _imageFile | Yes | Location of image file (wmf) |

| | | | |
|------------|-------------|-----|--|
| tag | name | No | Tag name |
| | description | Yes | A description of the tag - appears in tag tab and for elements in property window setting notes |
| | values | Yes | List of possible values. Values separated by " " (<space> <space>). eg. "true false". For elements will populate drop combo in tag section of docked properties window |
| | default | Yes | A default value. eg. "true" |
| constraint | name | Yes | constraint name |
| | type | Yes | constraint type (eg. pre for precondition, post for postcondition) |
| | notes | No | Additional explanatory notes |
| | | | |

6.6.3.4 Example Profile

Below is an example UML Profile showing the structure of file and usage:

```
<?xml version="1.0" encoding="UTF-8"?>
<UMLProfile>
  <Documentation id="EAExample" name="UML Profile for Example" version="1" notes="An example set of stereotypes
and tagged values" />
  <!--The profile content -->
  <Content>
    <!--List of stereotypes used in this profile-->
    <Stereotypes>
      <!--A profile is a list of stereotypes, that will apply to elements, links and features in a UML model. Stereotypes
may have set tagged values, constraints,
Valid targets, default dimensions etc. The examples below are a good starting point -->
      <Stereotype name="SimpleStereotype" notes="Place notes about stereotype here" metafile="router.emf">
        <!--Place a list of types that this will apply to...
valid types are any UML element (class, interface, component etc.,
aggregation, generalization, association, stateflow etc., operation and attribute. Make sure you use
lowercase names, XML is case sensitive-->
        <AppliesTo>
          <Apply type="class"/>
          <Apply type="interface"/>
          <Apply type="node"/>
        </AppliesTo>
        <!--Add one or more tagged values for this stereotype. These will be automatically added to the target
element when created
Note that you can specify a default value using "default=" and a pick list of values eg. " true | false" note
the use
of a " | " to separate values -->
        <TaggedValues>
          <Tag name="hasNamespace" description="Indicates element is bound to Namespace" default="true"
values="true | false"/>
          <Tag name="targetNamespacePrefix" description="Prefix associated with namespace" />
        </TaggedValues>
        <!-- Zero or more constraints to apply to element - specify name, type and notes -->
```



```

        <Constraints>
            <Constraint name="constraint1" type="pre" notes="MyNotes"/>
        </Constraints>
    </Stereotype>
    <!-- End of stereotype. When writing your own, you can duplicate a stereotype selection as above and
    change it to start work on a new stereotype-->
    <!-- <<AnotherExample>> -->
    <Stereotype name="AnotherExample" cx="130" cy="100" notes="This element has a default height and width
specified">
        <AppliesTo>
            <Apply type="class"/>
            <Apply type="operation"/>
            <Apply type="attribute"/>
        </AppliesTo>
        <TaggedValues>
            <Tag name="memberNames" description="Schema version"/>
        </TaggedValues>
        <Constraints>
            <Constraint name="constraint1" type="pre" notes="MyNotes"/>
        </Constraints>
    </Stereotype>
    <!-- <<Aggregation>> -->
    <Stereotype name="aggregationLink" type="weak" notes="">
        <AppliesTo>
            <Apply type="aggregation"/>
        </AppliesTo>
    </Stereotype>
    <!-- <<Composition>> -->
    <Stereotype name="compositionLink" type="strong" notes="">
        <AppliesTo>
            <Apply type="aggregation"/>
        </AppliesTo>
    </Stereotype>
    <!-- <<IndexKey>> -->
    <Stereotype name="UniqueID" notes="">
        <AppliesTo>
            <Apply type="operation"/>
        </AppliesTo>
        <TaggedValues>
            <TagName="indexed" description="indicates if indexed or not" values="true | false" default="true"/>
        </TaggedValues>
        <Constraints>
            <Constraint name="constraint1" type="pre" opType="pre" notes="MyNotes"/>
            <Constraint name="constraint2" type="pre" opType="post" notes="MyNotes"/>
        </Constraints>
    </Stereotype>
    <!-- <<Attribute>> -->
    <Stereotype name="attname" notes="">
        <AppliesTo>
            <Apply type="attribute"/>
        </AppliesTo>
        <Constraints>
            <Constraint name="constraint1" type="pre" notes="MyNotes"/>
        </Constraints>
    </Stereotype>
    <!-- <<Association>> -->
    <Stereotype name="assocname" notes="">
        <AppliesTo>
            <Apply type="association"/>
        </AppliesTo>
        <Constraints>
            <Constraint name="constraint1" type="pre" notes="MyNotes"/>
        </Constraints>
    </Stereotype>
</Stereotypes>
</Content>
</UMLProfile>

```

6.7 UML Patterns

What is a Pattern?

Patterns are parameterized collaborations - that is they are a group of collaborating objects/classes that can be abstracted from a general set of modeling scenarios. Patterns are an excellent means of achieving re-use and building in robustness. As patterns are discovered in any new project, the basic pattern template from previous engagements can be re-used with the appropriate variable names modified for the current project.

Patterns generally describe how to solve an abstract problem, and it is the task of the pattern user to modify the pattern elements to meet the demands of the current engagement.

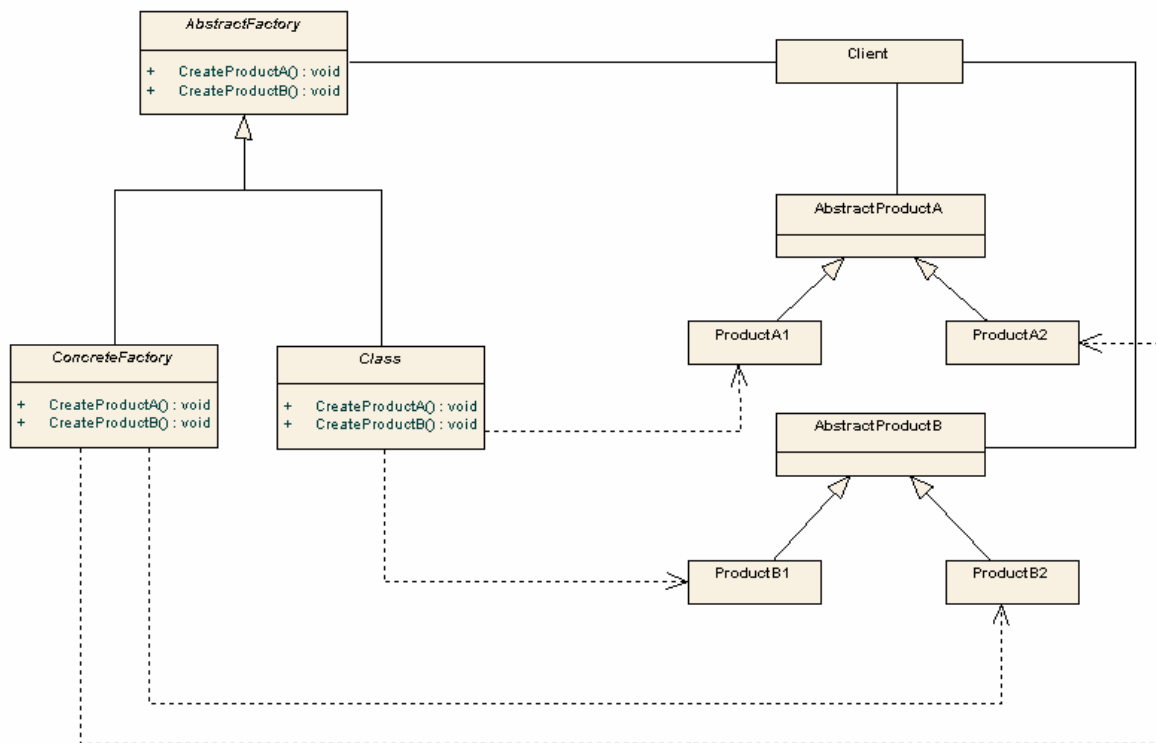
Before using a pattern it must first be [created](#) as a standard UML diagram and then saved as a XML pattern file. This XML file can then be [imported](#) as a UML Resource that may be [used](#) in any model.

Sparx created GoF Patterns

To get you started with Design Patterns in Enterprise Architect, the following zip file contains the Patterns described in the book "Design Patterns - Elements of Reusable Object-Oriented Software" by Gamma et al., referred to as the 'Gang of Four' or GoF for short. To download an example of the Gang of Four patterns for Enterprise Architect from www.sparxsystems.com.au/uml_patterns.html.

6.7.1 Create a Pattern

To create a pattern you will first need to model the pattern as standard UML Diagram from within EA. The following diagram was created from an example as set out in the GOF Design Patterns book.



Save a Diagram as a Pattern

To save a diagram as a pattern, follow the steps below:

1. From the *Diagram* menu, select *Save as UML Pattern* to open the *Save Diagram as UML Pattern* dialog.

Pattern Name: Abstract Factory

Filename: C:\Documents and Settings\John\Desktop\Patterns\GoF Patterns\Creation

Category: Basic Patterns Version: 1.0

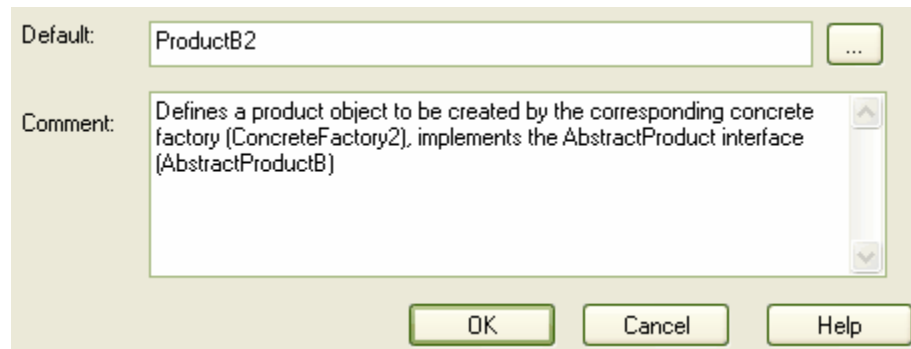
Notes: The Abstract Factory is provides an interface for the creation of related or dependant objects without creating any concrete classes. Also Known as a Kit

| Name | Type | Create | Merge | Instance | Type | Default | Comment |
|----------------------|-------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|------------------|-------------------------|
| Client | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Client | Uses only the interf... |
| ProductB1 | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ProductB1 | Defines a product ... |
| ProductB2 | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ProductB2 | Defines a product ... |
| AbstractProd... | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | AbstractProductB | Declares an interfa... |
| ProductA1 | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ProductA1 | Defines a product ... |
| ProductA2 | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ProductA2 | Defines a product ... |
| AbstractProd... | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | AbstractProductA | Declares an interfa... |
| ConcreteFact... | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ConcreteFactory2 | implements the op... |
| ConcreteFact... | Class | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ConcreteFactory1 | implements the op... |
| AbstractFactoryClass | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | AbstractFactory | Decalares an interf... |

OK Cancel Help

2. Set the *Pattern Name*.
3. Enter a *Filename* (.XML) to save the pattern as.
4. Select or enter a *Category* for the pattern to appear in under UML Patterns (required).
5. Enter the *Version* number and any additional *Notes*.
6. Select the appropriate actions for the elements that are contained in the pattern by checking the check boxes as appropriate. These actions are performed when the pattern is used (for more detail refer to the [Using Patterns](#) topic). The available actions are:
 - Create: Creates the pattern element directly without modification.
 - Merge: Merges the pattern element with an existing element, allowing the existing element to take on the role of the selected pattern element.
 - Instance: Creates the pattern element as an instance of an existing element. (this option is available if the pattern element supports this action).
 - Type: Creates the pattern element types as an existing element (this option is available if the pattern element supports this action).
7. To change the name of one of the elements, highlight the element and then click on the element to bring up the *Edit* dialog, from this dialog the user can change the name of the element as well as

adding comments detailing the elements purpose.



8. Press **OK** to save the pattern. Once saved you can load it into EA as a Pattern in the Resources tab.

6.7.2 Import a Pattern

Before using a previously [created pattern](#) file in a UML model it must first be imported into the current UML model which will then be available from the [Resource View](#) and optionally from the UML toolbox. To import a UML Pattern you have previously saved, follow the steps outlined below:

1. Select the **Resource View** tab on the Project Browser.
2. Right click on the **UML Pattern** node.
3. Select **Import UML Pattern** from the context menu
4. In the **File Find** dialog, locate the XML file to import.
5. Press **OK** to import the pattern.



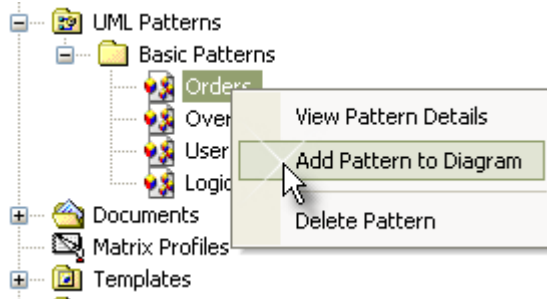
The imported pattern will be placed in the appropriate category as defined in the XML file. If the Category does not already exist under UML Patterns, a new one will be created. To download an example of the Gang of Four pattern for Enterprise Architect click [here](#).

6.7.3 Use a Pattern

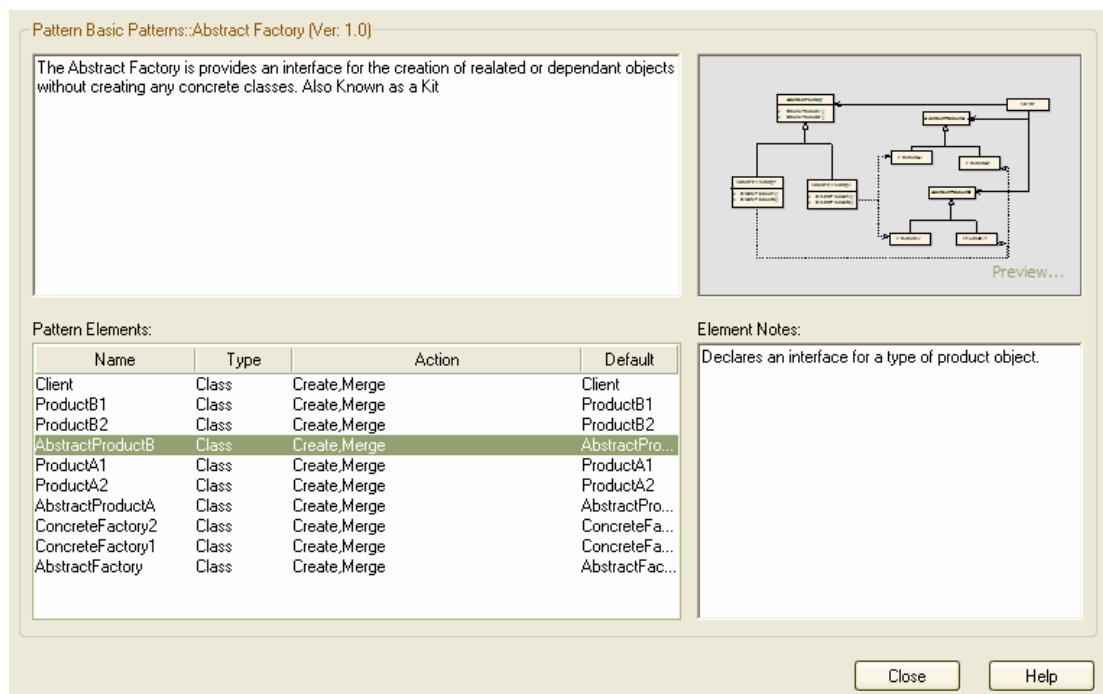
Using a pattern allows items that defined in the pattern to be used with the UML model. Using patterns allows for the rapid creation of template solutions for code structures have performed the same type of task in other situations.

To use a Pattern that you have [previously imported](#) into the model, follow the steps outlined below:

1. Open the diagram into which you want to add the UML pattern.
2. Select the *Resource View* tab on the Project Browser.
3. Open the *UML Pattern* folder and find the pattern you wish to add.
4. Right click on the pattern and select the *Add Pattern to Diagram* menu item or drag and drop the pattern from the Resource View onto the diagram or view the Pattern details by selecting the *View Pattern Details* menu item to view the pattern details in read only mode.



5. This will open the *Add Pattern* dialog (below).



6. The *Add Pattern* dialog allows users to perform several operations, these are detailed in the table below:

| Control | Description |
|---------|---|
| Preview | This pane allows the preview of the pattern, click on the Preview link to open a larger preview of the pattern. |

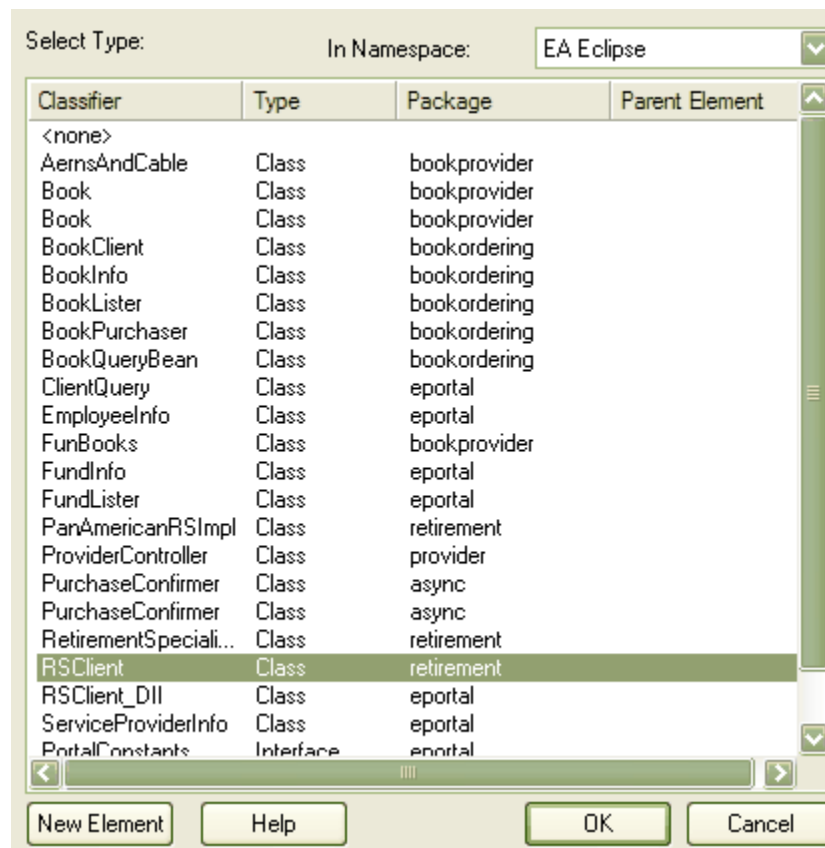
| | |
|------------------|--|
| Element Note | This pane is used to display the comments that describe the element in the pattern, highlight the element in the Pattern Elements pane to view the notes. |
| Pattern Elements | This pane provides access to the individual elements contained in the pattern, from here you can select the action for the individual element (Create , Merge, Instance or Type - as applicable for each element), Modify the name of the pattern element and choose the namespace for the merged element. |
| OK | Add the pattern to the diagram. |
| Cancel | Cancel the addition of the pattern to the diagram |
| Help | Opens the help file to display contextual help (this topic). |

- Once the appropriate selections have been made press the **OK** button to import the pattern into the model recreating the original diagram with new GUIDs.

Changing Pattern Element Name

The name of the pattern element may be changed by using the following instructions:

- From the **Add Pattern** dialog select the individual element in the Pattern Element pane.
- Click on the **...** button to bring up the **Edit** dialog, the specific method for changing the element name will be dependant upon the entry in the **Action** section of the **Pattern Elements** pane.
- If the Action selection is set to **Create** then in the **Edit** dialog the process used to rename the element is to delete the existing default name and replacing with a user defined name. The element name will now be changed in the **Add Pattern** dialog.
- If the Action for the element is set as **Merge**, in the **Edit** dialog press the **...** button to browse to an existing element classifier. This will bring up the **Set Element Classifier** dialog.
- From the **Set Element Classifier** dialog select an existing element classifier from the list of available classifiers or restrict the number of available choices by selecting the elements from a specific namespace by selecting a namespace from the **In Namespace** drop down menu. For more information regarding setting Element Classifiers refer to the [Using Classifiers](#) topic.



6.8 Requirements Management

Requirements Management

Gathering requirements is typically the first step in developing a solution, be it for the development of a software application or the detailing of a business process. Requirements are essentially "what the system needs to do". The requirements management built into EA can be used to define requirement elements, link requirements to model elements that implement that requirement, link requirements together into a hierarchy, report on requirements and move requirements into and out of model element requirements.

Typical Tasks

Typical tasks you might want to perform when using requirements with EA include:

- [Creating Requirements](#)
- [External Requirements](#)
- [Color Coding External Requirements](#)
- [Internal Requirements](#)
- [Move Internal Requirements to External Requirements](#)
- [Requirement Properties](#)
- [Composition](#)
- [Implementation](#)
- [Requirements's Hierarchy](#)
- [The Matrix](#)
- [Dependency Report](#)

6.8.1 Creating Requirements

Create Requirements at Diagram Level

To create requirements at the diagram level, follow the steps below:

1. Open the *Custom* group on the UML Toolbox.
2. Drag the requirement element onto the current diagram.
3. Enter *Name* and other details for the requirement.

EA creates a requirement in the current diagram and in the current package.

Create Requirements at Package Level

To create a requirement at the package level, follow the steps below:

1. Right click on a package to open the context menu.
2. Select *New Element* from the *Insert* menu. Alternatively, press *Ctrl+M*
3. In the *New Element dialog* select *Requirement* type.
4. Enter *Name* (or select *Auto*) and press *OK*.

EA creates a requirement in the current package.

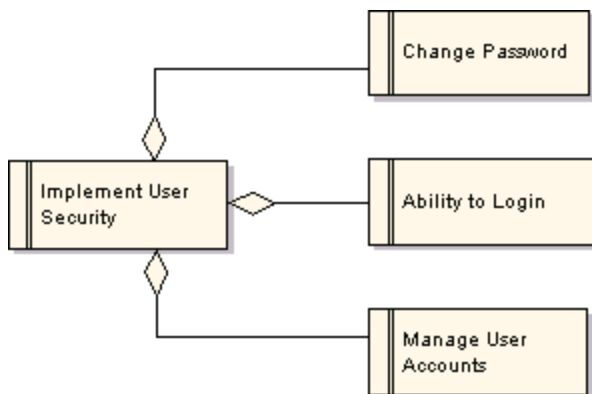
Tip: You may also move internal responsibilities of an element to external requirements - see the section on *Moving Internal requirements*

6.8.2 External Requirements

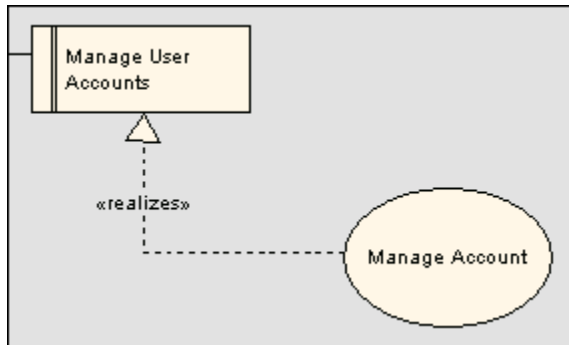
Separate Requirement elements can be manipulated at the diagram level. These correspond to the 'system level requirements' and may be linked using Realization type connectors to other model elements which take on the responsibility of implementing the requirement. Requirements at this level have their own properties and are reported on separately in the RTF documentation.

In this context, requirements may also form a hierarchy, as in the example below.

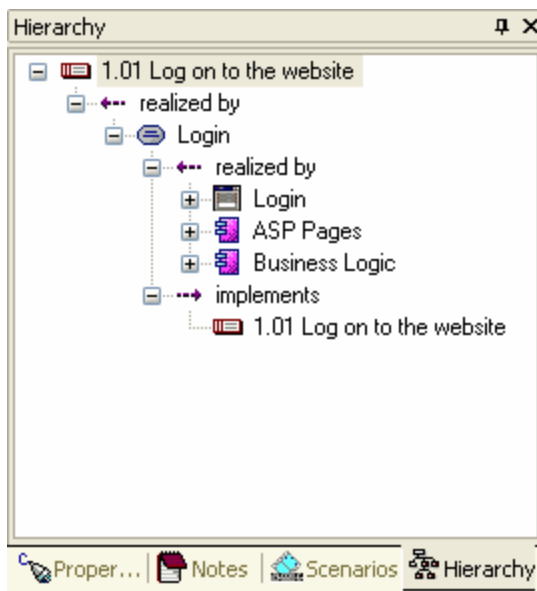
Tip: Using Aggregation, requirements can be linked to show construction of a complete requirements 'tree'.



Implementation is managed using Realization links as in the example below:



Once the links are established, the [Hierarchy window](#) will display the complete requirement implementation / composition details (see example below).



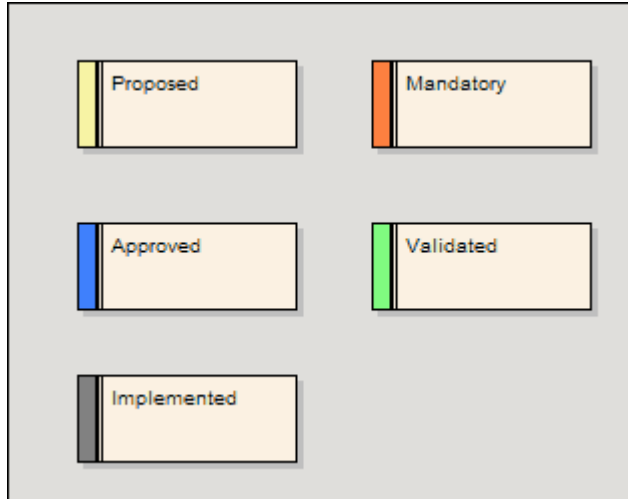
Tip: Use the Relationship Matrix to create and manage the relationships between the requirements - this is a convenient way of quickly building up complex relationships and hierarchies.

6.8.3 Color Coded External Requirements

External requirements may be color coded to enable quick visual cues indicating the status of an requirement. To enable color coded external requirements use the following instructions:

1. From the **Tools** menu select **Options** to bring up the **Local Options** dialog.
2. From the hierarchical tree select **Objects**. Ensure that the **Show status colors on diagrams** checkbox is marked.

3. This will enable the status of external requirements to be represented by color coding.
4. The color code requirements will use the following conventions, yellow for Proposed, blue for Approved, Green for validated, Orange for Mandatory and Black for Implemented.



6.8.4 Internal Requirements

Internal requirements in EA are class (or any element) Responsibilities.

In the example below an Internal responsibility to allow the user to login to the system has been defined for the Login use case. This is a *Responsibility* of the Use Case - some action it is responsible for carrying out - and it applies only to this Use Case.

The use case also has connections to external requirements - which are system functions that the Use Case will implement either in full or in part.

Requirement: 1. Login to System Type: Functional

Status: Approved Difficulty: Medium Priority: Medium Last Update: 1/11/2004

Buttons: Move External, New, Save, Delete

| Requirement | Type | External |
|---|------------|----------|
| 1. Login to System | Functional | |
| 1.01 | Functional | Yes |
| 2. Go to the Registered screen for non- ... | Functional | |
| 3. Encrypt user details over HTTPS | Functional | Yes |
| 4. Logout of system | Functional | |

Buttons: Apply, OK, Cancel, Help

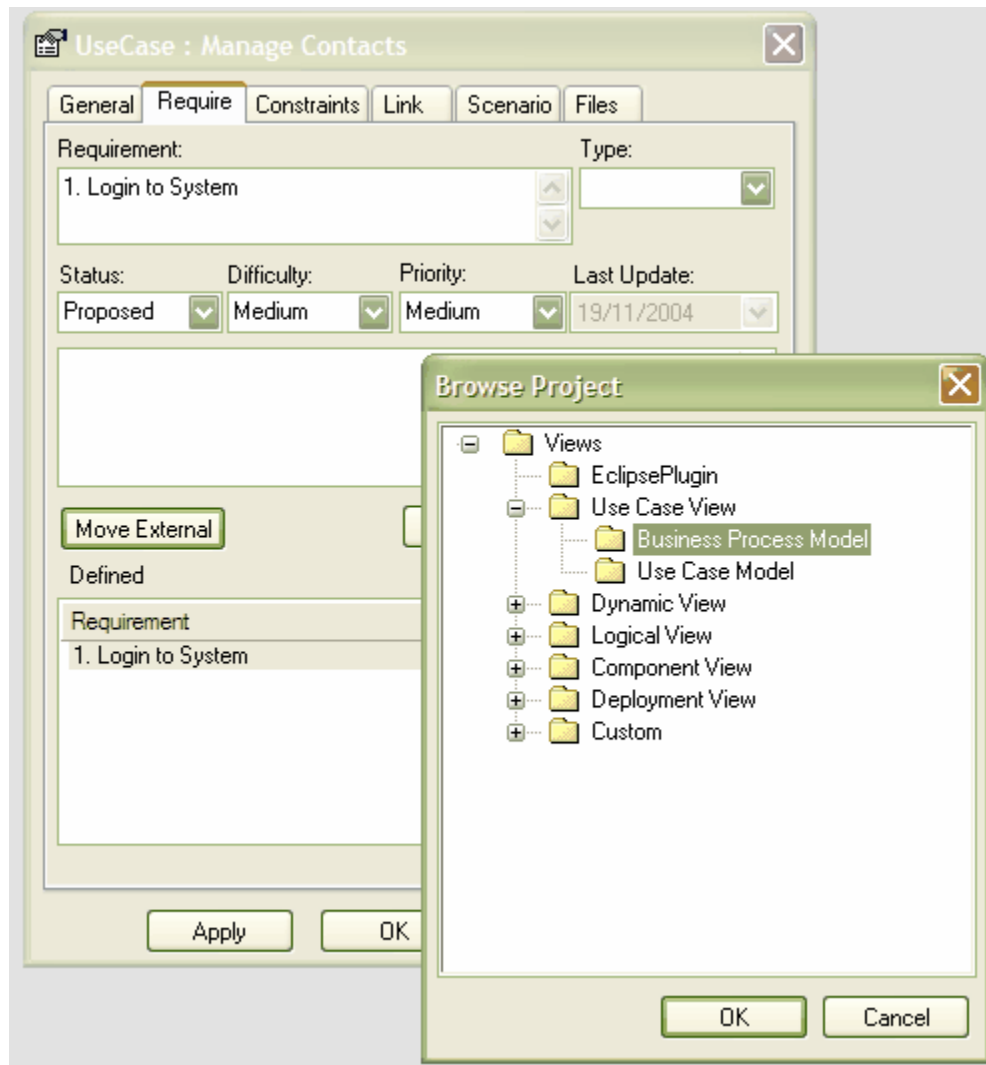
6.8.5 Move Internal Requirements to External Requirement

Elements in EA have internal requirements (what they must do or accomplish). These often overlap or duplicate more formal requirements that the system in general must meet. You can move internal requirements to external requirements in one go using the 'Move External' function.

Move an Internal Requirement to External Requirement

If you have defined an internal requirement for an element and want to move it out (where it will perhaps be implemented by multiple elements) then follow these steps:

1. Open the element properties by double clicking the element in a Diagram, or in the Project Browser.
2. Go to the *Require* tab.
3. Locate and highlight the requirement.
4. Press *Move External*.
5. Select the Package to place the new requirement in.
6. Press *OK*.

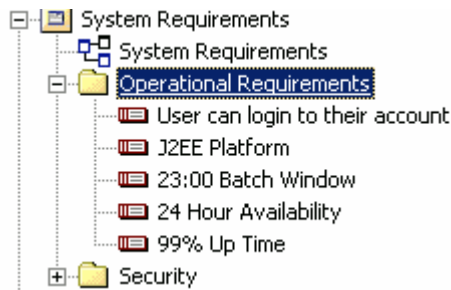


EA will create a new Requirement element in the target package and automatically generate a Realization link from the element to the requirement

| Requirement | Type | Extern |
|--|------------|--------|
| 1. Login to system | Functional | |
| 1.01 Log on to the website | Functional | Yes |
| 2. Go to register screen for non-registered user | Functional | |
| 3. Encrypt user details over HTTPS | Functional | Yes |
| 4. Logout of system | Functional | |

Notice the requirement is now marked external and the form grayed out. You can double click on the requirement to edit its details.

Also notice that a requirement has been created in the target package.



6.8.6 Requirement Properties

If you double click on a requirement in a diagram or right-click on a requirement in the Project Browser and select *Properties*, you will be able to edit the main properties for the relevant element. Requirement properties are a little different to normal properties - they are mainly focused on the status of the requirement, who owns it and when it was created and/or updated.

Enter a short description to describe the requirement in a sentence - then enter the full requirement details in the notes section at the bottom of the dialog. Set the status, difficulty, priority and other parameters as desired.

When you have completed filling in details for the requirement - press the *OK* button.

6.8.7 Composition

Requirements that are linked by aggregation relationships form a composition hierarchy. High level requirements may be composed of lower level requirements, which in turn are made up of finer and more specialized requirements. This hierarchical structure helps manage the complexity of large systems with thousands of requirements and many elements being employed to implement the requirements.

6.8.8 Implementation

Requirements are implemented by model elements - such as Use Cases, Classes, Interfaces, Components, etc. You may specify this relationship in EA using the Realization link. A model element is marked as 'Realizing' a requirement. Once this link exists, EA will display the requirement in the element Require tab, in the requirement hierarchy tab and in the dependency and implementation reports, as well as the standard RTF output.

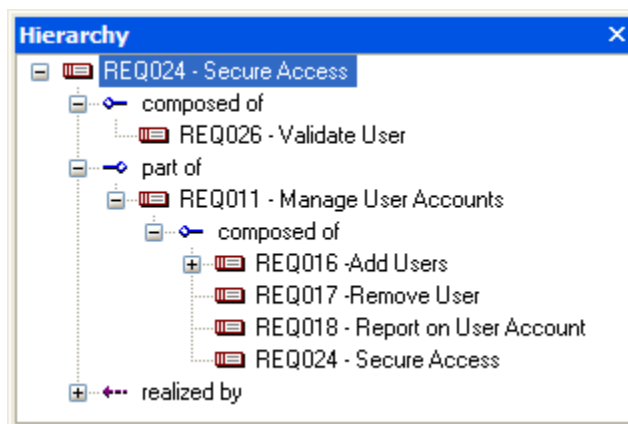
A quick method of generating a Realization link is to drag a Requirement element from the Project Browser over an element in a diagram which is to be the implementing element. EA will interpret this as a request to create the realization link and do so automatically. To confirm this, perform the action, and then go to the Require tab page of the target element - there should now be an external relationship to the requirement that was dragged over the target.

6.8.9 Requirements Hierarchy

Requirements may be linked to other elements or other requirements to create a relationship hierarchy.

In general, the aggregation relationship can be used to good effect to show how high level relationships are composed of smaller requirements. This hierarchy is useful in managing the composition of your model and the dependencies between elements and requirements.

The example below shows the hierarchy structure for AccountItem. See the [Hierarchy](#) topic for further information.



6.8.10 The Matrix

The relationship matrix is a spreadsheet like display of relationships between model elements. You select a source package and a target package, the relationship type and direction, and Enterprise Architect will highlight all the relationships between source and target elements by highlighting a grid square.

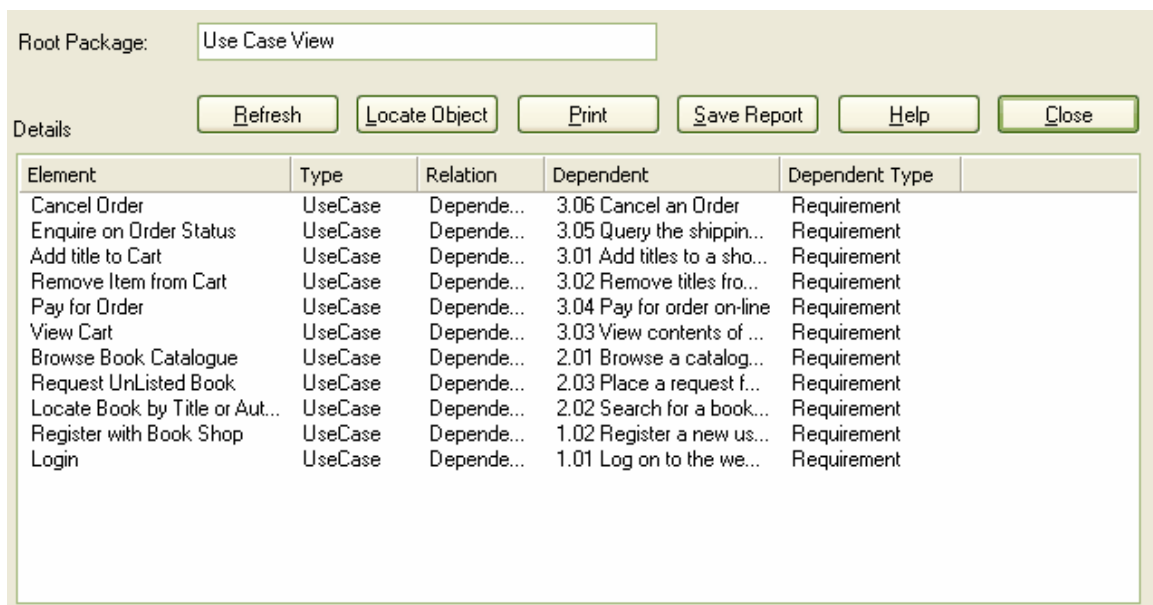
The matrix is a convenient method of visualizing relationships quickly and definitively. It is also possible to delete and create relationships in the Matrix view - another quick way to set up complex sets of element relationships with a minimum of effort.

6.8.11 Dependency Report

To run a dependency report follow these steps:

1. In the Project Browser, select the package you wish to report on (the report includes all sub-folders as well)
2. From the *Project | Documentation* submenu, select *Dependency Details...*

The *Dependencies* dialog displays a list of all elements that implement other elements in the provided list, together with the elements that are dependent. You may print out the results if required.



| Control | Description |
|--------------------|--|
| Root Package | The root package. All elements and packages under this will appear in the report. |
| Locate Object | This button lets you locate the element selected in the report list in the Project Browser. |
| Refresh | Run the report again. |
| Dependency Details | List of dependency details - lists elements in the current hierarchy and elements that implement them. |
| Print | Print the list. |

6.9 Element Relationship Matrix

The relationship matrix allows you to study the relationships between model elements within packages. It also allows you to create, modify and delete relationships between elements with a single mouse click.

Tip: The relationship matrix provides a quick overview of relationships between elements in packages.

| Source: Business Model | Type: <All> | Link Type: Aggregation | Profile: | | | | | | | | | | | | | | | |
|--------------------------------|-------------------|-----------------------------|------------------------|------------------------------|----------------------------|-----------------------|--------------------------------|------------------------------|--------------------------------|------------------------|----------------------------|------------------------|----------------------|-----------------|---------------|-------------------|-----------------------|---------------|
| Target: Business Model | Type: <All> | Direction: Source -> Target | Refresh | | | | | | | | | | | | | | | |
| | Accounts::Account | Accounts::Message Folder | Accounts::Staff Member | Address Book::Address Book C | Address Book::Address Book | Address Book::Contact | Business Model::Business Pr... | Business Model::Domain Model | Business Process Model::Class1 | Domain Model::Accounts | Domain Model::Address Book | Domain Model::Messages | Messages::Attachment | Messages::Email | Messages::Fax | Messages::Message | Messages::Postal Mail | Messages::SMS |
| Accounts::Account | | | | | | | | | | | | | | | | | | |
| Accounts::Message Folder | X | | | | | | | | | | | | | | | | | |
| Accounts::Staff Member | | | | | | | | | | | | | | | | | | |
| Address Book::Address Book... | | | | X | X | | | | | | | | | | | | | |
| Address Book::Address Book | X | | | | | | | | | | | | | | | | | |
| Address Book::Contact | | | | X | X | | | | | | | | | | | | | |
| Business Model::Business Pr... | | | | | | | | | | | | | | | | | | |
| Business Model::Domain Model | | | | | | | | | | | | | | | | | | |
| Business Process Model::Class1 | | | | | | | | | | | | | | | | | | |
| Domain Model::Accounts | | | | | | | | | | | | | | | | | | |
| Domain Model::Address Book | | | | | | | | | | | | | | | | | | |
| Domain Model::Messages | | | | | | | | | | | | | | | | | | |
| Messages::Attachment | | | | | | | | | | | | | X | | | | | |
| Messages::Email | | | | | | | | | | | | | | | | | | |
| Messages::Fax | | | | | | | | | | | | | | | | | | |
| Messages::Message | | X | | | | | | | | | | | | | | | | |
| Messages::Postal Mail | | | | | | | | | | | | | | | | | | |
| Messages::SMS | | | | | | | | | | | | | | | | | | |

6.9.1 Open the Relationship Matrix

To open the Relationship Matrix you can:

- Select **Relationship Matrix** from the **View** menu, or
- Right click on any package in the Project Browser, select **Documentation | Open in Relationship Matrix**, and select **As Source** or **As Target**.

Once the Relationship Matrix opens you can:

- [Set the source and target packages](#)
- [Select which element type to show](#)
- [Select link type and direction to show](#)

The matrix will refresh itself after every change you make to the input parameters.

Tip: The matrix will include -ALL- child elements in a hierarchy - sometimes in a large model this can be a lot of elements- possibly too many to be useful. Take care in selecting the source and target package.

6.9.2 Setting Source and Target Package

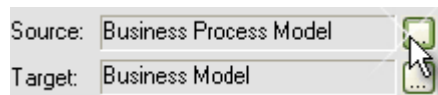
You need to set both the Source and Target packages for the Matrix before relationships can be displayed.

Tip: Set the source and target packages *AFTER* setting the link and element types/details - as EA refreshes the content after each change, this is usually faster.

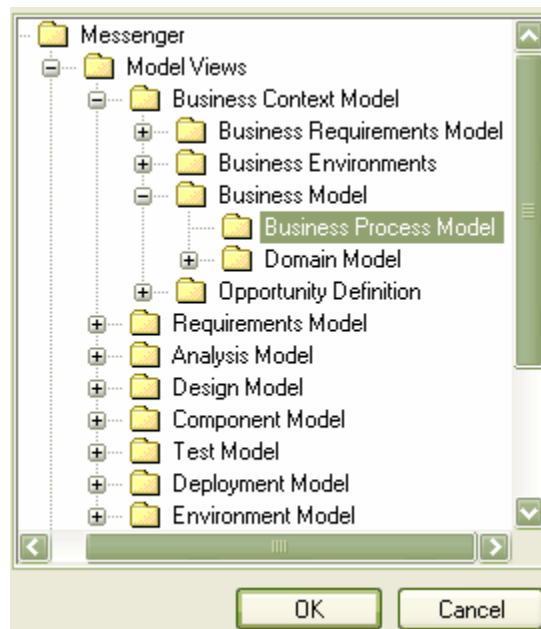
Set the Source or Target Package

To set the source or target package, follow the steps below:

1. Press the **Browse [...]** button at the end of the source or target item.



2. A Project Browser window will pop up - select the required package and press **OK**.



6.9.3 Setting Element Type

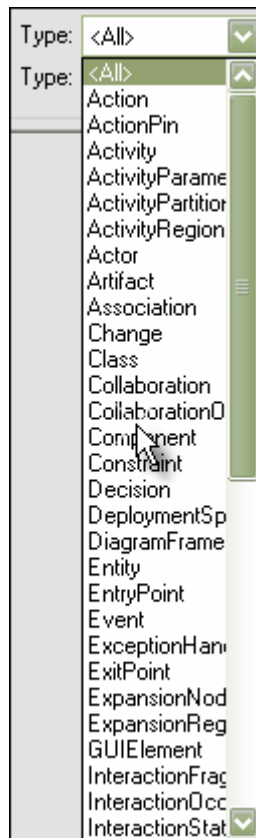
The Relationship Matrix can show *all* element types, or you can specify which type to show.

Set Element Type

To set element type, follow the steps below:

1. Left click on the drop down list of link types for the Source or Target package.
2. Find the required link and select.

EA will refresh the matrix content.



6.9.4 Setting the Link Type and Direction

The matrix requires that you set the link type to report on -and- the link direction.

Set Link Type

To set link type, follow the steps below:

1. Left click on the drop down list of link types.
2. Find the required link and select the appropriate link type, EA will then refresh the matrix content.



Set Link Direction

To set link direction, follow the steps below:

1. Left click on the drop down list of link directions.
2. Select the required type.

EA will refresh the matrix content.

6.9.5 Matrix Options

You can set some local settings for how the Matrix works from the *Matrix Options* menu:

- *Allow Link Creation* - select to allow access to the right-click link creation behavior
- *Open Maximized* - select to ensure Matrix opens in full screen mode
- *Include Source Children* - select to recursively include child packages and contents under the Source
- *Include Target Children* - select to recursively include child packages and contents under the Target
- *Sort Axes* - when selected, package elements appear in alphabetical order
- *Show Package Names* - hide or show package names in the Matrix, this is useful for shortening the displayed texts, especially in circumstances where packages have long names.

To access the Matrix Options menu right click on the Matrix Menu background to access the Matrix Menu then mouse over the *Options* menu item.

6.9.6 Modifying Relationships in Matrix

Relationships may be modified or deleted, or new relationships may be created, directly from the Relationship Matrix.

To Modify or Delete Relationships

1. Right click on a highlighted relationship to open the context menu (shown below).
2. Select from the following options:
 - *View relationship...* - opens the Property dialog for the selected relationship
 - *Source element properties...* - opens the Property dialog for the source element
 - *Target element properties...* - opens the Property dialog for the target element
 - *Delete relationship*
3. If you have selected *Delete relationship*, you will be prompted to confirm this action

-OR-

If you have selected one of the other options, modify any properties you need to, and Press *OK* to save changes

| | Activity Diagrams::Add Water to Coffee | Activity Diagrams::Brew Coffee | Activity Diagrams::Drink Beverage | Activity Diagrams::Find Beverage | Activity Diagrams::Get Can of Coffee | Activity Diagrams::Get Cups | Activity Diagrams::Pour Coffee | Activity Diagrams::Put Coffee in Filter | Activity Diagrams::Put Filter in Machine | Activity Diagrams::Some Other Beverage | Activity Diagrams::Test |
|--|--|--------------------------------|-----------------------------------|----------------------------------|--------------------------------------|-----------------------------|--------------------------------|---|--|--|-------------------------|
| Activity Diagrams::Add Water to Coffee | | | | | | | | | | | |
| Activity Diagrams::Brew Coffee | | | | | | | | | | | |
| Activity Diagrams::Drink Beverage | | X | | | | | | | | | |
| Activity Diagrams::Find Beverage | | | | | | | | | | | |
| Activity Diagrams::Get Can of Coffee | | | | X | Y | | | | | | |
| Activity Diagrams::Get Cups | | | | | | | | | | | |
| Activity Diagrams::Pour Coffee | X | | | X | X | | | | | | |
| Activity Diagrams::Put Coffee in Filter | | | | | | | | | | | |
| Activity Diagrams::Put Filter in Machine | | | | | | | | | | | |
| Activity Diagrams::Some Other Beverage | | | | | | | | | | | |
| Activity Diagrams::Test | | | | | | | | | | | |

View relationship...
 Source element properties...
 Target element properties...
 Delete relationship

To Create a New Relationship

1. Right click on an empty space to open the context menu (shown below).
2. Select *Create new relationship...* to create a new connector between two elements

| | Activity Diagrams::Add Water to | Activity Diagrams::Brew Coffee | Activity Diagrams::Drink Beverage | Activity Diagrams::Find Beverage | Activity Diagrams::Get Can of C | Activity Diagrams::Get Cups | Activity Diagrams::Pour Coffee | Activity Diagrams::Put Coffee in | Activity Diagrams::Put Filter in M | Activity Diagrams::Some Other | Activity Diagrams::Test |
|------------------------------------|---------------------------------|--------------------------------|-----------------------------------|----------------------------------|---------------------------------|-----------------------------|--------------------------------|----------------------------------|------------------------------------|-------------------------------|-------------------------|
| Activity Diagrams::Add Wate... | | | | | | | | | | | |
| Activity Diagrams::Brew Coffee | | | | | | | | | | | |
| Activity Diagrams::Drink Bev... | | X | | | | | | | | | |
| Activity Diagrams::Find Beve... | | | | | | | | | | | |
| Activity Diagrams::Get Can o... | | | | X | X | | | | | | |
| Activity Diagrams::Get Cups | | | | | | | | | | | |
| Activity Diagrams::Pour Coffee | X | | | X | X | | | | | | |
| Activity Diagrams::Put Coffe... | | | | | | | | | | | |
| Activity Diagrams::Put Filter i... | | | | | | | | | | | |
| Activity Diagrams::Some Oth... | | | | | | | | | | | |
| Activity Diagrams::Test | | | | | | | | | | | |

Tip: Use the Matrix relationship management features to quickly create and manage relationships like Realization and Aggregation between Requirements and implementation elements (such as Use Cases)

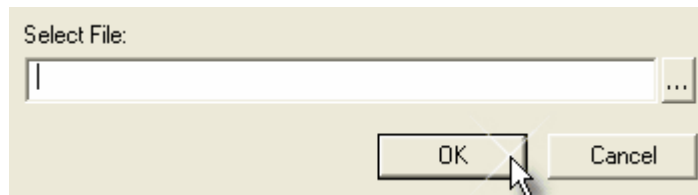
6.9.7 Exporting to CSV

The contents of the relationship matrix may be exported to a CSV file. This provides a convenient mechanism for moving the matrix data to a spreadsheet environment such as Microsoft Excel.

Export to CSV

To export to CSV, follow the steps below:

1. To access the Matrix sub-menu *right click* on the Matrix Menu background to access the Matrix Menu then select mouse over the *Matrix* menu item.
2. Select the menu option *Export to CSV*.
3. Enter a Filename to export to.



4. Press **OK** to export the data.

6.9.8 Printing the Matrix

You can print a WYSIWYG representation of the matrix using the current printer settings. To access the Matrix sub-menu right click on the Matrix Menu background to access the Matrix Menu then mouse over the Matrix menu item then select the **Print** or **Print Preview** Options.

- To print the matrix select **Matrix | Print**
- To preview prior to printing, select **Matrix | Print Preview**

Source: Business Model Type: <All> Link Type: Aggregation Profile: [v]
 Target: Business Model Type: <All> Direction: Source -> Target Refresh

| | Accounts::Account | Accounts::Message Folder | Accounts::Staff Member | Address Book::Address Book C | Address Book::Address Book | Address Book::Contact | Business Model::Business Proc | Business Model::Domain Model | Business Process Model::Class | Domain Model::Accounts | Domain Model | Domain Model | Domain Model | Messages::Att | M | M | M | M | M |
|---------------------------------|-------------------|--------------------------|------------------------|------------------------------|----------------------------|-----------------------|-------------------------------|------------------------------|-------------------------------|------------------------|--------------|--------------|--------------|---------------|---|---|---|---|---|
| Accounts::Account | | | | | | | | | | | | | | | | | | | |
| Accounts::Message Folder | X | | | | | | | | | | | | | | | | | | |
| Accounts::Staff Member | | | | | | | | | | | | | | | | | | | |
| Address Book::Address Book... | | | | X | X | | | | | | | | | | | | | | |
| Address Book::Address Book | X | | | | | | | | | | | | | | | | | | |
| Address Book::Contact | | | | X | X | | | | | | | | | | | | | | |
| Business Model::Business Pr... | | | | | | | | | | | | | | | | | | | |
| Business Model::Domain Model | | | | | | | | | | | | | | | | | | | |
| Business Process Model::Class 1 | | | | | | | | | | | | | | | | | | | |
| Domain Model::Accounts | | | | | | | | | | | | | | | | | | | |
| Domain Model::Address Book | | | | | | | | | | | | | | | | | | | |
| Domain Model::Messages | | | | | | | | | | | | | | | | | | | |
| Messages::Attachment | | | | | | | | | | | | | | | | | X | | |
| Messages::Email | | | | | | | | | | | | | | | | | | | |
| Messages::Fax | | | | | | | | | | | | | | | | | | | |
| Messages::Message | | X | | | | | | | | | | | | | | | | | |
| Messages::Postal Mail | | | | | | | | | | | | | | | | | | | |
| Messages::SMS | | | | | | | | | | | | | | | | | | | |

Matrix Menu: Matrix, Options, Profiles, Help, Print, Print Preview, Export to CSV...

6.9.9 Profiles

You can save a certain Matrix configuration as a named profile for later recall. To do this, follow the steps below:

1. Set up the Matrix the way you want - with source and target, element types and relationship types.
2. To access the *Profiles* sub-menu right click on the Matrix Menu background to access the Matrix Menu, then mouse over the *Profiles* menu item then select the *Save as New Profile* option.
3. Enter a *Name* (a limit of 12 characters applies) and press *OK*.
4. Once you have created one or more Profiles, you can select them from the drop list on the matrix screen.

6.10 Business Modeling

Modeling the Business Process

Modeling the business process is an essential part of any software development process. It allows the analyst to capture the broad outline and procedures that govern what it is a business does. This model provides an overview of where the proposed software system being considered will fit into the organisational structure and daily activities. It may also provide the justification for building the system by capturing the current manual and automated procedures that will be rolled up into a new system, and the associated cost benefit.

As an early model of business activity, it allows the analyst to capture the significant events, inputs, resources and outputs associated with business process. By connecting later design elements (such as Use Cases) back to the business process model through implementation links, it is possible to build up a fully traceable model from the broad process outlines to the functional requirements and eventually to the software artefacts actually being constructed.

As the Business Process Model typically has a broader and more inclusive range than just the software system being considered, it also allows the analyst to clearly map what is in the scope of the proposed system and what will be implemented in other ways (eg. a manual process).

Process Modeling Notation

A business process model typically defines the following elements:

- The Goal or reason for the process;
- Specific inputs;
- Specific outputs;
- Resources consumed;
- Activities that are performed in some order; and
- Events that drive the process.

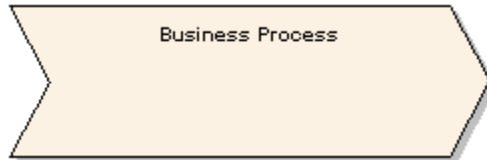
The business process:

- May affect more than one organisational unit.
- Have a horizontal organisational impact;
- Creates value of some kind for the customer. Customers may be internal or external

The Business Process

A business process is a collection of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how the work is done within an organisation, in contrast to a

product's focus on what. A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly defined inputs and outputs: a structure for action. The notation used to depict a business process is illustrated below.

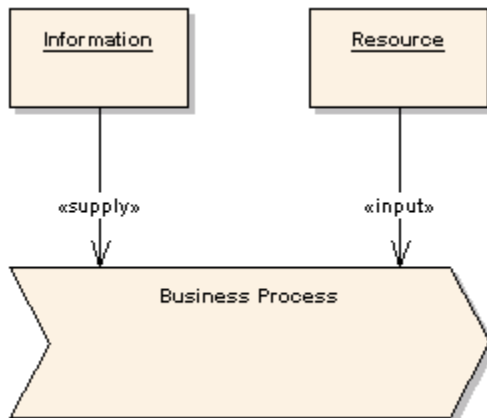


The process notation implies a flow of activities from left to right. Typically an event element is placed to the left of the process and the output to the right. To specifically notate the internal activities, UML activity elements may be placed inside the process element.

Inputs, Resources and Information

Business processes use information to tailor or complete their activities. Information, unlike resources, is not consumed in the process - rather it is used as part of the transformation process. Information may come from external sources, from customers, from internal organisational units and may even be the product of other processes. A resource is an input to a business process, and, unlike information, is typically consumed during the processing. For example, as each daily train service is run and actuals recorded, the service resource is 'used up' as far as the process of recording actual train times is concerned.

The notation to illustrate information and resources is shown below.

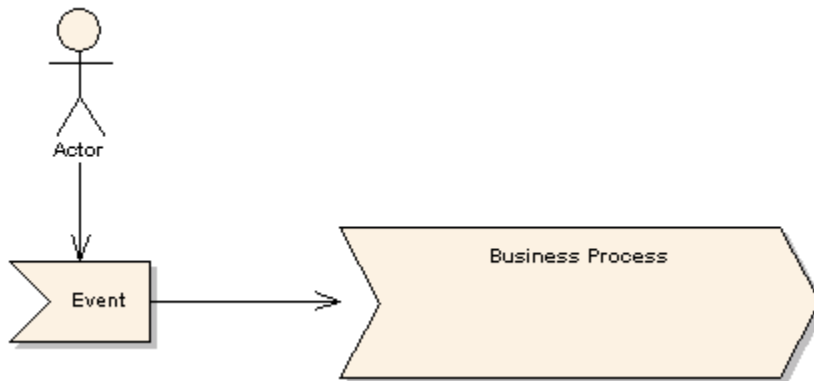


A supply link indicates that the information or object linked to the process is not used up in the processing phase. For example, order templates may be used over and over to provide new orders of a certain style - the templates are not altered or exhausted as part of this activity.

An input link indicates that the attached object or resource is consumed in the processing procedure. As an example, as customer orders are processed they are completed and signed off, and typically are used only once per unique resource (order).

Events

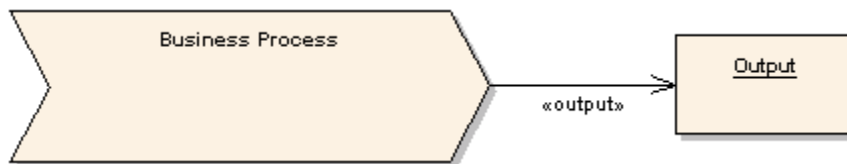
An event is the receipt of some object, a time or date reached, a notification or some other trigger that initiates the business process. The event may be consumed and transformed (for example a customer order) or simply act as a catalyst (e.g. nightly batch job).



Outputs

A business process will typically produce one or more outputs of value to the business, either for internal use or to satisfy external requirements. An output may be a physical object (such as a report or invoice), a transformation of raw resources into a new arrangement (a daily schedule or roster) or an overall business result such as completing a customer order.

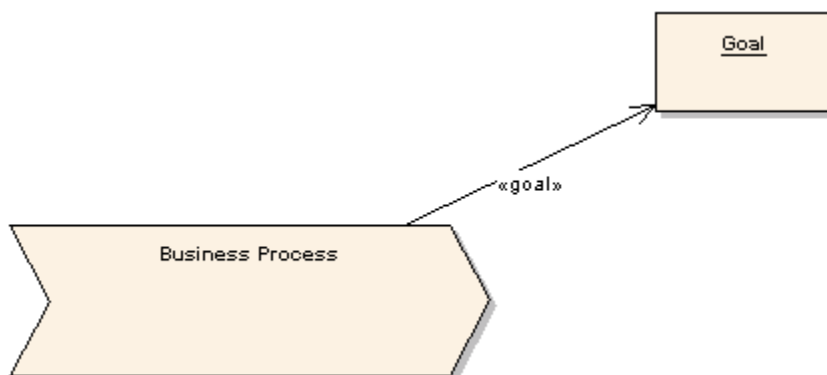
An output of one business process may feed into another process, either as a requested item or a trigger to initiate new activities.



An output link indicates that the business process produces some object (either physical or logical) that is of value to the organisation, either as an externally visible item or as an internal product (possibly feeding another process).

Goals

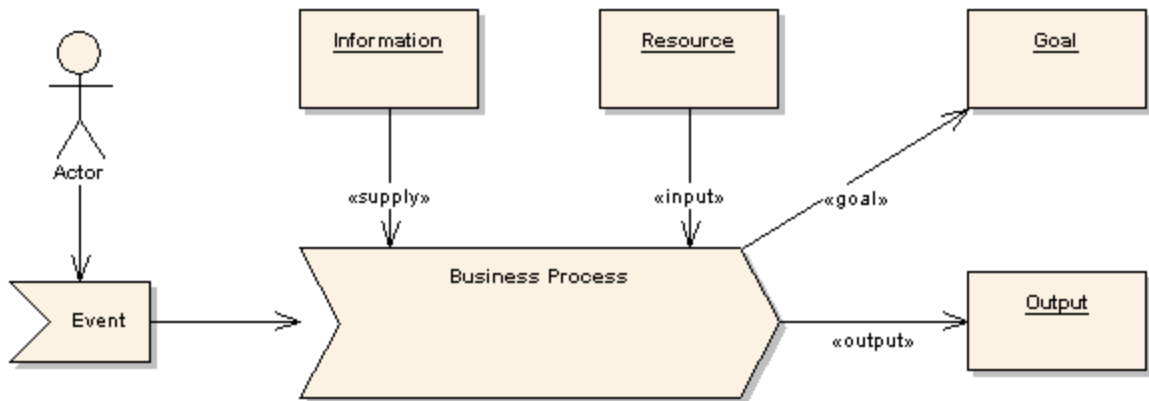
A business process has some well defined goal. This is the reason the organization does this work, and should be defined in terms of the benefits this process has for the organization as a whole and in satisfying the business needs.



A goal link indicates the attached object to the business process describes the goal of the process. A goal is the business justification for performing the activity.

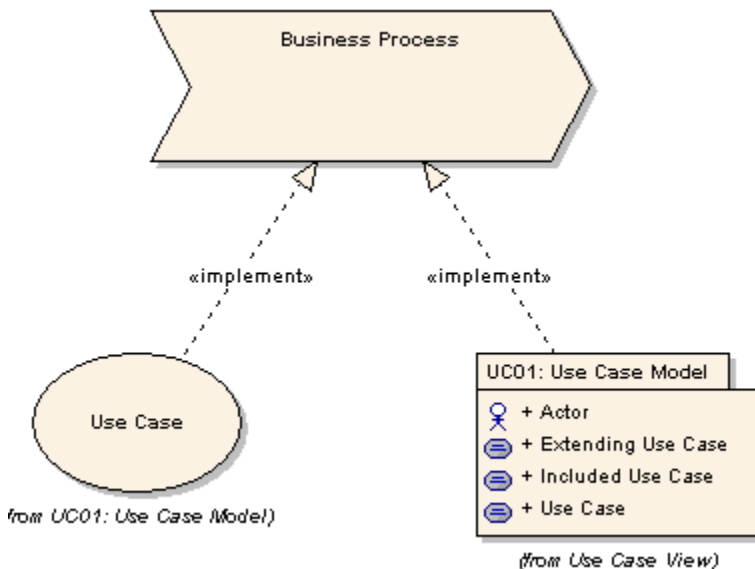
Putting it together

The diagram below illustrates how the various model elements may be grouped together to produce a coherent picture of a named business process. Included are the inputs, outputs, events, goals and other resources which are of significance.



Traceability

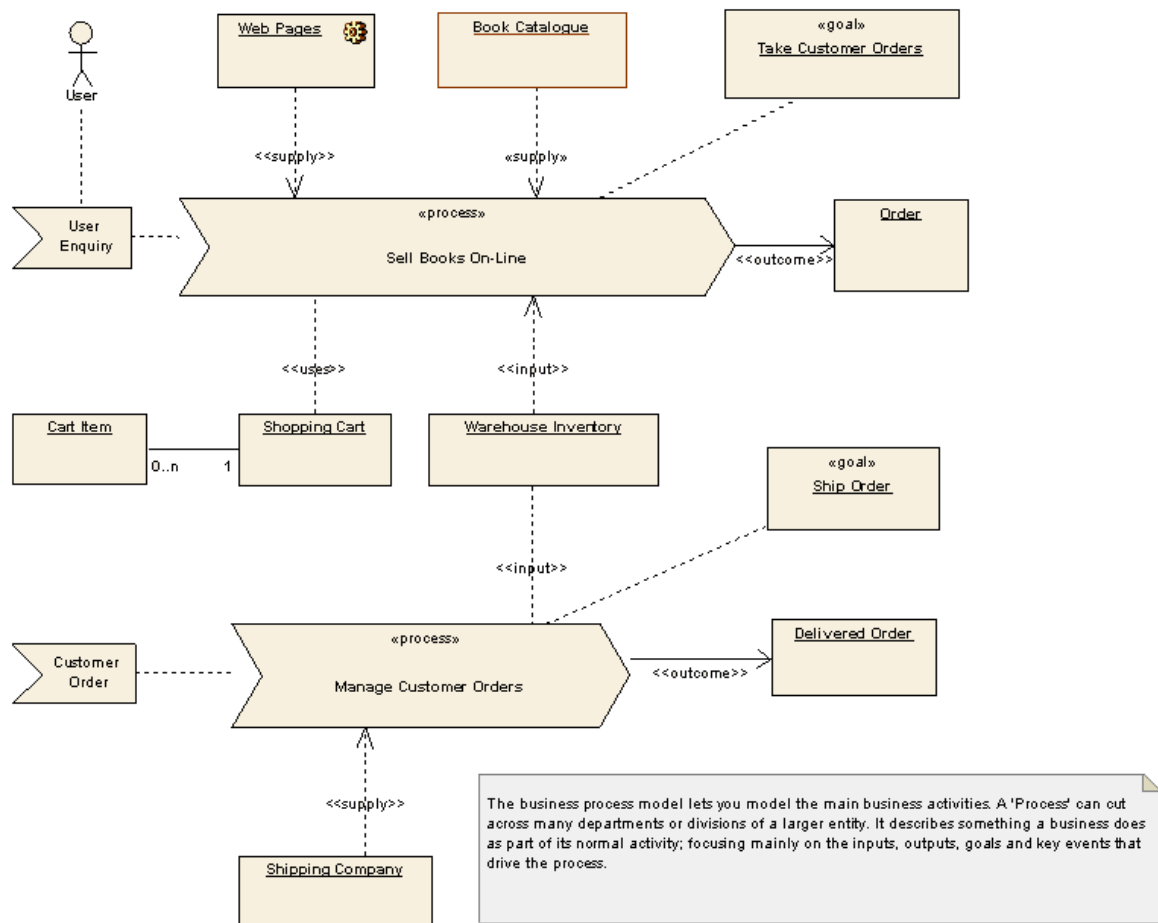
Traceability defines the way a given business process will be implemented in the proposed system. In an implementation diagram, use cases, packages and other model artefacts may be linked back to the business process with <<implementation>> links to signify the dependent relationship. The example below illustrates how a Business Process is implemented by a Use Case and a package. As the model develops and functional software components are built and linked to Use Cases, the business justification for each element can be derived from this model.



Note that this model also implies what is NOT being delivered. The Business Process will typically include a wide range of manual and automated procedures. This model illustrates exactly what functionality (Use Cases) will be built to service a particular business process: any missing functionality must come from other (manual or automated) systems and procedures.

An Example

The example below is an example of the kind of model that may be built up to represent a business process. In this model, the goal of the business process is to take customer orders and to ship those orders out. A user starts the process with an inquiry, which leads to the involvement of the Book Catalogue, Shopping Cart, On-line pages and warehouse inventory. The output of significance to the business is a customer order.



The second half of the process model is to respond to a customer order and ship the required items. The second process involves the warehouse inventory, shipping company and completes when an order is delivered to the customer.

See Also

- [Business Modeling Stereotypes](#)

Part

7

7 MDA Transforms

MDA transforms provide a fully configurable way of converting model elements and model fragments from one domain to another. This will typically involve converting Platform-Independent Model (PIM) elements to Platform-Specific Model (PSM) elements. A single element from the PIM may be responsible for creating multiple PSM elements across multiple domains.

Transformations are a huge productivity boost, and reduce the need to manually implement stock classes and elements for a particular implementation domain - for example database tables generated from persistent PIM classes. EA includes some basic built-in transformations, such as PIM to Data Model, PIM to C#, PIM to Java and PIM to XSD. Sparx Systems will make additional transformations available over time - either as built in transformations, or as downloadable modules from the Sparx Systems website.

For a further productivity boost, EA can automatically generate code for your transformed classes that target code languages. To enable this, check Generate Code on result in the [Model Transformation](#) dialog. With this option checked, the first time you transform to that class EA will allow you to select a filename to generate to. Subsequent transformations will automatically generate any class with a filename set.

A Transformation is defined using the same simple code generation template language that has been in EA for some years now, and involves no more than writing a template to create a simple intermediary source file. EA reads the source file and binds that to the new PSM.

EA also creates internal bindings between each PSM created and the original PIM. This is essential, as it allows you to forward synchronize from the PIM to the PSM many times, adding or deleting features as you go. So for example, adding a new attribute to a PIM class can be forward synchronized to a new column in the Data Model.

EA will not delete or overwrite any element features that were not originally generated by the transform. So you can add new methods to your elements, and EA will leave them alone during the forward generation process.

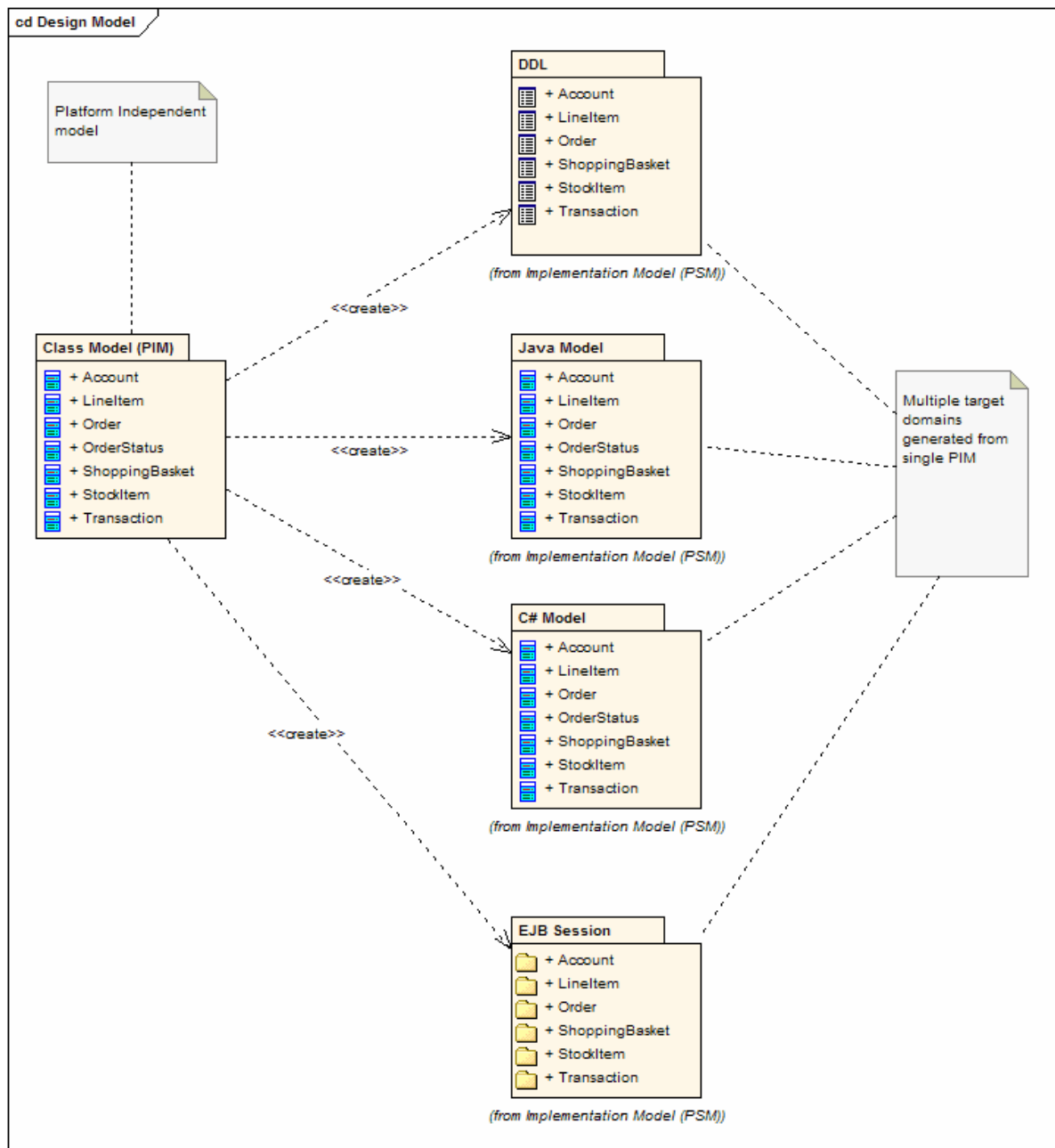
Transformations that are currently built in include:

- **DDL.** Transforms platform-independent class elements to platform-specific table elements.
- **EJB Entity.** Transforms platform-independent class elements to packages containing the class and interface elements that comprise an EJB Entity Bean.
- **EJB Session.** Transforms platform-independent class elements to packages containing the class and interface elements that comprise an EJB Session Bean.
- **Java.** Transforms platform-independent elements to Java language elements.
- **JUnit.** Converts a java model to a model where test methods are created for each public method of any original class.
- **C#.** Converts a PIM to a standard C# implementation set.
- **NUnit.** Converts a .Net language specific model to a model where test methods are created for each public method of any original class.
- **WSDL.** Converts a simple representation of a WSDL interface, into the elements required to generate that interface.
- **XSD.** Transforms platform-independent elements to XSD elements.

Transformations are described in the following sections:

- [Transforming Elements](#)
- [Importing Transforms](#)
- [Transformation Templates](#)
- [Built-in Transformations](#)
- [Writing Transformations](#)
- [Chaining Transformations](#)

The following diagrams highlight how Transforms work and how they can significantly boost your productivity:



7.1 Transforming Elements

There are two modes for initiating a Model Transformation, each of which can be started in two ways.

- To transform selected elements on a diagram, either:
 - From the main menu, select **Project | Model Transformations | Transform Selected Elements**, or
 - From the context menu of the classes on the diagram, select **Transform Selected Element(s)**.
- To transform elements in the package currently selected in the Project View, either:
 - From the main menu, select **Project | Model Transformations | Transform Current Package**, or

- From the context menu of the package in the Project View, select "**Transform Current Package**".

This will display the Model Transformation dialog.

When the dialog is opened, all elements will be selected and all transformations previously performed from any of these classes will be checked.

| Control | Description |
|-----------------------------------|---|
| Elements | Selects the elements that are to be included in the transformation. |
| All | Selects all of the elements from the list to be included in the transformation. |
| None | Deselects all of the elements from the list. |
| Transformations | Allows you to select which transformations to perform and the package each of them should be transformed to. |
| Select Package [...] | Use the [...] button to select the package in which the transformed elements will be created. |
| Generate Code on result | Specifies whether or not to automatically generate code from the target classes. |
| Perform Transformations on result | Specifies if transformations previously done on target classes should be automatically executed. See Chaining Transformations for more information. |
| Intermediary File Path | Specifies the filename of the intermediary file (if any). |
| Write Always | Specifies whether an intermediary file should be written to disk. |
| Write Now | Generates the intermediary file but doesn't perform the transform. |
| Do Transform | Executes the transform command. |
| Close | Exits the Model Transformation dialog. |
| Help | Opens the help file on this page. |

7.1.1 Chaining Transformations

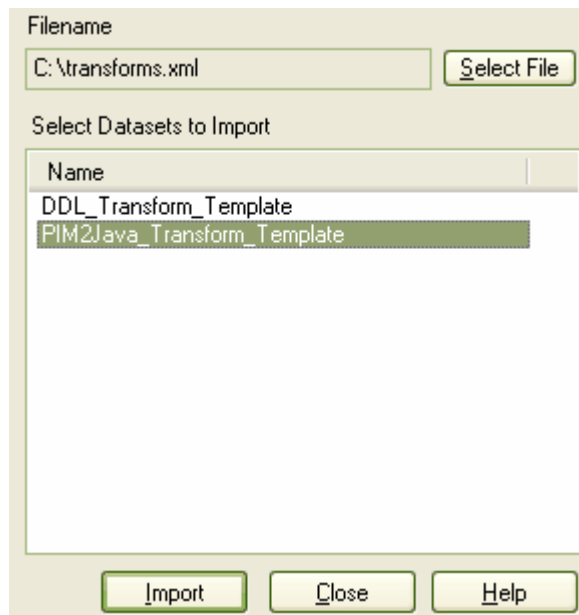
Chaining transformations provides an extra degree of flexibility and power to transformations. For example, you may have a situation where two transformations have a common element. This can then be separated into one transformation, and then the original transformations can be transformed from the common point. The separated transform could even produce a useful model itself.

EA provides for chaining transformations, by allowing transformations that have already been performed on target classes to be performed automatically next time that class is transformed to. To enable this check the box labelled *Perform Transformations on result* in the *Model Transformation Dialog*.

7.2 Importing Transforms

Transform templates can be transferred between models using the Import/Export Reference Data mechanism. To import a Transform Template:

1. From the **Tools** menu click **Import Reference Data** to display the Import Reference Data dialog.
2. Click **Select File** and browse to an .XML file containing the required Transform Template.
3. Select the name of one or more template dataset and click **Import**.

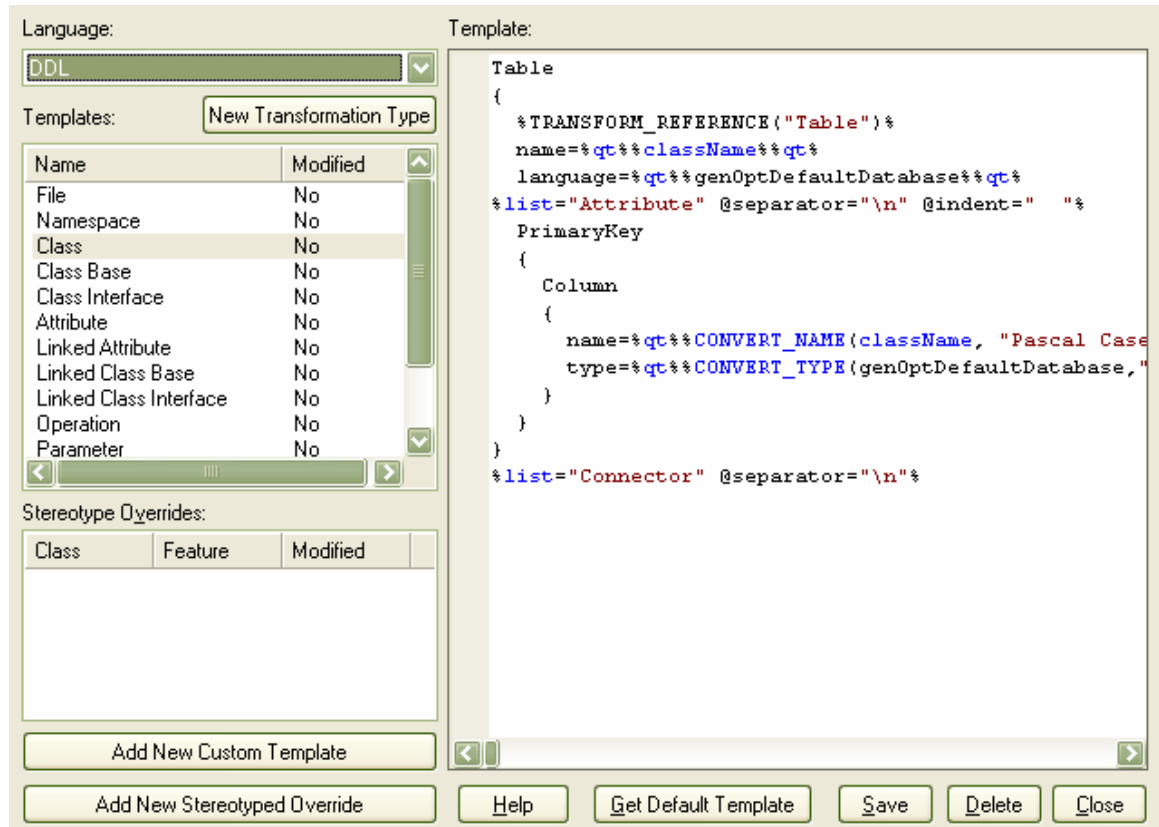


7.3 Transformation Templates

To modify Transformation Templates:

1. From the **Configuration** menu, click **Transformation Templates**.
2. In the **Transformation Templates Editor** dialog, set **Language** to show the name of the transformation that you want to modify.
3. Select a template from the list of **Templates**, and edit its contents in the editor pane.

- Click **Save**.



| Control | Description |
|------------------------------|---|
| Language | Selects the name of the transformation. |
| Template | Displays the contents of the active template. Provides the editor for modifying templates. |
| Templates | Lists the base transformation templates. The active template is highlighted. The Modified field indicates whether the user has changed the default template for the current transformation. |
| New Transformation Type | Creates a new transformation. |
| Stereotype Overrides | Lists the stereotyped templates, for the active base template. The Modified field indicates whether a default stereotyped template has been modified by the user. |
| Add New Stereotyped Override | Invokes a dialog for adding a stereotyped template, for the currently selected base template. |
| Add New Custom Template | Invokes a dialog for creating a custom stereotyped template. |
| Help | Launches the EA help file. |
| Get Default Template | Updates the editor display with the default version of the active template. |
| Save | Overwrites the active templates with the contents of the editor. |
| Delete | If the active template has been overridden by the user, the override is deleted and replaced by the corresponding default transformation template. |
| Close | Exits the Transformation Template Editor dialog. |

Note that the Transformation Template mechanism is based very strongly on the Code Generation Template mechanism. Extra information on Transformation Templates can be gained by reading the help sections for

code generation. The following pages may be of particular interest:

- [Overriding Default Templates](#)
- [Adding New Stereotyped Templates](#)
- [Creating Templates For Custom Languages](#)
- [Importing and Exporting Code Templates](#)

7.4 Built-in Transformations

Enterprise Architect comes with some built-in transformation types. These transformations have been designed to be useful to as many users as possible, to be a good base to modify to include the specifics of your custom domain, and to be good examples of how to write transformations.

The following transformations are included in EA.

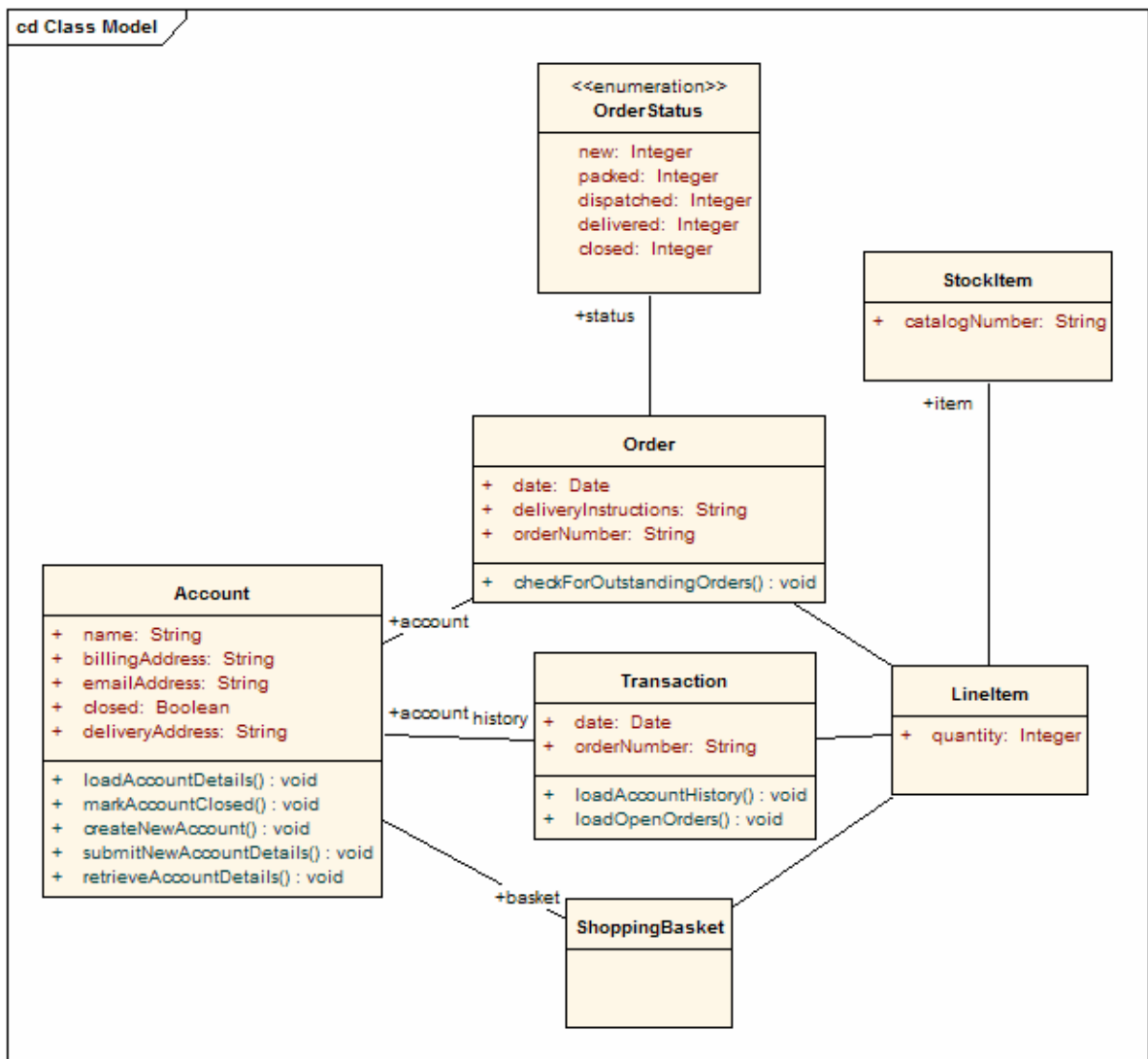
- [C#](#)
- [DDL](#)
- [EJB Entity](#)
- [EJB Session](#)
- [Java](#)
- [JUnit](#)
- [NUnit](#)
- [WSDL](#)
- [XSD](#)

7.4.1 C# Transformation

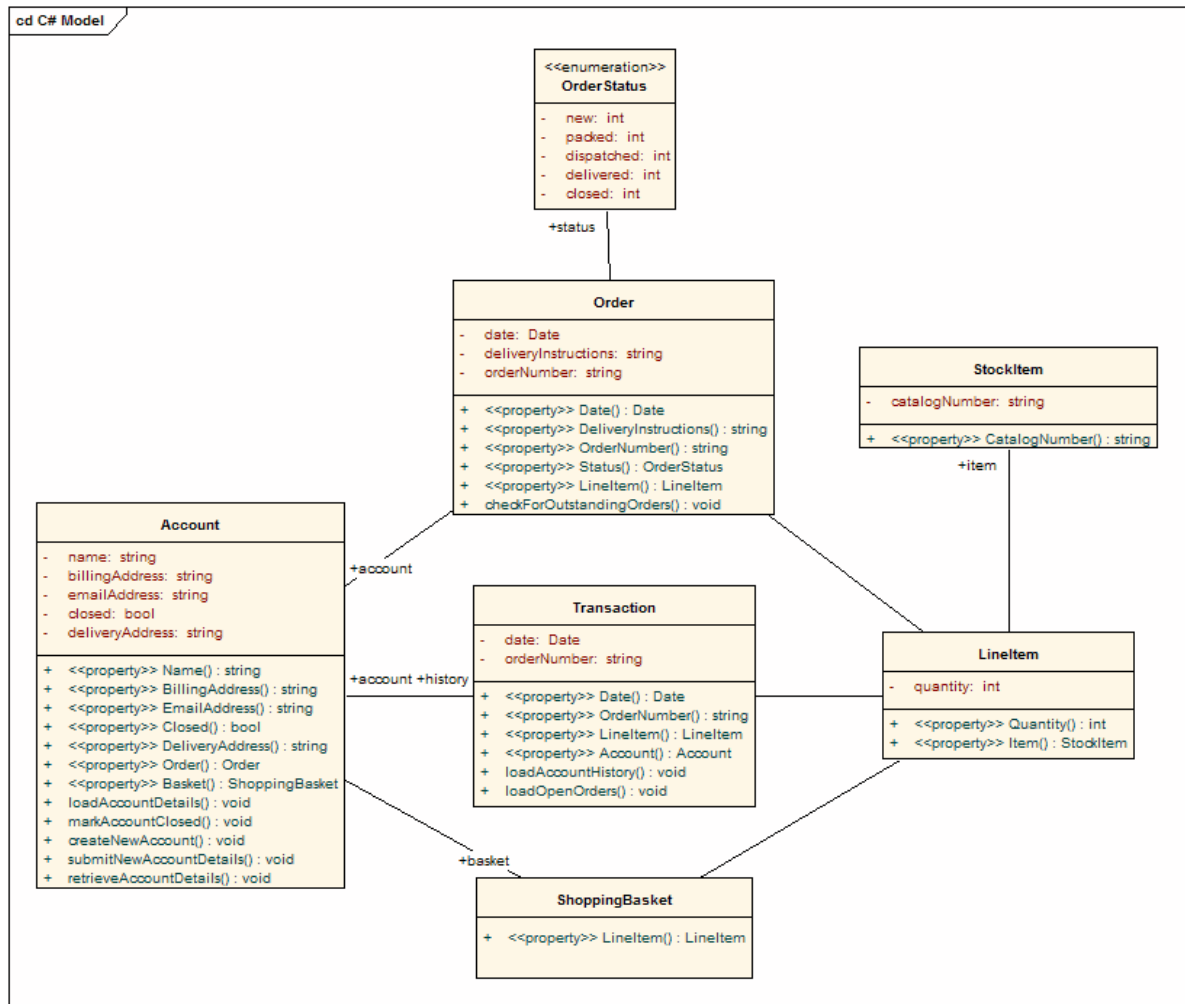
The purpose of the C# Transformation is to convert Platform-Independent Model (PIM) elements to language-specific C# class elements. The transformation converts the PIM model types to C# types and creates encapsulation according to EA's options for creating properties from C# attributes according to the rules as they have been defined by the user.

To set the code generation options for C# code generation, choose the **Options** command from the **Tools** menu and go to the "**Generation | C# Specifications**" page.

Our PIM:



Then becomes:



7.4.2 DDL Transformation

The purpose of the DDL Transformation is to create a data model from the logical model, generating a model targeted at the default database type that is ready for DDL generation.

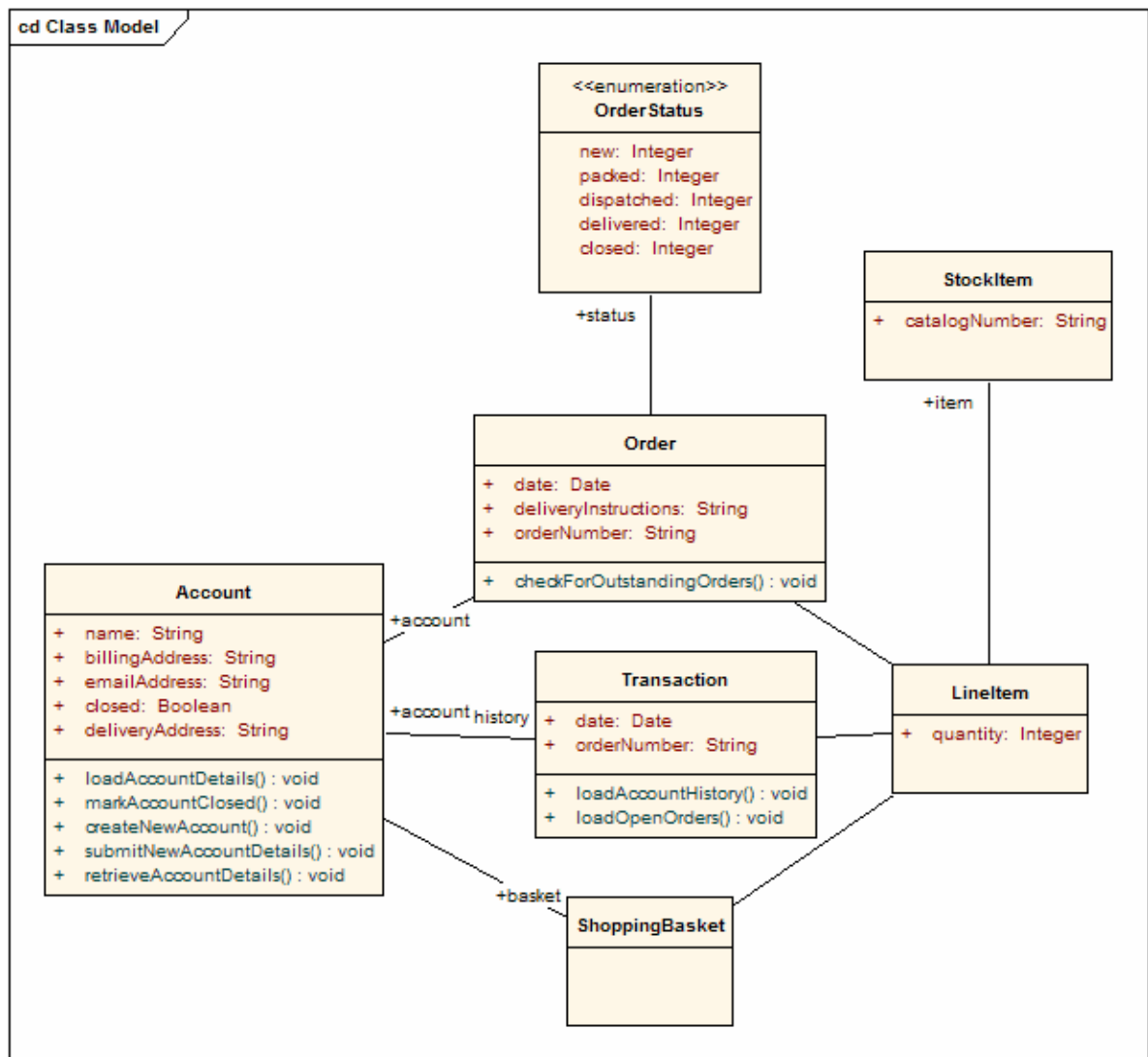
The Data Model may then be used to automatically generate DDL statements to run in one of the EA supported database products.

It utilises and demonstrates support in the [intermediary language](#) for the following database specific concepts:

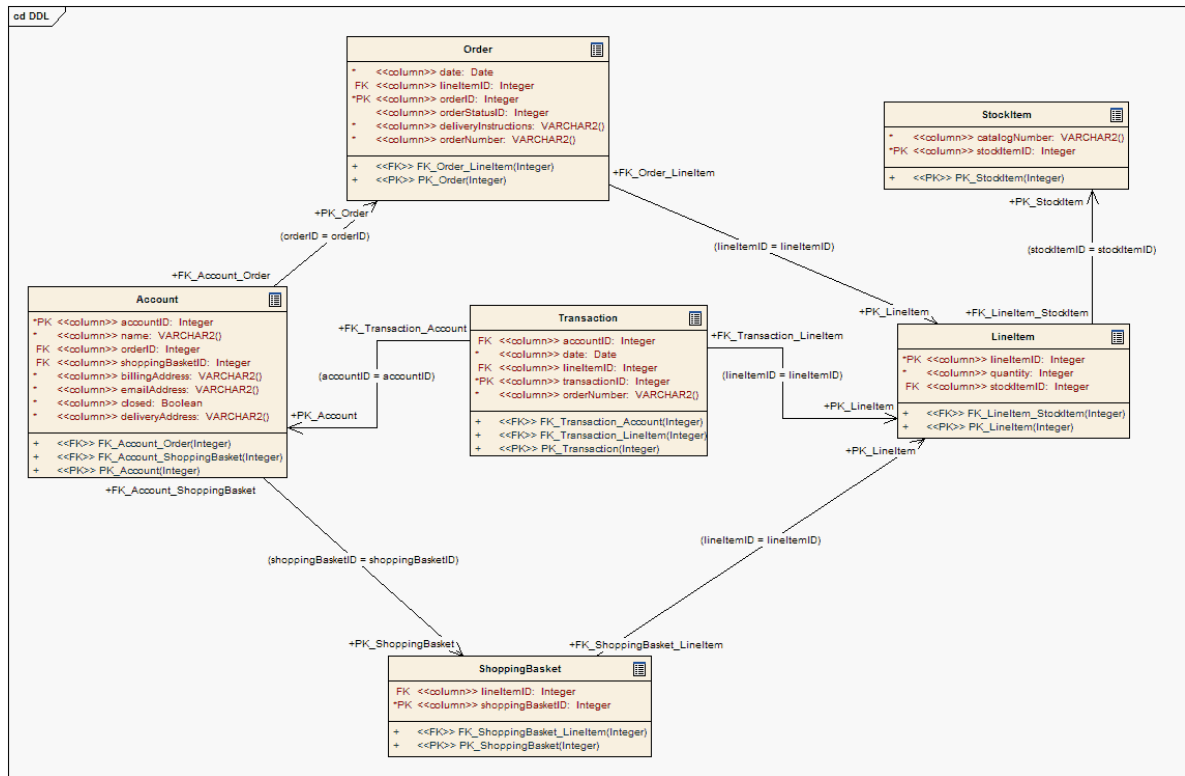
| | |
|-------------|---|
| Table | Mapped one-to-one onto class elements |
| Column | Mapped one-to-one onto attributes |
| Primary Key | List all the columns involved and this will ensure that they exist in the class and create a primary key method for them. |
| Foreign Key | This is a special sort of connector. In the Source and Target sections, list all of the columns involved and this will ensure that they exist, ensure that a matching primary key exists in the destination class and create the appropriate foreign key. |

The following two diagrams show a typical PIM to Data Model Transformation

Firstly the PIM:



And Secondly the PSM as automatically generated:



7.4.3 EJB Transformations

The purpose of the **EJB Session Bean Transformation** and the **EJB Entity Bean Transformation** is to reduce the work required in generating the internals of Enterprise Java Beans, thus allowing the modeler to concentrate on modelling at a higher level of abstraction.

The **EJB Session Bean Transformation** will generate the following from a single class element containing the attributes, operations and references required for code generation by the `javax.ejb.*` package:

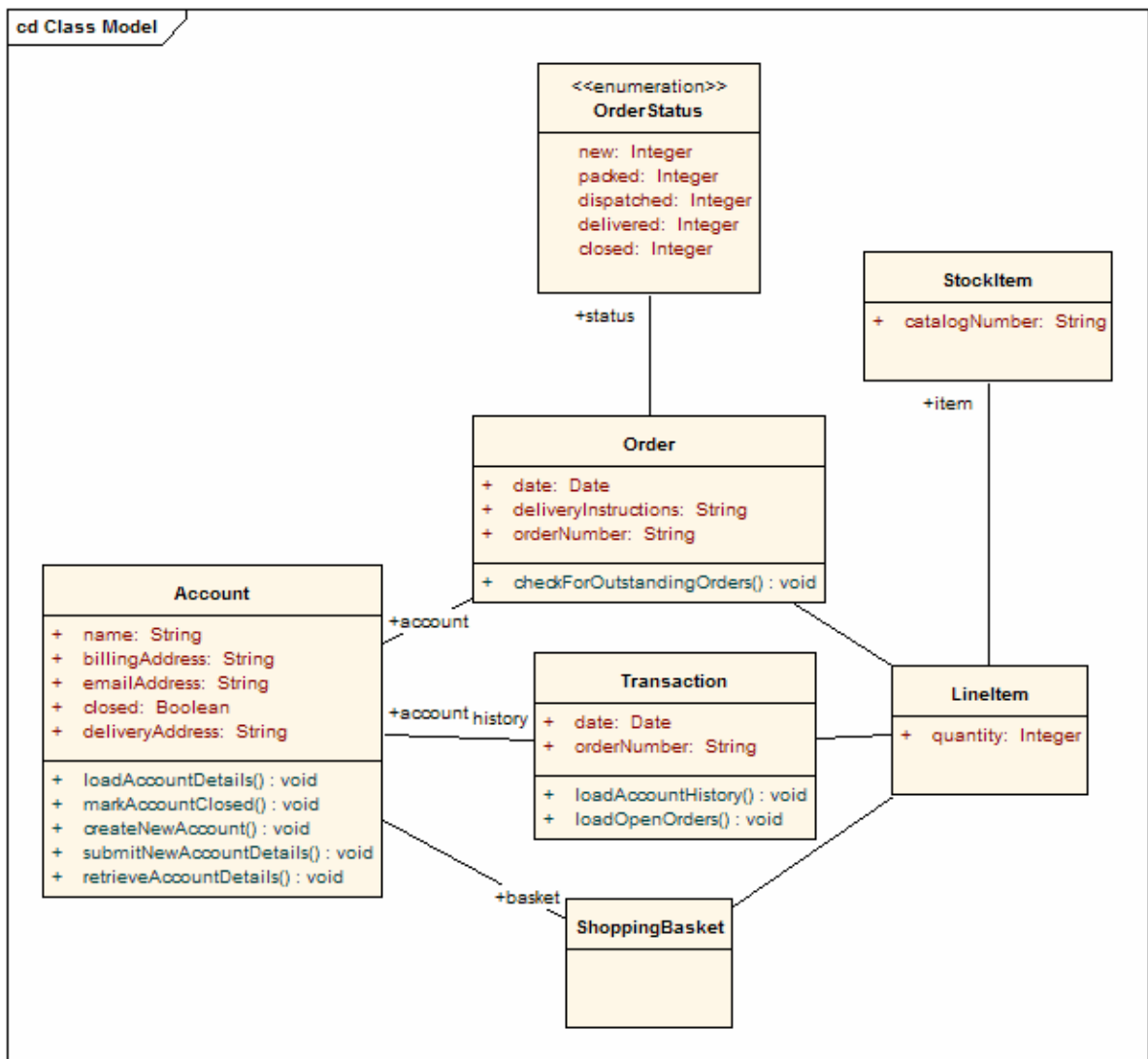
- An implementation class element.
- A home interface element.
- A remote interface element.

The **EJB Entity Bean Transformation** will generate the following from a single class element containing the attributes, operations and references required for code generation by the `javax.ejb.*` package:

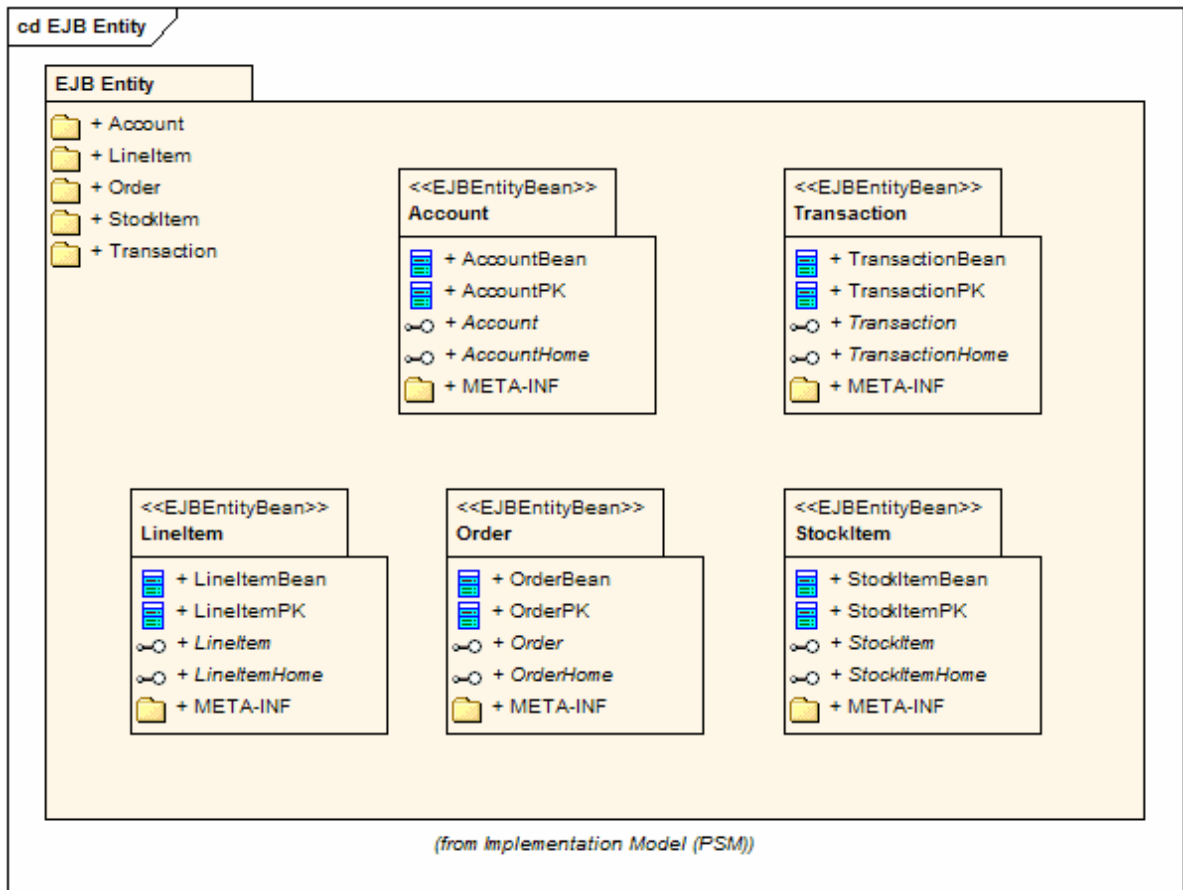
- An implementation class element.
- A home interface element.
- A remote interface element.
- A primary key element.

Both transformations will also generate a META-INF package containing a deployment descriptor element.

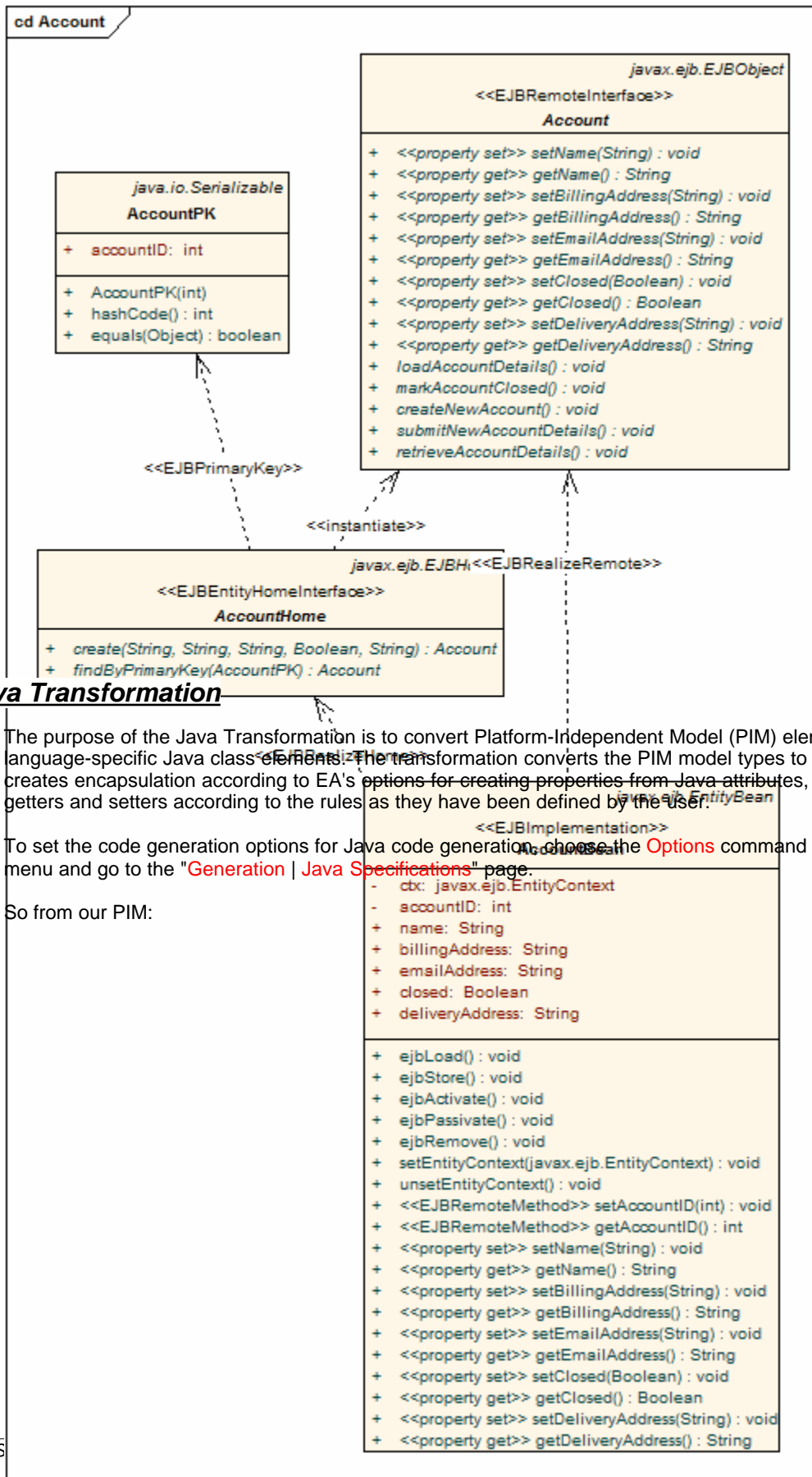
From our standard PIM:



We can generate a set of Entity Beans like this:



An example of an Entity Bean package (for the Account PSM) is below:

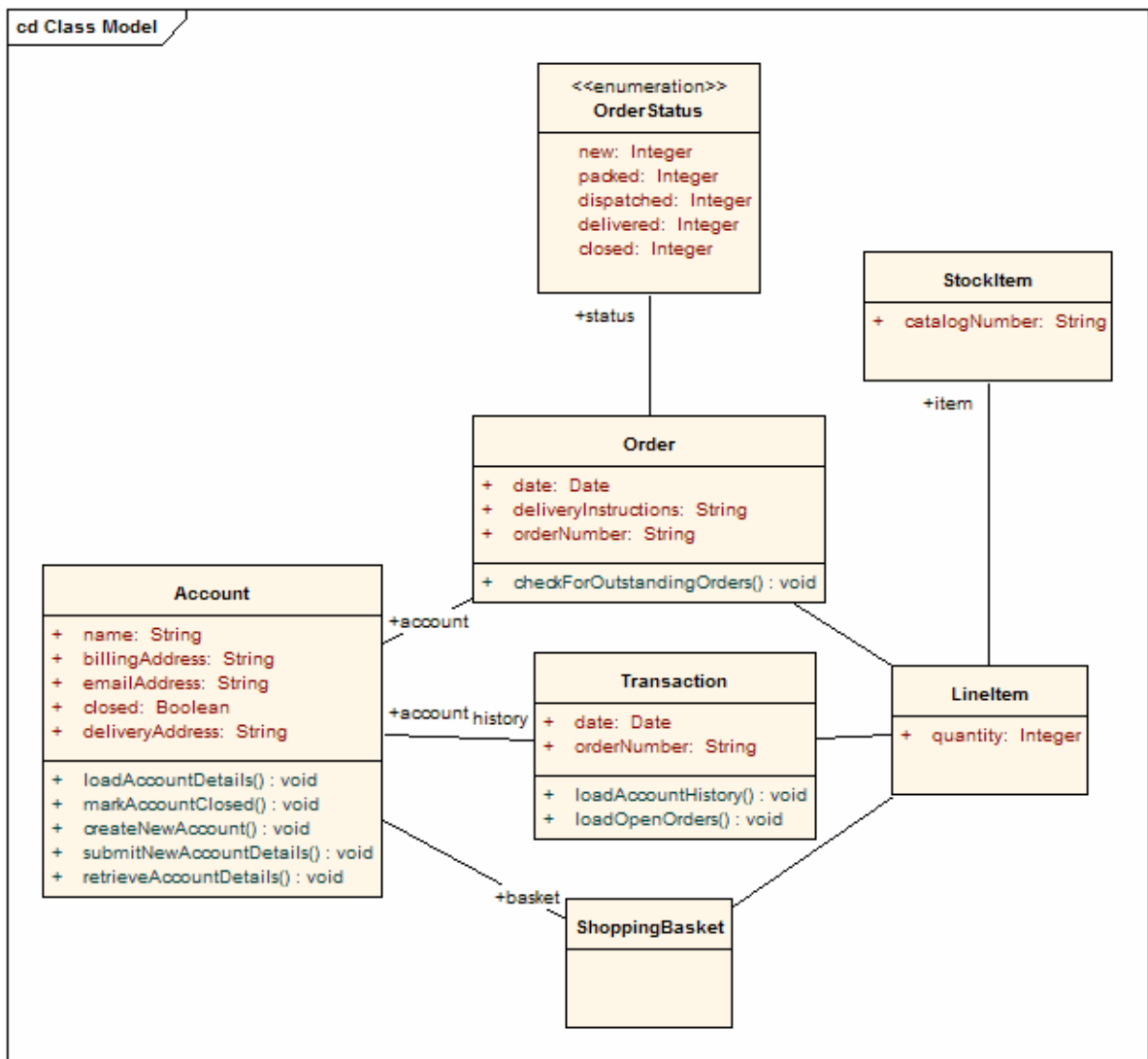


7.4.4 Java Transformation

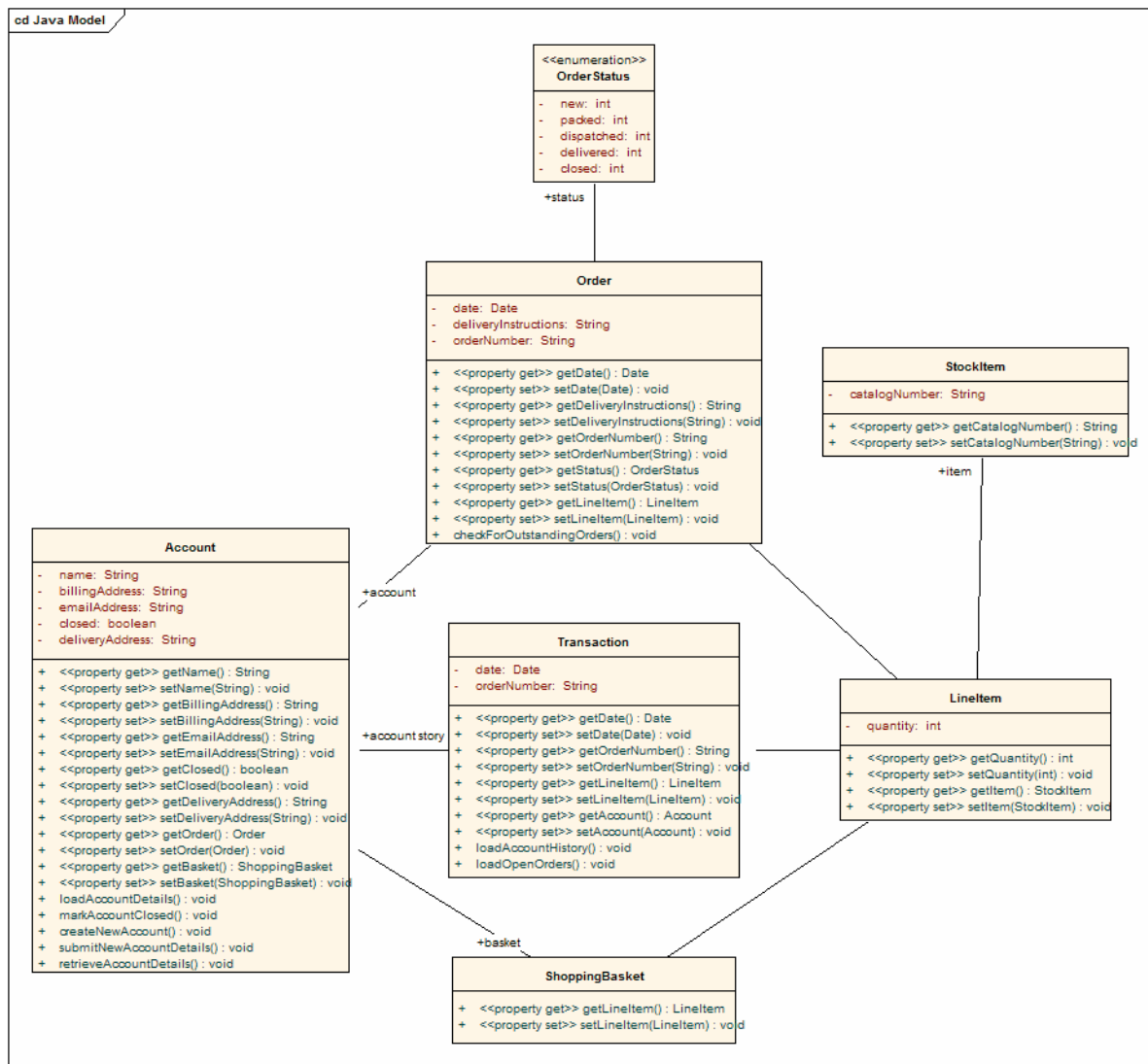
The purpose of the Java Transformation is to convert Platform-Independent Model (PIM) elements to language-specific Java class elements. The transformation converts the PIM model types to Java types and creates encapsulation according to EA's options for creating properties from Java attributes, i.e. producing the getters and setters according to the rules as they have been defined by the user.

To set the code generation options for Java code generation, choose the **Options** command from the **Tools** menu and go to the **"Generation | Java Specifications"** page.

So from our PIM:



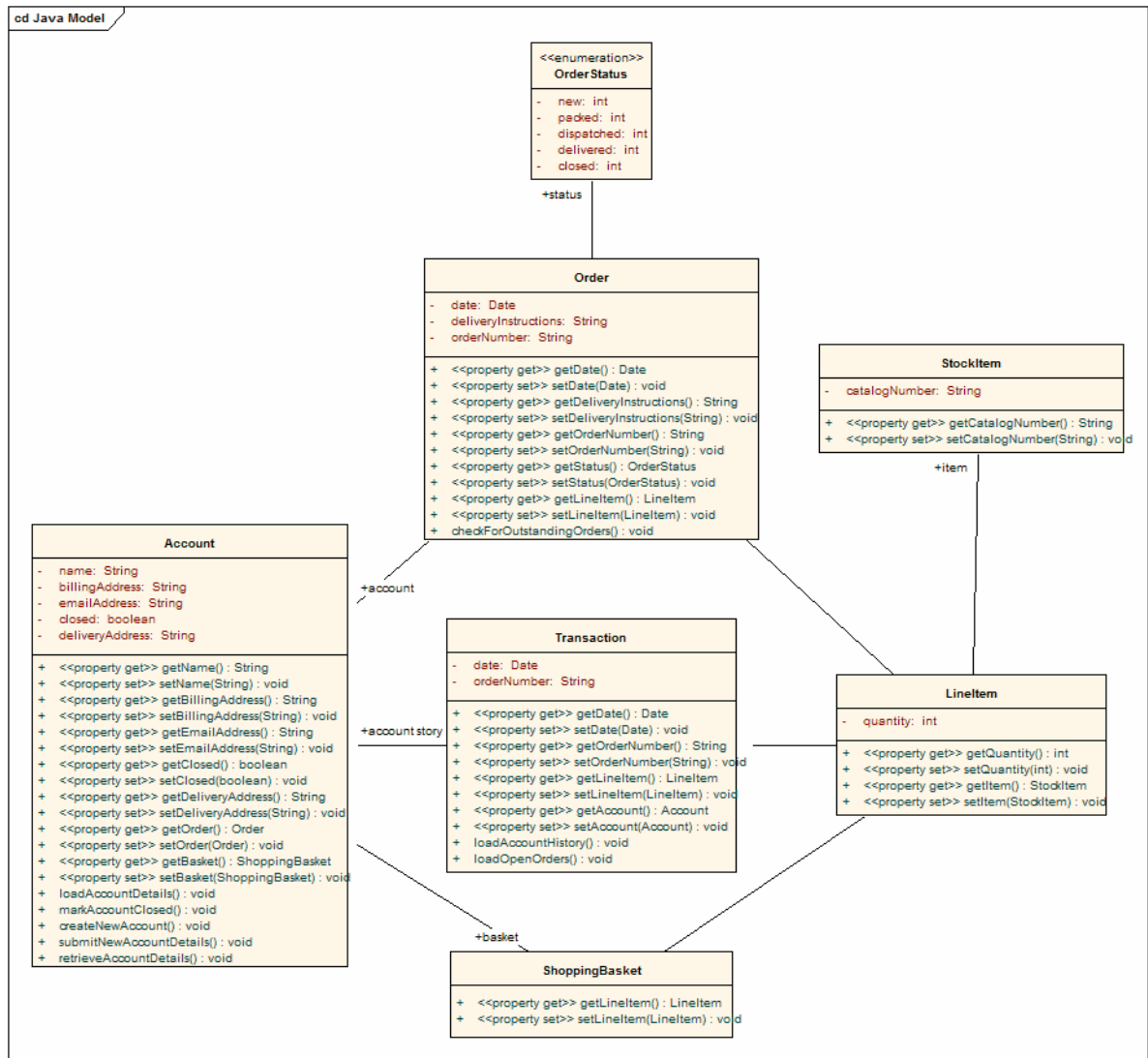
We transform to:



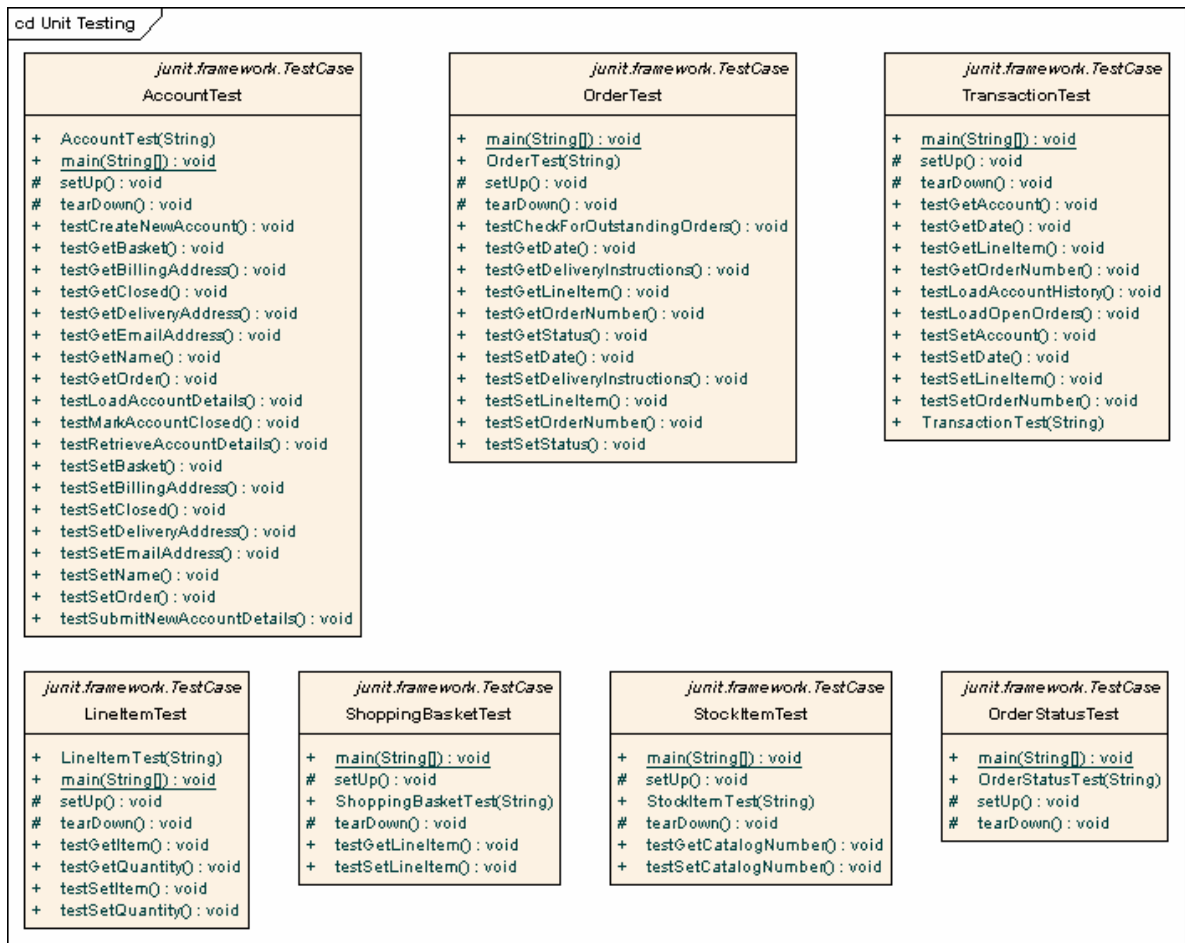
7.4.5 JUnit Transformation

The purpose of the JUnit transformation is to create a class with test methods for all public methods of an existing Java class. The resulting class can then be generated and the tests filled out and run by JUnit.

So from the java model we originally transformed from our PIM:



We transform to:



Note that for each class in the java model, a corresponding test class has been created. Each of these test classes contains a test method for every public method in the source class, plus the methods required to appropriately set up the tests. It is the responsibility of the user to fill in the details of each test.

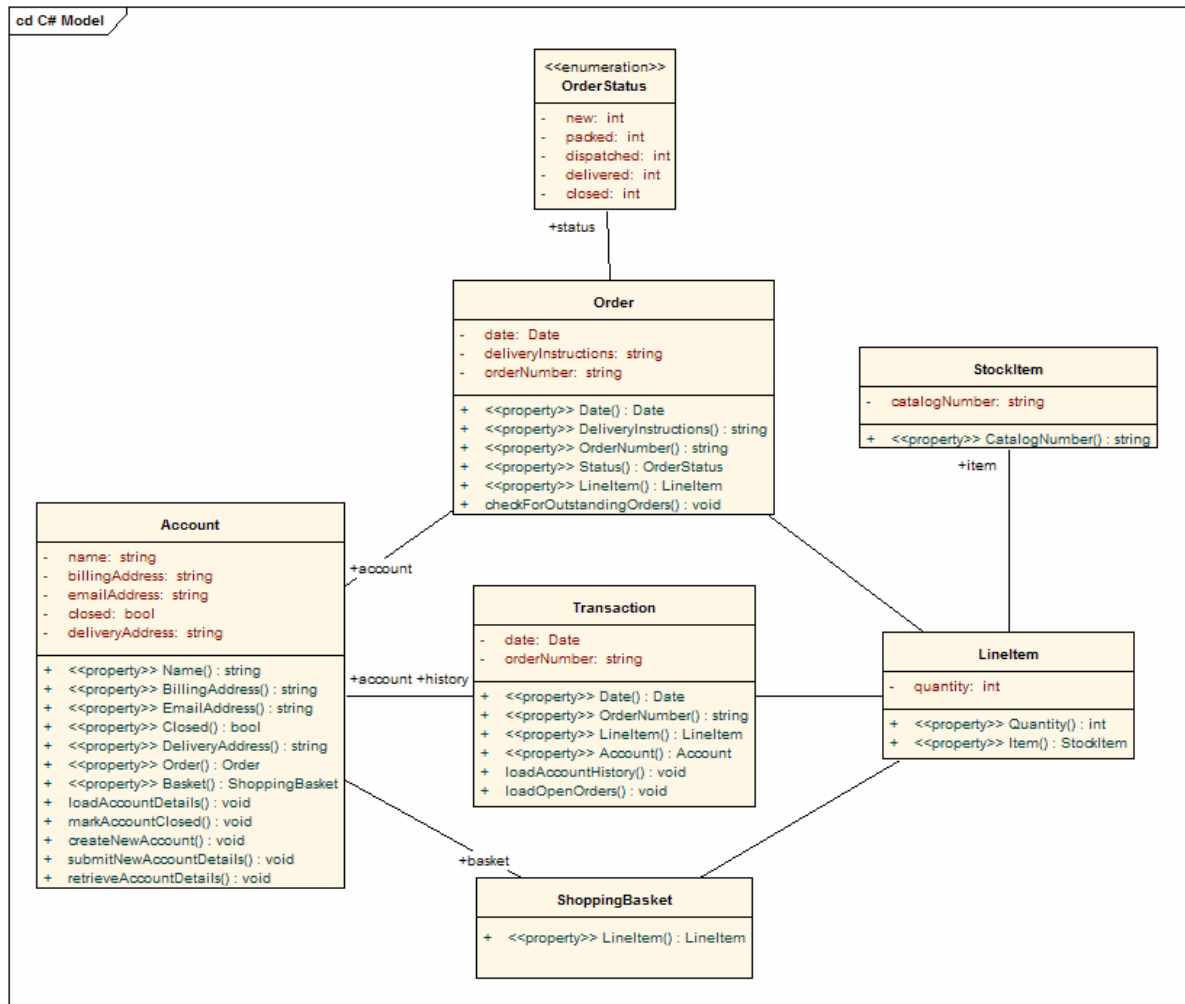
See Also

- [Unit Testing](#)

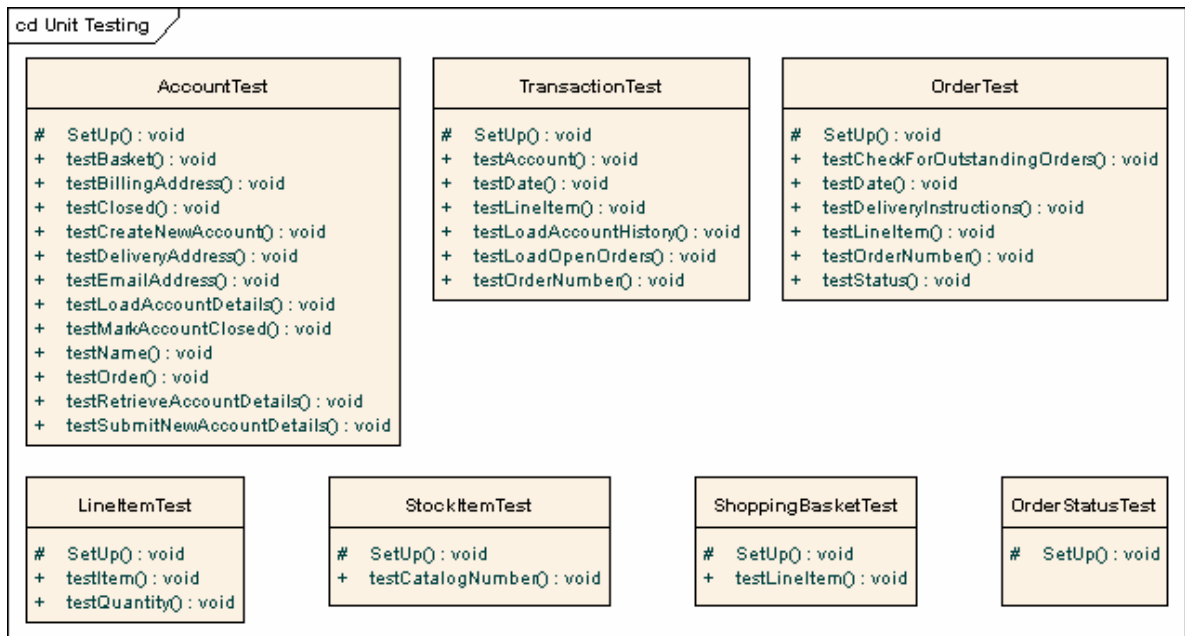
7.4.6 NUnit Transformation

The purpose of the NUnit transformation is to create a class with test methods for all public methods of an existing .Net compatible class. The resulting class can then be generated and the tests filled out and run by NUnit.

So from the C# model we originally transformed from our PIM:



We transform to:



Note that for each class in the C# model, a corresponding test class has been created. Each of these test classes contains a test method for every public method in the source class, plus the methods required to appropriately set up the tests. It is the responsibility of the user to fill in the details of each test.

See Also

- [Unit Testing](#)

7.4.7 WSDL Transformation

The purpose of the WSDL transformation is to take a simple model and create an expanded model of a WSDL interface that is suitable for generation.

Take the following example interface.



This generates the corresponding WSDL component, service, port type, binding and messages as follows.

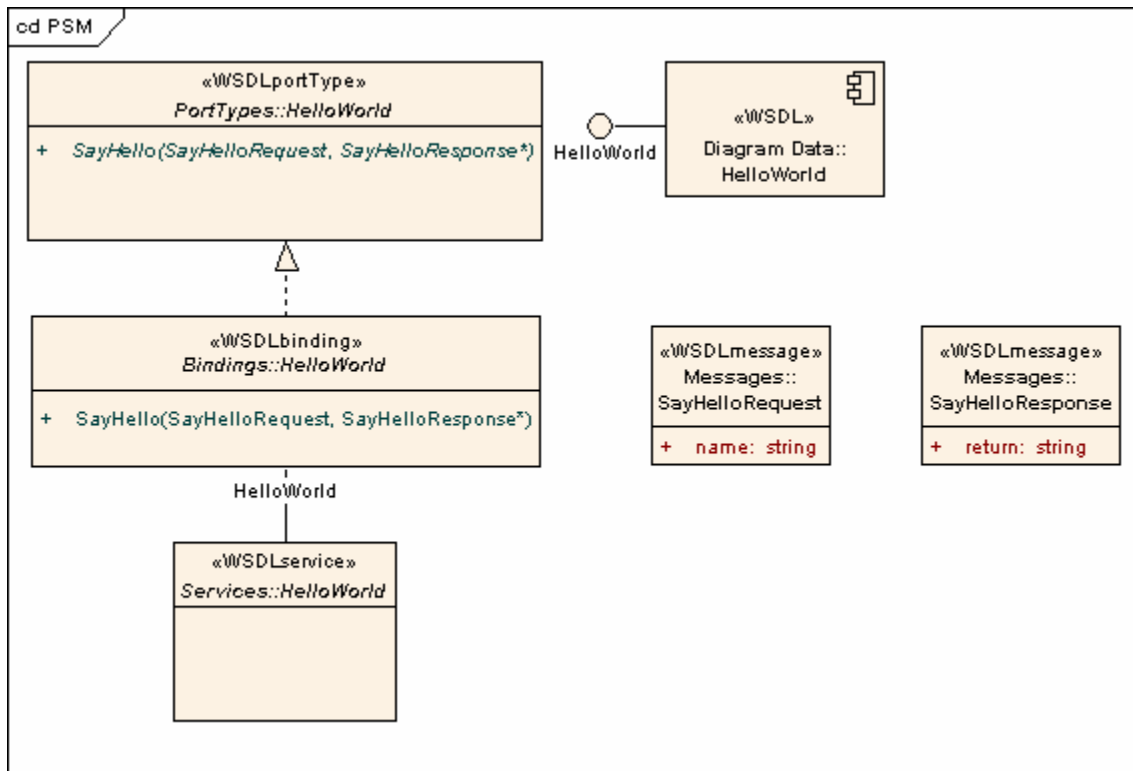
Classes are handled in the same way as the [XSD Transformation](#).

All in parameters are transformed into messageParts in the Request message.

The return value and all out and return parameters are transformed into messageParts in the Request message.

All methods where a value is returned are transformed into Request-Response operations while all methods not returning a value are transformed into OneWay operations.

The transformation does not currently handle generation of Solicit-Response and Notification methods or faults.



The resulting package can then have the specifics filled out using the WSDL editing capabilities of EA, and finally generated using the WSDL generation.

See Also

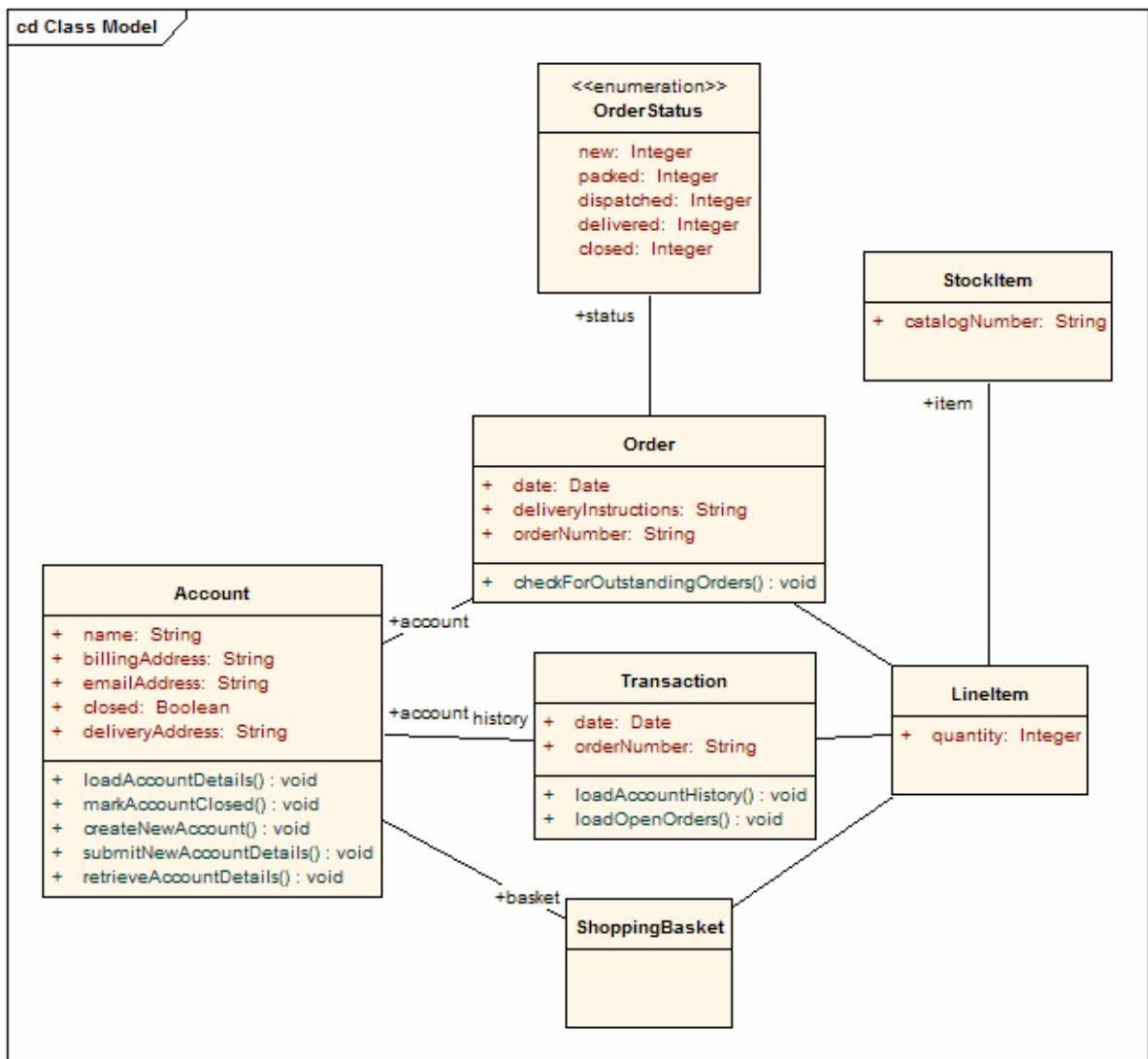
- [Model WSDL](#)
- [Generate WSDL](#)

7.4.8 XSD Transformation

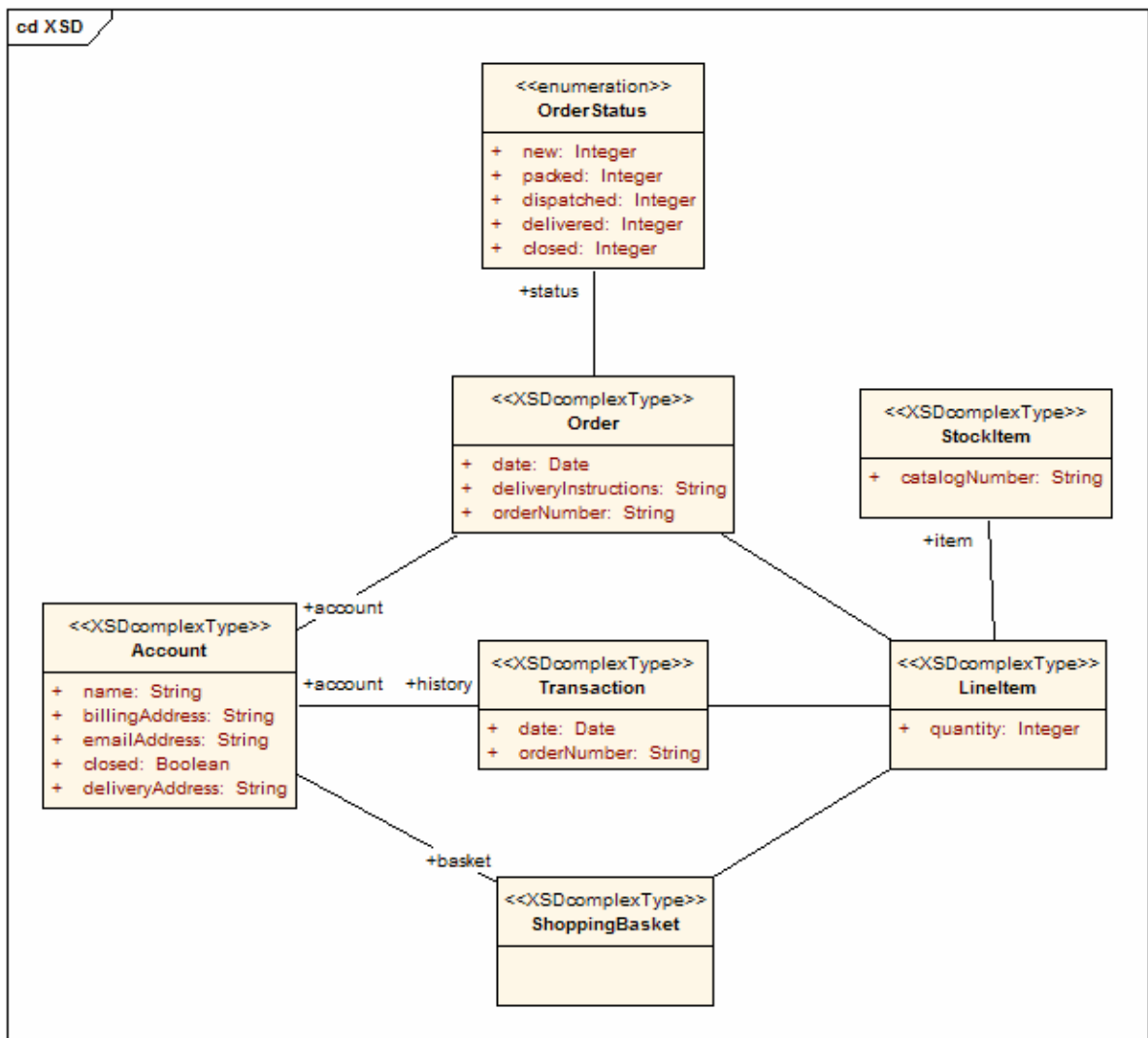
The purpose of the XSD Transformation is to convert Platform-Independent Model (PIM) elements to UML Profile for XML elements as an intermediary step to creating an XML Schema. Each selected PIM class element is converted to a <<XSDcomplexType>> element.

For more information, see [XML Schema Generation](#).

From our standard PIM



We Transform to:



Which in turn generates the XSD below:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Account" type="Account" />
  <xs:complexType name="Account">
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="billingAddress" type="xs:string" />
      <xs:element name="emailAddress" type="xs:string" />
      <xs:element name="closed" type="xs:boolean" />
      <xs:element name="deliveryAddress" type="xs:string" />
      <xs:element ref="Order" />
      <xs:element ref="ShoppingBasket" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="LineItem" type="LineItem" />
  <xs:complexType name="LineItem">
    <xs:sequence>
      <xs:element name="quantity" type="xs:integer" />
      <xs:element ref="StockItem" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Order" type="Order" />

```

```

<xs:complexType name="Order">
  <xs:sequence>
    <xs:element name="date" type="xs:date" />
    <xs:element name="deliveryInstructions" type="xs:string" />
    <xs:element name="orderNumber" type="xs:string" />
    <xs:element ref="LineItem" />
    <xs:element name="status" type="OrderStatus" />
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="OrderStatus">
  <xs:restriction base="xs:string">
    <xs:enumeration value="new" />
    <xs:enumeration value="packed" />
    <xs:enumeration value="dispatched" />
    <xs:enumeration value="delivered" />
    <xs:enumeration value="closed" />
  </xs:restriction>
</xs:simpleType>
<xs:element name="ShoppingBasket" type="ShoppingBasket" />
<xs:complexType name="ShoppingBasket">
  <xs:sequence>
    <xs:element ref="LineItem" />
  </xs:sequence>
</xs:complexType>
<xs:element name="StockItem" type="StockItem" />
<xs:complexType name="StockItem">
  <xs:sequence>
    <xs:element name="catalogNumber" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Transaction" type="Transaction" />
<xs:complexType name="Transaction">
  <xs:sequence>
    <xs:element name="date" type="xs:date" />
    <xs:element name="orderNumber" type="xs:string" />
    <xs:element ref="Account" />
    <xs:element ref="LineItem" />
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

See Also

- [Model XSD](#)
- [Generate XSD](#)

7.5 Writing Transformations

This section provides help in writing your own transformations. Subjects covered are:

- [Default Transformation Templates](#)
- [General Syntax for the Intermediary Language](#)
- [Syntax for Creating Objects](#)
- [Syntax for Creating Connectors](#)
- [Transforming Duplicate Information](#)
- [Converting Types](#)
- [Converting Names](#)
- [Cross References](#)

Further hints and tips can be gleaned by a close study of the Transformation Templates provided with EA. Note also that writing transformations is very similar to writing code generation templates, so an understanding of the [Code Template Framework](#) will greatly assist in the understanding of transformations.

Transformation Templates are accessed from the [Settings | Transformation Templates](#) item in the main menu.

7.5.1 Default Transformation Templates

In most transformations, there will be a lot of information that is simply copied to the target model. In order to make writing new transformations simpler EA provides a default set of transformation templates. These templates perform a simple copy of the source model to the target model. This means that in order to write a new transformation you can modify the default templates to make the changes you need.

Note: When creating a new transformation you need to modify at least one template before the transformation becomes available.

7.5.2 Intermediary Language

All transformations in EA work by generating a text form of the model that you wish to generate.

Any element is represented in this language by the type of element (eg. Class, Action, Method, Generalization or Tag) followed by the properties of the element and the elements that it is made from. The grammar for this looks like the following.

```

element:
    elementName { (elementProperty | element)* }

elementProperty:
    packageName
    stereotype
    propertyName = " propertyValueSymbol* "

packageName:
    name = " propertyValueSymbol* " ( . " propertyValueSymbol* " )*

stereotype:
    stereotype = " propertyValueSymbol* " ( , " propertyValueSymbol* " )*

propertyValueSymbol:
    \\
    \"
    Any character except " (U+0022), \ (U+005C)

```

- *elementName* is any one of the set of element types as described in [Objects](#) and [Connectors](#).
- *propertyName* is any one of the set of properties as described in [Objects](#) and [Connectors](#).

Literal strings can be included in property values by escaping a quote character. For example.

```
default = "\"Somestringvalue.\""
```

7.5.3 Objects

Objects are created in EA by generating some text in the following form.

```

objectType
{
  o bjectProperties
}

```

where:

objectType is one of the following object types:

- Action
- ActionPin
- Activity

- ActivityParameter
- ActivityPartition
- ActivityRegion
- Actor
- Association
- Change
- Class
- Collaboration
- CollaborationOccurrence
- Component
- DeploymentSpecification
- DiagramFrame
- Decision
- EntryPoint
- Event
- ExitPoint
- ExceptionHandler
- ExpansionNode
- ExpansionRegion
- ExposedInterface
- GUIElement
- InteractionFragment
- InteractionOccurrence
- InteractionState
- Interface
- InterruptibleActivityRegion
- Issue
- Iteration
- Object
- ObjectNode
- MessageEndpoint
- Node
- Package
- Parameter
- Part
- Port
- ProvidedInterface
- RequiredInterface
- Requirement
- Sequence
- State
- StateNode
- Synchronization
- Table
- TimeLine
- UMLDiagram
- UseCase

objectProperties is zero or more of the following properties:

- Abstract
- Alias
- Arguments
- Author
- Cardinality
- Classifier
- Complexity
- Concurrency
- Filename

- Header
- Import
- IsActive
- IsLeaf
- IsRoot
- IsSpecification
- Keyword
- Language
- Multiplicity
- Name
- Notes
- Persistence
- Phase
- Scope
- Status
- Stereotype
- Version
- Visibility

and zero or more of the following elements:

- Attribute
- Parameter
- Operation
- Parent
- Tag
- XRef
- Any object

Note: Some of the above only apply to certain object types.

Note: Every object created in a transformation should include an [XRef element](#) as it allows EA to synchronize with the element and makes it possible to create a connector to that class in a transformation.

Classes

A simple class can be created as follows.

```
Class
{
  name = "Example"
}
```

It is then easy to add to this. The following example sets the language to C++, adds a tagged value and an attribute.

```
Class
{
  name = "Example"
  language = "C++"
  Tag
  {
    name = "defaultCollectionClass"
    value = "List"
  }
  Attribute
  {
    name = "count"
    type = "int"
  }
}
```

Attributes

Attributes are created with the same structure as objects, and include the following properties

- Alias
- Collection

- Container
 - Containment
 - Constant
 - Default
 - Derived
 - LowerBound
 - Name
 - Notes
 - Ordered
 - Scope
 - Static
 - Stereotype
 - Type
 - UpperBound
 - Volatile
- and the following elements
- Tag
 - XRef

Operations

Operations are created with the same structure as objects, and include the following properties

- Abstract
 - Alias
 - Behavior
 - Code
 - Constant
 - IsQuery
 - Name
 - Notes
 - Pure
 - ReturnArray
 - Scope
 - Static
 - Stereotype
 - Type
- and the following elements
- Parameter
 - Tag
 - XRef

Parameters

Parameter are created with the same structure as objects, and include the tag element and the following properties

- Default
- Fixed
- Name
- Notes
- Kind
- Stereotype

Packages

Packages differ from other objects the following ways.

- A reduced set of properties of alias, author, name, namespaceRoot, notes, scope, stereotype and version.
- An extra property namespaceRoot.
- Must have a name specified.
- Name can be a qualified name. When a qualified name is specified the properties given are applied only to the final package.
- Can contain other packages.
- Can't contain attributes and operations.

Tables

Tables are a special sort of object with the following difference from other object types.

- Can include columns and primary keys
- Can't include attributes and operations

Column

Columns are similar to attributes, but have an autonumber element containing startnum and increment and has the following properties added.

- Length
- NotNull
- Precision
- PrimaryKey
- Scale
- Unique

7.5.4 Connectors

Creating connectors in a transformation can be complex, but it follows the same form as creating elements. The difference is that you also need to specify each end.

The different connectors that can be created are as follows.

- Aggregation
- Assembly
- Association
- Collaboration
- ControlFlow
- Connector
- Delegate
- Dependency
- Deployment
- ERLink
- ForeignKey
- Generalization
- Instantiation
- Interface
- InterruptFlow
- Manifest
- Nesting
- NoteLink
- ObjectFlow
- Package
- Realisation
- Sequence
- StateFlow
- Uses

Note: ForeignKey is a special case where not just a connector is created: you also need to list the columns involved in the transformation.

There are two different types of classes that you can use as a connector end: one created by a transformation; or one for which you already know the GUID.

Connecting to a class created by a transformation

The most common way is to connect to a class created by a transformation. To do this you need to know three things.

- The original class GUID.
- The name of the transformation.
- The name of the transformed class.

This type of link is created using the TRANSFORM_REFERENCE function macro. When the element is in the current transformation, it can be safely omitted from the transformation. The easiest example of this is when you have created multiple classes from a single class in a transformation and want a link between them. Consider this example from the EJB Entity transformation:

```

Dependency
{
  %TRANSFORM_REFERENCE("EJBRealizeHome",classGUID)%
  stereotype="EJBRealizeHome"
  Source
  {
    %TRANSFORM_REFERENCE("EJBEntityBean",classGUID)%
  }
  Target
  {
    %TRANSFORM_REFERENCE("EJBHomeInterface",classGUID)%
  }
}

```

There are three uses of the [TRANSFORM_REFERENCE](#) macro, one to identify this connector for synchronisation purposes and the other two to identify the ends. All three use the same source GUID, because they all come from the one original class. None of the three need to specify the transformation because the two references are referencing something in the current transformation. Each of them then only needs to identify the transform name.

Of course it is also possible to create a connector from another connector.

You can create a connector template and list over all connectors connected to a class from the class level templates. You don't need to worry about only generating it once, as if you have created a TRANSFORM_REFERENCE for the connector then EA will automatically synchronise them. The following copies the source connector.

```

%connectorType%
{
  %TRANSFORM_CURRENT() %
  %TRANSFORM_REFERENCE("Connector",connectorGUID)%
  Source
  {
    %TRANSFORM_REFERENCE("Class",connectorSourceGUID)%
    %TRANSFORM_CURRENT("Source") %
  }
  Target
  {
    %TRANSFORM_REFERENCE("Class",connectorDestGUID)%
    %TRANSFORM_CURRENT("Target") %
  }
}

```

Connecting to a class for which you know the GUID

The second type of class that you can use as a connector end is one that you know the current GUID of. To do this, specify the GUID of the target class in either the Source or Target end. The following example creates a dependency from a class created in a transformation, to the class it was transformed from.

```

Dependency
{
  %TRANSFORM_REFERENCE("SourceDependency",classGUID)%
  stereotype="transformedFrom"
  Source
  {
    %TRANSFORM_REFERENCE("Class",classGUID)%
  }
  Target
  {
    GUID=%qt%%classGUID%%qt%
  }
}

```

```
}
}
```

7.5.5 Duplicate Information

In many transformations there will be a substantial amount of information that will be copied. It would be tedious to type all of the common information into a template so that it is copied to the transformed class. The alternative is to use the TRANSFORM_CURRENT function macro to do exactly this.

TRANSFORM_CURRENT(<listOfExcludedItems>)

Will generate an exact copy of all the properties of the current item, except for the items named in <listOfExcludedItems>.

Another form of this is available when transforming connectors that allows either end of the connector to be copied.

TRANSFORM_CURRENT(<connectorEnd>, <listOfExcludedItems>)

Will generate an exact copy of the connector end specified by <connectorEnd> except for the items named in <listOfExcludedItems>.

Where <connectorEnd> is either "Source" or "Target".

7.5.6 Converting Types

Different target platforms will almost certainly require different types, so very often you will want a method of converting between different types. The following macro offers that.

CONVERT_TYPE(<destinationLanguage>, <originalType>)

Will convert <originalType>, to the corresponding type in <destinationLanguage> using the datatypes and common types defined in the model.

Where:

- <originalType> is assumed to be a platform independent common type.

7.5.7 Converting Names

Different target platforms may use different naming conventions, as a result you may not want to directly copy the names of your elements directly into the transformed models. To facilitate this need EA's transformation templates provide a CONVERT_NAME function macro.

CONVERT_NAME(<originalName>, <originalFormat>, <targetFormat>)

Will convert <originalName>, which is assumed to be in <originalFormat> to <targetFormat>.

The supported formats are:

- Camel Case: New words start with a capitalised letter. The first letter of the first word is lower case.
- Pascal Case: Same as camel case but the first letter of the first word is upper case.
- Spaced: Words are separated by spaces.
- Underscored: Words are separated by underscores and assumed to be lower case.

Note: Acronyms are not supported when converting from camel case or pascal case.

Another way in which you may want to transform a name is to remove a prefix from the original name.

REMOVE_PREFIX(<originalName>, <prefixes>)

Removes any prefix found in <prefixes> from <originalName>. The prefixes are specified in a semi-colon separated list.

This will often be used in conjunction with the CONVERT_NAME macro. For example, the following creates a get property name according to the options for Java.

```
$propertyName=%REMOVE_PREFIX(attName,genOptPropertyPrefix)%  
%if genOptGenCapitalisedProperties=="T"%  
$propertyName=%CONVERT_NAME($propertyName, "camel case", "pascal case")%  
%endif%
```

7.5.8 Cross References

Cross References are an important part of transformations. They are used for:

- Finding the transformed class to synchronise with.
- Creating connectors between transformed classes.
- Determining where to transform to for future transformations.

Each cross reference is made up from three different parts. They are:

- A Namespace, corresponding to the transformation that generated the element.
- A Name, is a unique reference to something that can be generated in the above transformation.
- A Source, is the GUID of the element that this element was created from.

When writing the templates for a transformation, it is easiest to create the cross references using the TRANSFORM_REFERENCE function macro which is defined for this purpose. It has three parameters, each of which is optional.

TRANSFORM_REFERENCE(<name>, <sourceGuid>, <namespace>)

Generates the a reference that can be used in the ways described above. It will look like the following.

```
XRef{namespace="<namespace>" name="<name>" source="<sourceGuid>" }
```

Where:

- If <name> is not specified it gets the name of the current template.
- If <sourceGUID> is not specified it gets the GUID of the current class.
- If <namespace> is not specified it gets the name of the current transformation.

Note: The only time that this should be specified is when creating a connector to a class created in a different transformation.

A good example of the use of cross references is in the [DDL](#) templates provided with EA. In the Class template a cross reference is created with the name table. Then up to two different connectors can be created, each of which needs to identify the two classes it connects using cross references while needing its own unique cross reference.

Part

8

8 Model Management

The Model Management section covers the following topics:

Model Files

An Enterprise Architect model is stored in a data repository. EA allows you to work with [.EAP files](#) and [DBMS repositories](#). A [.EAP file](#) is a Microsoft JET database. You can also work with the following data repositories in EA Corporate edition:

- [SQL Server](#)
- [MySQL](#)
- [Oracle 9i and 10g](#)
- [PostgreSQL](#)
- [Adaptive Server Anywhere](#)
- [MSDE Server](#)
- [Progress OpenEdge](#)

Information on how to get started with models can be found in the [Create/Open Model Files](#) section.

Replication

Note: *This functionality is available in the Professional and Corporate editions only. The Desktop edition is intended for single users, so does not support replication.*

In addition to sharing projects in real time over a network, EA also allows for projects to be shared using [replication](#). Replication is a powerful means of distributing a single .EAP model over a wide area, with occasional synchronization between Design Master and Replica.

Project Sharing

Note: *This functionality is available in the Professional and Corporate editions only. The Desktop edition is intended for single users, so does not support shared files.*

Enterprise Architect allows [project sharing](#) for efficient management of team development. You can create a replica of your project, make changes to it, then merge your changes back into the master project.

User Security

Note: *This feature is available in the Corporate edition only.*

[User security](#) provides a means of limiting access to update functions in a model. Elements may be locked on a per user or per group basis and a password required to login.

Data Transfer

Note: *This feature is available in the Corporate edition only.*

Enterprise Architect provides functionality to perform [data transfer](#) between data repositories, row by row, table by table.

See Also

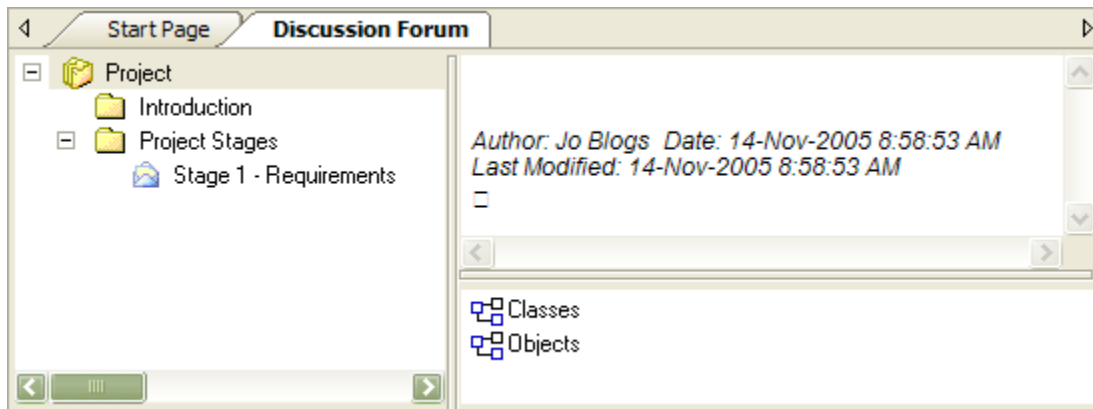
- [Create and Open Model Files](#)
- [Upgrading Models](#)
- [Data and Model Integrity](#)
- [Data Transfer](#)
- [Upsizing Models](#)
- [Model Maintenance](#)
- [Manage Views](#)
- [Import and Export](#)

- [Version Control Options](#)
- [Team Development](#)
- [Spell Checking](#)
- [Reference Data](#)

8.1 Project Discussion Forum

The *Project Discussion Forum* can be used to discuss the development and progress of a project. The forum window consists of three main areas, the message thread area, the message contents area and the linked objects area. The message thread area which is located in the left pane is used to create new categories and topics and to edit and delete messages. The message contents section is used to view discussion topics and is located in the top right hand section of the discussion forum. The linked elements area is located in the lower right hand portion of the discussion forum and is used to locate model elements of interest and to associate model elements to the discussion forum.

You can access the *Project Discussion Forum* via the main menu, select *View | Project Discussion Forum*. [*Ctrl+Alt+U*]



- Post read.
- Post unread.
- Post Reply
- Unread Reply
- Category
- Category unread
- Topic read.
- Topic unread.

See Also

- [Categories, Topics and Posts](#)
- [Message Dialog](#)
- [Adding Element Links](#)
- [Copy Path to Clipboard](#)
- [Forum Connections](#)

8.1.1 Categories, Topics and Posts

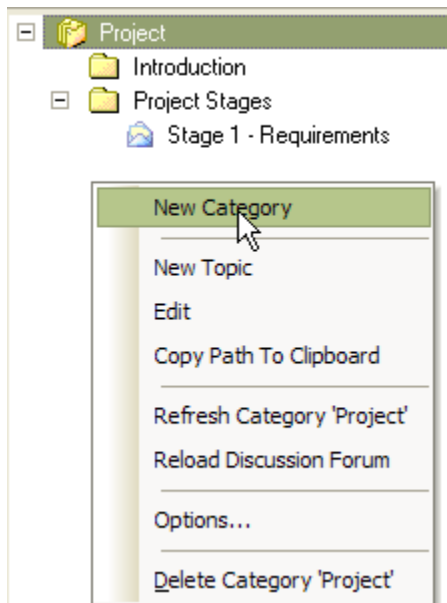
The *Project Discussion Forum* allows users to create *Categories* which contain *Topics* which contain *Posts*. Users also have the ability to reply to posts.

See Also

- [Adding a New Category](#)
- [Adding a New Topic](#)
- [Adding a New Post](#)
- [Replying to a Post](#)
- [Editing an Item](#)
- [Deleting an Item](#)
- [Forum Connections](#)

8.1.1.1 Adding a New Category

To create a new *Category*, right click on a blank area in the message thread window, select *New Category* from the context menu or alternatively press [**Ctrl+N**].



This will bring up the [Create New Category](#) dialog, Enter the name and any relevant details into the text field as well as the name of the author before pressing the **OK** button. New topics can now be added to the category, refer to section, [Adding a New Topic](#).

| Control | Description |
|--|---|
| New Category | Creates a new category. |
| New Topic | Creates a new topic under the selected category in the tree. |
| Edit | Edits the currently select item, either a category, topic or post. |
| Copy Path To Clipboard | Will copy the path to the currently selected item to the clipboard. |
| Refresh Category | Refreshes the currently selected category. |
| Reload Discussion Forum | Reload the discussion forum. Used when multiple users are connected to a model on a server. |

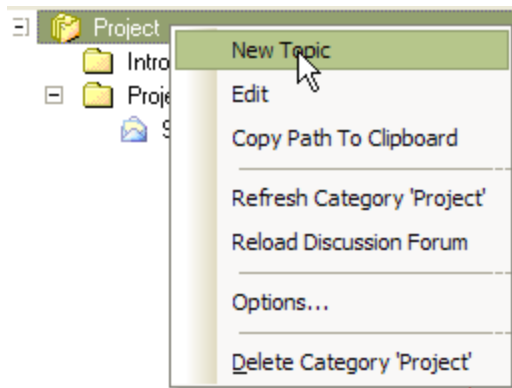
| | |
|-------------------------|-------------------------------------|
| Options | Brings up the options dialog. |
| Delete Category | Deletes the category from the tree. |

See Also

- [Adding a New Topic](#)
- [Adding a New Post](#)
- [Replying to a Post](#)
- [Editing an Item](#)
- [Deleting an Item](#)
- [Forum Connections](#)

8.1.1.2 Adding a New Topic

To create a new *Topic*, right click on a *Category* from the message thread window. Select *New Topic* from the context menu.



This will bring up the [Create New Topic](#) dialog, Enter the name and any relevant details into the text field as well as the name of the author before pressing the **OK** button. New posts can now be added to the topic, refer to section, [Adding a New Post](#).

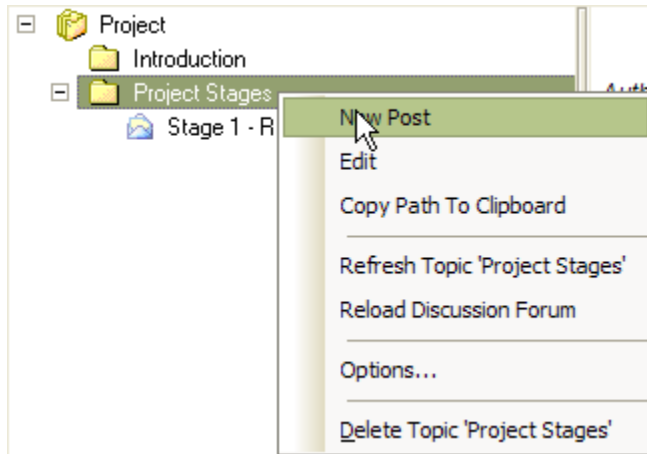
| Control | Description |
|---------------------------|---|
| New Topic | Creates a new topic under the selected category in the tree. |
| Edit | Edits the currently select item, either a category, topic or post. |
| Copy Path To Clipboard | Will copy the path to the currently selected item to the clipboard. |
| Refresh Category | Refreshes the currently selected category. |
| Reload Discussion Forum | Reload the discussion forum. Used when multiple users are connected to a model on a server. |
| Options | Brings up the options dialog. |
| Delete Category | Deletes the currently selected category from the tree. |

See Also

- [Adding a New Category](#)
- [Adding a New Post](#)
- [Replying to a Post](#)
- [Editing an Item](#)
- [Deleting an Item](#)
- [Forum Connections](#)

8.1.1.3 Adding a New Post

To create a new *Post*, right click on a *Topic* in the message thread window. Select *New Post* from the context menu.



This will bring up the [Create New Post](#) dialog. Enter the name and any relevant details into the text field as well as the name of the author before pressing the **OK** button. Users can now reply to post, refer to section, [Replying to a Post](#).

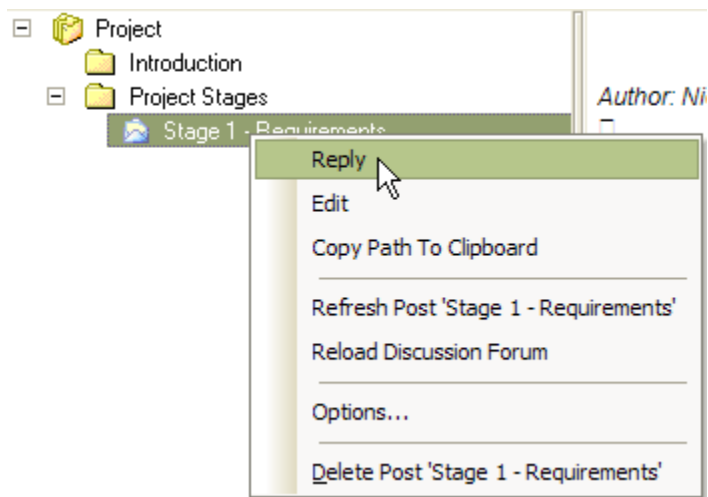
| Control | Description |
|-------------------------|---|
| New Post | Creates a new post under the selected topic in the tree. |
| Edit | Edits the currently select item, either a category, topic or post. |
| Copy Path To Clipboard | Will copy the path to the currently selected item to the clipboard. |
| Refresh Topic | Refreshes the currently selected topic. |
| Reload Discussion Forum | Reload the discussion forum. Used when multiple users are connected to a model on a server. |
| Options | Brings up the options dialog. |
| Delete Topic | Deletes the currently selected topic from the tree. |

See Also

- [Adding a New Category](#)
- [Adding a New Topic](#)
- [Replying to a Post](#)
- [Editing an Item](#)
- [Deleting an Item](#)
- [Forum Connections](#)

8.1.1.4 Replying to a Post

To reply to a post, right click on a *Post* in the message thread window. Select *Reply* from the context menu.



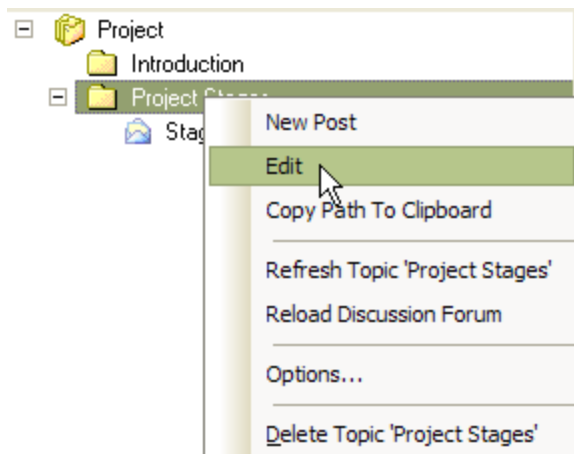
This will bring up the [Reply to Post](#) dialog. Enter the name and any relevant details into the text field as well as the name of the author before pressing the **OK** button.

See Also

- [Adding a New Category](#)
- [Adding a New Topic](#)
- [Adding a New Post](#)
- [Editing an Item](#)
- [Deleting an Item](#)
- [Forum Connections](#)

8.1.1.5 Editing an Item

To edit a *Category*, *Topic* or *Post*, right click on the item in the message thread window. Select *Edit* from the context menu or alternatively press [**Ctrl+E**].



This will bring up the *Edit* dialog. Modify any relevant details into the text field. You will be unable to edit the name of the author. Press the **OK** button.

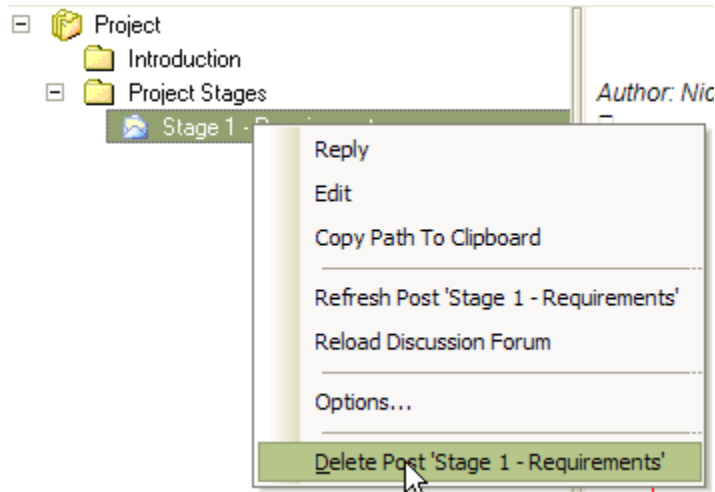
See Also

- [Adding a New Category](#)

- [Adding a New Topic](#)
- [Adding a New Post](#)
- [Replying to a Post](#)
- [Deleting an Item](#)
- [Forum Connections](#)

8.1.1.6 Deleting an Item

To delete a *Category*, *Topic* or *Post*, right click on the item in the message thread window. Select *Delete* from the context menu.



A confirmation dialog will appear, press the *OK* button.

See Also

- [Adding a New Category](#)
- [Adding a New Topic](#)
- [Adding a New Post](#)
- [Replying to a Post](#)
- [Editing an Item](#)
- [Forum Connections](#)

8.1.1.7 Forum Connections

Forum Connections enables you to access other discussion forums from other EA models or models located on servers.

By right clicking on an item in the tree and selecting *Options*, the *Forum Connections* dialog will appear.

Switching to another Discussion Forum.

1. Click on the *New* button and select the EA model in which to access the discussion form. Click *Open* and the model will appear in the *Forum Connections* window.
2. Select the model in the *Forum Connections* window.
3. Press the *Switch To Selected* button. The forum will now switch to the forum in the selected model from the list.
4. Press *OK*.

Note: If you have items in the list, just select the item from the list and press the *Switch To Selected* button. Press **OK**.

| Control | Description |
|---------------------------------------|---|
| Connection Name | The connection name will take on the name of the model you selected. |
| Connection Type | The type of EA model. It may be a local <i>.EAP</i> file or a model on a remote server. |
| Target Model | The path to the selected model. |
| New | Create a new <i>Discussion Forum</i> connection. |
| Save | Save the connection to the <i>Forum Connections</i> list. |
| Delete | Delete the currently selected connection from the <i>Forum Connections</i> list. |
| Forum Connections | List all <i>Forum Connections</i> created. |
| Always load current model forum first | Will always load the forum for the current model. |
| Always prompt for connection | With this check box enabled, each time you select the Discussion Forum from the main menu, the <i>Forum Connections</i> dialog will appear. |
| Switch To Selected | This will switch the Discussion Forum to the one selected in the <i>Forum Connections</i> list. |
| Cancel | Cancel out. |
| OK | Press OK when finished. |

See Also

- [Adding a New Category](#)
- [Adding a New Topic](#)
- [Adding a New Post](#)
- [Replying to a Post](#)
- [Editing an Item](#)
- [Deleting an Item](#)

8.1.2 Message Dialog

The project discussion forum message dialogs (*Create New Category*, *Create New Topic*, *Create New Post* and *Reply to Post*) all share the same functionality.

The screenshot shows a dialog box with the following elements:

- Name:** EA Version 6.0
- Author:** Geoffrey Sparks
- Rich Text Editor:** Includes icons for Bold (B), Italic (I), Underline (U), Text Color (ABC), Background Color (A), Bulleted List, Numbered List, and Link.
- Text Content:**

Welcome to the Sparx Systems Discussion Forum

The Sparx Systems Discussion Forum can be used to discuss project development. With the Forum users can:

 - Create New Categories and Posts
 - Create New Posts and Reply to Posts
 - Format your messages
 - Link EA elements to the Discussion
- Buttons:** OK, Cancel, Help

The table below describes the operation of each option available for the dialogs.

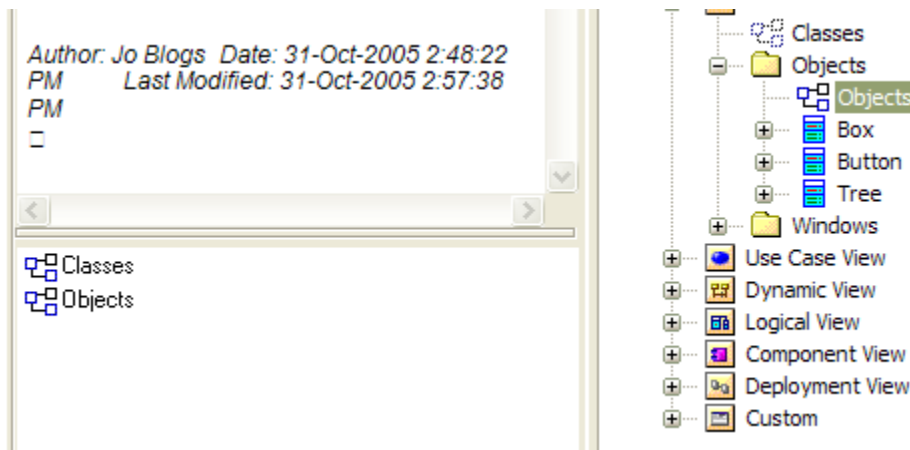
| Control | Description |
|------------------|--|
| Name | The name of the message category or message topic. |
| Author | Use this drop down list to select the message author or type in a new Author if the Author name is not present in the list. The Authors in the drop down list are defined in the model Authors list. For more information see Model Authors topic. |
| Formatting Tools | These are standard formatting options for text |
| OK | Confirm the forum message |
| Cancel | Cancel the forum message |
| Help | Brings up the help topic. |

8.1.3 Adding Element Links

The project discussion forum can have element links associated with message threads. This allows for the rapid navigation to the objects in Project View, access to the elements properties, and with diagrams the ability to open the diagram directly from the forum. Elements can be associated with the discussion forum message by dragging the element from the *Project View* into the linked elements section of the project discussion forum. To access the navigations options of each element in the linked elements section, right click

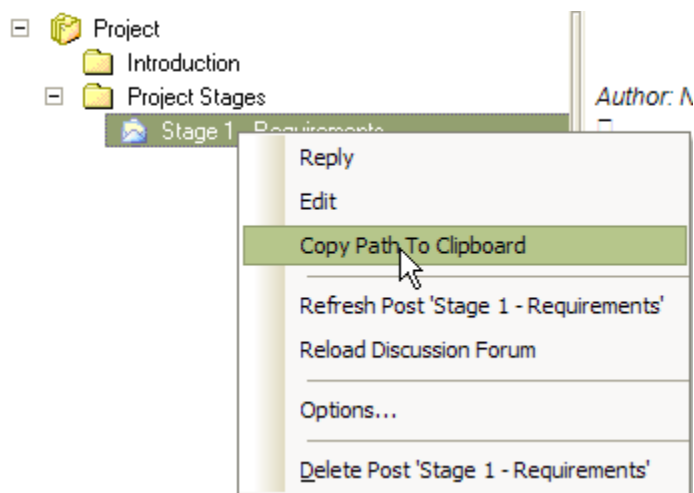
on the object to bring up the navigation context menu. The options are detailed in the table below.

| | |
|-------------|--|
| Open | Opens the diagram. |
| Properties | Bring up the element properties for the selected element. |
| Usage | Shows the usage of the element |
| Delete Link | Deletes the association between the message and the element. |



8.1.4 Copy Path to Clipboard

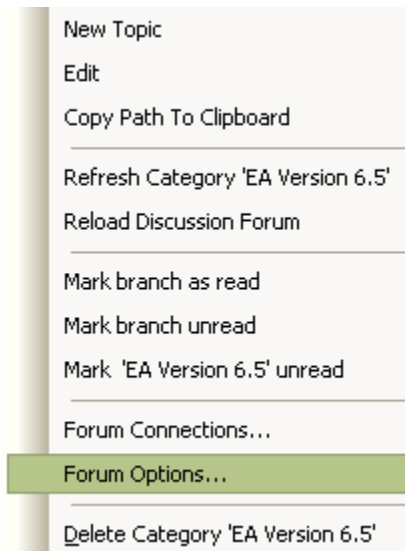
The current path in the discussion forum tree may be copied to the clipboard. Right click on any element in the tree and select *Copy Path To Clipboard* from the context menu or press [Ctrl+C].



The clipboard will now contain the path to the selected item in the tree. For the above example, the clipboard will contain, *EA Version 6.5::Help File::MOF::RE:MOF*.

8.1.5 Context Menu

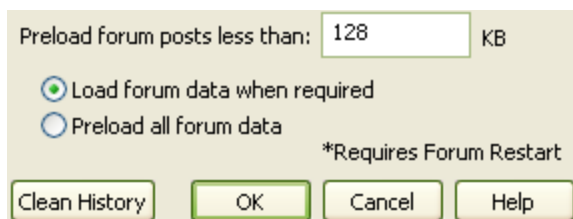
The Discussion Forum Context Menu allows you access to the following functions.



| | |
|--|--|
| New Topic | Adds a New Topic to the forum |
| Edit | Allows Editing of an Item |
| Copy Path to Clipboard | Copies the Path to the current selected post to the clipboard |
| Refresh Category 'xyz' | Refreshes the named category, getting new posts and topics |
| Reload Discussion Forum | Reloads the entire Discussion Forum, getting new posts and topics |
| Mark branch as Read | Marks this topic and all sub-topics and posts as read |
| Mark branch as Unread | Marks this topic and all sub-topics and posts as unread |
| Mark 'xyz' unread | Marks this topic as unread |
| Forum Connections... | Forum Connections enables you to access other discussion forums from other EA models or models located on servers. |
| Forum Options... | The Forum Options dialog allows you to change the loading behaviour of the forum. |
| Delete Category 'xyz' | Delete this topic and all sub-posts and sub-topics |

8.1.6 Forum Options

The Forum Options dialog allows gives you the ability to change the loading behaviour of the forum.



From here you can:

- **Clean History.** The history keeps track of which posts you have read.
- **Load forum data when required.** This is the fastest loading option. Forum data will only be loaded on demand, for example when you read a post.
- **Preload all forum data.** This will cache the entire contents of the forum on load. This will take longer to load but once completed, maneuvering the forum will be faster.

8.2 EA Project Files

An Enterprise Architect project is stored in a *data repository*. In EA Desktop and Professional editions, you work with a single file having a *.EAP* extension. A suitable *DBMS* database can be used for project files in EA Corporate edition.

Project Files

.EAP Files

A single file with a *.EAP* extension is used to store projects in EA Desktop and Professional editions. An *.EAP* file is a Microsoft JET database, so can also be opened using MS Access or any other reporting tool that can work with JET databases.

DBMS Repositories

A suitable *DBMS* database can be used for model files in EA Corporate edition. *DBMS* project files have the same logical structure as *.EAP* model files, but must be connected to using ADO/ODBC. EA currently supports the following data repositories:

- MS Access (in all editions - *.EAP* files are stored in Microsoft JET databases as mentioned above)
- [SQL Server](#)
- [MySQL](#)
- [Oracle 9i and 10g](#)
- [PostgreSQL](#)
- [MSDE](#)
- [Adaptive Server Anywhere](#)
- [Progress OpenEdge](#)

Whenever you launch Enterprise Architect, the first thing you will see is the [Start Page](#) (shown below). From here, you can [create a new project](#), [open a project](#) and [connect to a data repository](#) (Corporate Edition only).

Create a New Project File

On creating a [new project](#), the [Model Wizard](#) will enable you to select various model packages.

Open an Existing Project

There are various ways to [open a projects](#) in Enterprise Architect. New users are advised to explore the [EA Example file](#) supplied with Enterprise Architect.

Connect to a Data Repository

Note: *This feature is available in the Corporate edition only.*

Enterprise Architect allows you to connect to a data repository. EA currently supports the following data repositories:

- MS Access (in all editions - *.EAP* files are stored in Microsoft JET databases)
- [SQL Server](#)
- [MySQL](#)
- [Oracle 9i and 10g](#)
- [PostgreSQL](#)
- [MSDE](#)
- [Adaptive Server Anywhere](#)
- [Progress OpenEdge](#)

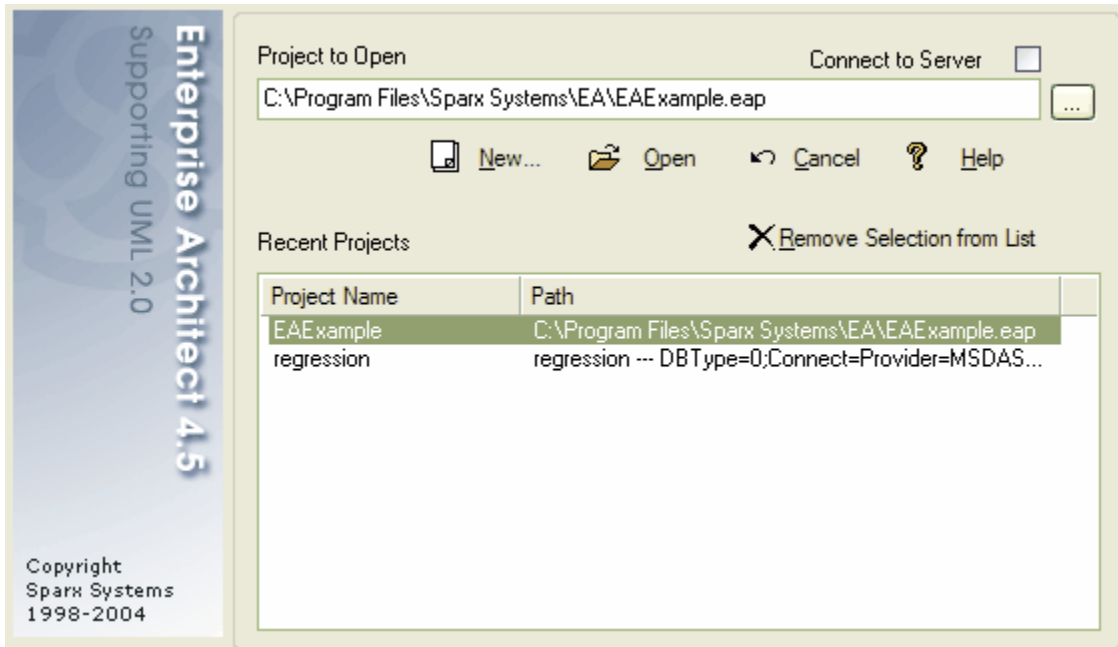
To create a new data repository, you first need to create a new database with the *DBMS* management software, then run supplied scripts to create the logical structure. You should then use EA data transfer functions to move a model from an *.EAP* or *DBMS* model into the new model.

8.2.1 Open a Project

Enterprise Architect support several different methods to open an EA project file.

From the Main Menu

Select **File | Open Project** from the main menu. From the **Open Project** dialog, select the path for the file to open then click the **Open**.



From the Start Page (see Start Page)

- Click on **Open a Project** File.
- The **Open Project** dialog will appear.
- Use the file browser to navigate to the project you wish to open having a file extension (***.EAP**) . Select the project and click **Open**.

Recently Opened Projects

Enterprise Architect keeps a list of recently opened projects and displays them on the Start Page for easy selection. If the project you wish to open is in the Recently Opened Projects list, simply click once on the name of the project to open it.

Note: If you already have a project open, you will be prompted to save changes before loading.

EA Example Project File

New Enterprise Architect users in particular should start by exploring the EAExample file supplied with Enterprise Architect. The example model file is stored in your EA installation directory. The default installation directories, depending on which version you have installed, are:

- Registered version: C:\Program Files\Sparx Systems\EA
- Trial version: C:\Program Files\Sparx Systems\EA Trial
- Lite version: C:\Program Files\Sparx Systems\EA Lite

Connecting to a Data Repository

Note: This feature is available in the Corporate edition only.

If you are using EA Corporate edition, you also have the option to connect to [SQL Server](#), [MySQL](#), [Oracle 9i and 10g](#) and [Progress OpenEdge](#) data repositories.

See Also

- [Start Page](#)
- [Connecting to a Data Repository](#)

8.2.2 Create a New Project

Enterprise Architect projects can be created via the main menu, select *File | New Project* to invoke the *Save New Enterprise Architect Project* dialog. Select a directory and enter a file name for your project then click the *Save* button. Once the project has been saved, the [Model Wizard](#) will appear. A selection of models will be available. Check the models to include and click the *OK* button.

Alternatively new projects can be created from the [Start Page](#), select *Create a New Project*.

The Model Wizard

The Model Wizard is used to add a selection of models to the project.

The EABase Project File

The default model file - EABase.EAP - is supplied when you install Enterprise Architect. By default, the example model file is stored in your EA installation directory. The default installation directories, depending on which version you have installed, are:

- Registered version: C:\Program Files\Sparx Systems\EA
- Trial version: C:\Program Files\Sparx Systems\EA Trial
- Lite version: C:\Program Files\Sparx Systems\EA Lite

Designing a Custom Template

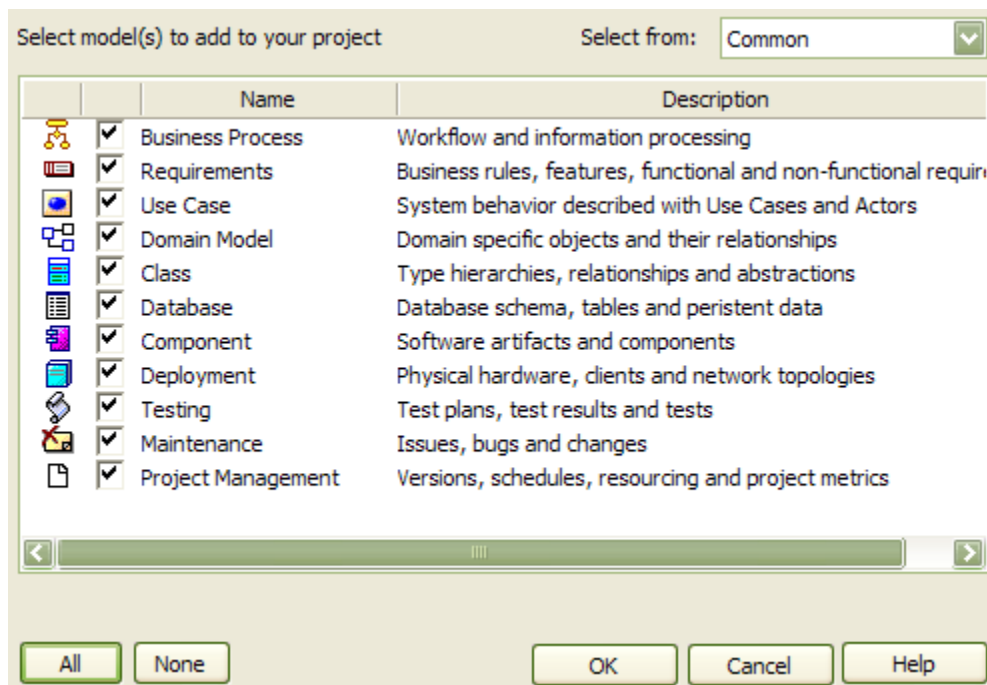
You can customize any Enterprise Architect project and use it as the base for the new project. This allows you or your organization to build a default project with company standards, tutorials, frameworks or any other common piece of modeling already in-built. A model project is no different to an ordinary project - Enterprise Architect simply copies and renames it as a starter for your new project. With careful planning you can save yourself many hours of work at project start-up.

See Also

- [Model Wizard](#)
- [Start Page](#)

8.2.2.1 Model Wizard

The Model Wizard is available on creating a new project. Once you have created a project. The Model Wizard can also be accessed from the Project View, right click on a root node and select, *Add Model From Pattern*.



| Control | Description |
|----------------|---|
| Selection From | |
| All | Select all of the models. |
| None | Clear all models selected. |
| OK | Press OK to create the Project Tree for your project. |
| Cancel | Press cancel for a blank project tree. |
| Help | Take you to the related help topic. |

Note: Once a project has been created, the model wizard can be accessed via the Project View. Right click on a root node and select *Add Model From Pattern*.

See Also

- [Quick Start](#)
- [Model Patterns](#)

8.2.3 Setting Up a Database Repository

Introduction

The Corporate Edition of EA allows you to work with database repositories rather than use the standard .EAP files. Available repositories are **SQL Server**, **MySQL**, **Oracle 9i and 10g**, **PostgreSQL**, **MSDE** and **Adaptive Server Anywhere**. Setting up a database repository is a two or three step process: Firstly, you set up an ODBC driver for your database; secondly, you create the repository tables using scripts downloaded from the Sparx Systems web site; and finally, you connect to the repository. Full instructions on all three steps are given below.

Setting Up an ODBC Driver

Setting up an ODBC driver is only necessary for **MySQL**, **PostgreSQL** and **Adaptive Server Anywhere**. The other supported databases connect using OLE DB, so this step can be skipped. To find out how to set up an ODBC driver, go to:

- [Setup a MySQL ODBC Driver.](#)
- [Setup a PostgreSQL ODBC Driver.](#)
- [Setup an Adaptive Server Anywhere ODBC Driver.](#)

- [Setup a Progress OpenEdge ODBC Driver.](#)

Creating a Repository

To find out how to download the scripts and create the data repository tables, go to:

- [Create a new MySQL Data Repository.](#)
- [Create a new SQL Server Data Repository.](#)
- [Create a new Oracle 9i and 10g Data Repository.](#)
- [Create a new PostgreSQL Data Repository.](#)
- [Create a new Adaptive Server Anywhere Data Repository.](#)
- [Create a new MSDE Server Data Repository.](#)
- [Create a new Progress OpenEdge Data Repository](#)

Connecting to a Repository

Once the repository is created, you can connect to it. To find out how, go to:

- [Connect to a MySQL Data Repository.](#)
- [Connect to a SQL Server Data Repository.](#)
- [Connect to an Oracle 9i and 10g Data Repository.](#)
- [Connect to a PostgreSQL Data Repository.](#)
- [Connect to an Adaptive Server Anywhere Data Repository.](#)
- [Connect to a MSDE Server Data Repository.](#)
- [Connect to a Progress OpenEdge Data Repository.](#)

8.2.3.1 Setting Up an ODBC Driver

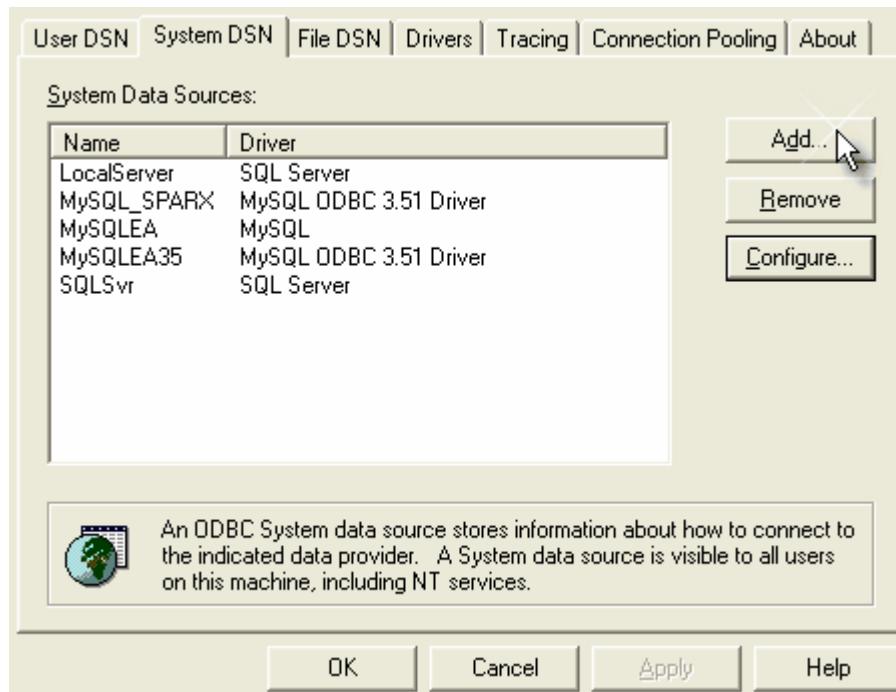
This section details how to set up the following ODBC drivers:

- [MySQL ODBC Driver](#)
- [PostgreSQL ODBC Driver](#)
- [Adaptive Server Anywhere ODBC Driver](#)

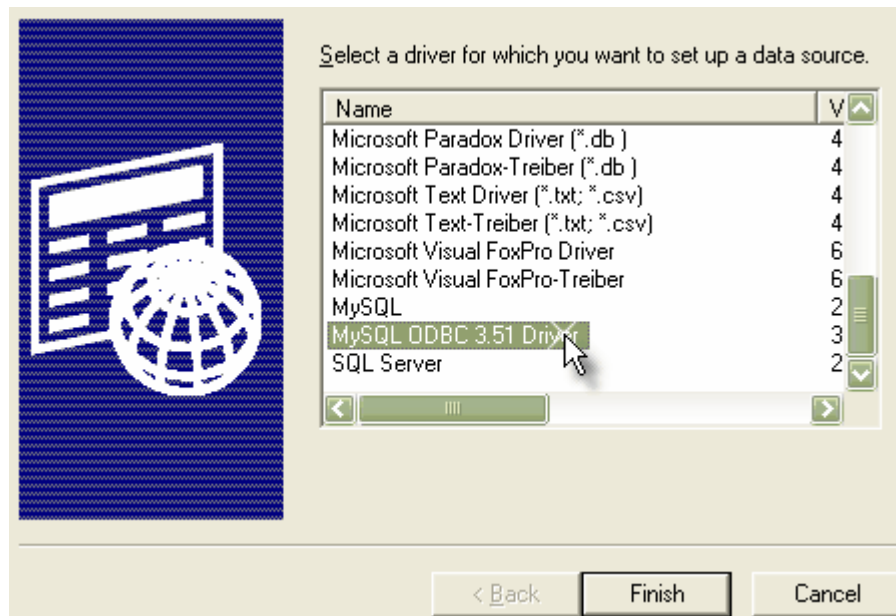
8.2.3.1.1 Setup a MySQL ODBC Driver

Before you can connect to a MySQL data repository, you must first set up a MySQL ODBC driver. To do this, you will require Microsoft MDAC components, MySQL DBMS system and MySQL ODBC driver (version 3.51 minimum). Follow the steps below to set up your MySQL ODBC Driver:

1. Open the *Control Panel* | *Administrative Tools* | *Data Sources (ODBC)* window (below).



2. Press **Add** - this will bring up the dialog below, which allows you to add a new DSN.



3. Select "MySQL ODBC 3.51 Driver" from the list.
4. Press **Finish**. This will open the DSN Configuration dialog.
5. Enter the following configuration details:
 - A name for the connection
 - Description (optional)
 - The host address of the DBMS server

- The Database Name on the selected server
- User name and Password

See the example below:

This dialog helps you in configuring the ODBC Data Source Name, that you can use to connect to MySQL server

DSN Information

Data Source Name:

Description:

MySQL Connection Parameters

Host/Server Name(or IP):


Database Name:

User:

Password:

Port (if not 3306):

SQL command on connect:



6. Press the *Options* button to set the advanced options.
7. Ensure that the *Don't Optimize Column Width* and *Return Matching Rows* items are checked, then press the *OK* button.

Options that affect the behaviour of Driver (Commonly used)

Don't Optimize Column Width Use Compressed Protocol

Return Matching Rows Change BIGINT Columns to INT

Allow BIG Results Safety (Check this if you have problems)

Miscellaneous options (Rarely used)

Don't Prompt on Connect Return Table Names in SQLDescribeCol

Enable Dynamic Cursor Ignore Space After Function Names

Ignore # in #.Table Force Use of Named Pipes

Use Manager Cursors No Catalog (exp)

Don't Use Setlocale Read Options From C:\my.cnf

Pad CHAR to Full Length Disable Transactions

Don't Cache Results(only for forward-only cursor) Force use of Forward-only cursors

Driver Trace Options (Debug version only)

Trace driver calls (c:\myodbc.log) Save queries to log file (c:\myodbc.sql)

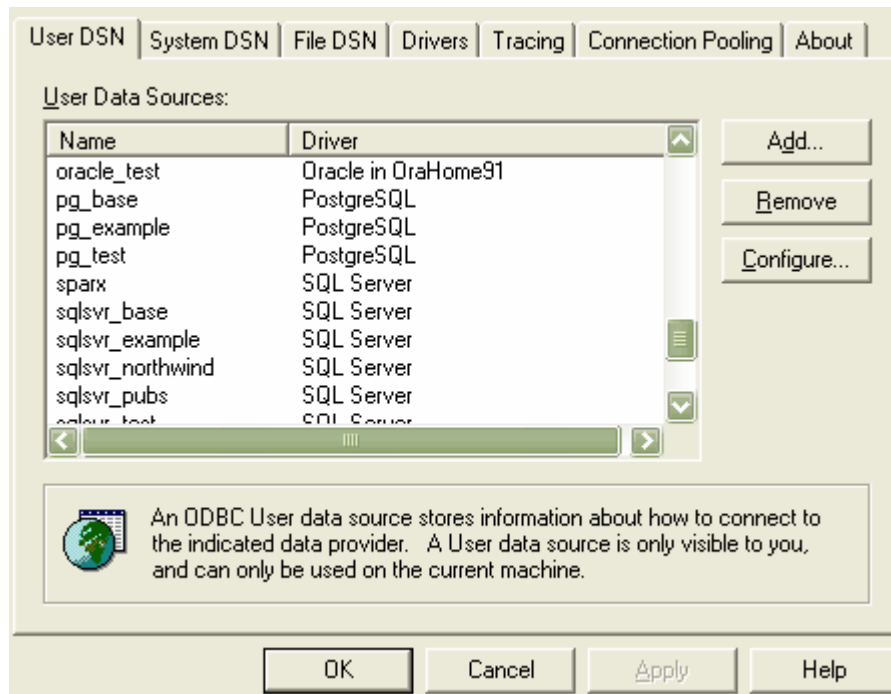
8. Press **Test Data Source** to confirm that the details are correct.
9. If the test succeeds, press **OK** to complete the configuration.
10. If the test does not succeed, review your settings.

Your MySQL connection is now available to use in Enterprise Architect.

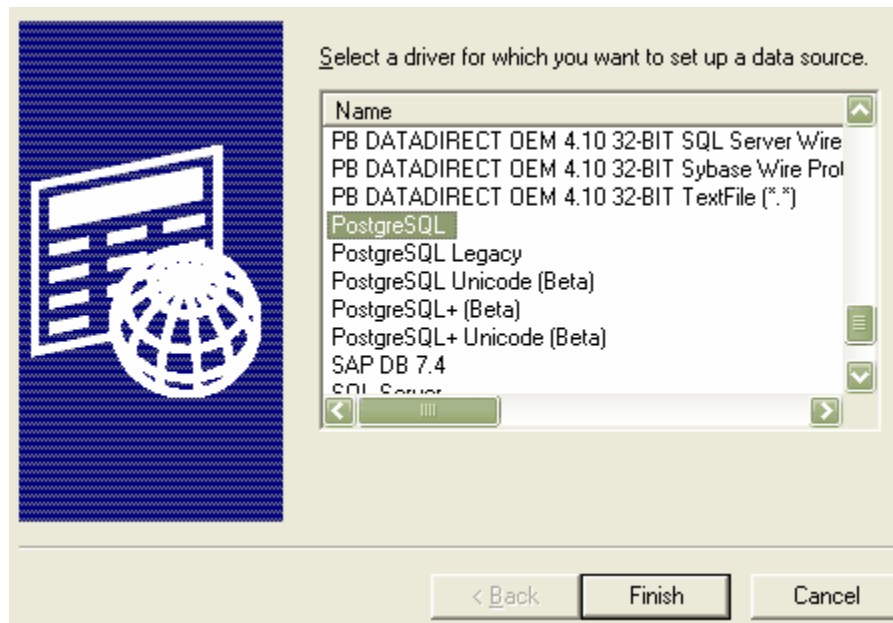
8.2.3.1.2 Setup a PostgreSQL ODBC Driver

Before you can connect to a PostgreSQL data repository, you must first set up a PostgreSQL ODBC driver. To do this, you will require Microsoft MDAC components, PostgreSQL DBMS system and PostgreSQL ODBC driver (version 7.03.01.00 minimum). Follow the steps below to set up your PostgreSQL ODBC Driver:

1. Open the **Control Panel | Administrative Tools | Data Sources (ODBC)** window (below).



2. Press the **Add** button - this will bring up the dialog below, which allows you to add a new DSN.



3. Select "PostgreSQL" from the list.
4. Click the *Finish* button.
5. Enter the following configuration details:
 - A name for the connection
 - The actual name of the database.
 - Description (optional)
 - The host address of the PostgreSQL server.
 - User name and password.

Data Source: Description:

Database:

Server: Port:

User Name: Password:

Options:

6. Click the *Data Source* button and set the options on Pages 1 and 2 as shown on the examples below:

Page 1 Page 2

Disable Genetic Optimizer ConnLog (C:\psqlodbc_xxxx.log)

KSQL (Keyset Query Optimization) Parse Statements

Recognize Unique Indexes Cancel as FreeStmt (Exp)

Use Declare/Fetch MyLog (C:\mylog_xxxx.log)

Unknown Sizes

Maximum Don't Know Longest

Data Type Options

Text as LongVarChar Unknowns as LongVarChar Booleans as Char

Miscellaneous

Max Varchar: 1024 Max LongVarChar: 1000000

Cache Size: 100 SysTable Prefixes: dd;

OK Cancel Apply Defaults

Page 1

Read Only Row Versioning

Show System Tables Disallow Premature

LF <-> CR/LF conversion True is -1

Updatable Cursors Server side prepare

bytea as LO

Int8 As

default bigint numeric varchar double int4

Protocol

7.X,6.4+ 6.3 6.2

OID Options

Show Column Fake Index

Connect Settings:

OK Cancel Apply

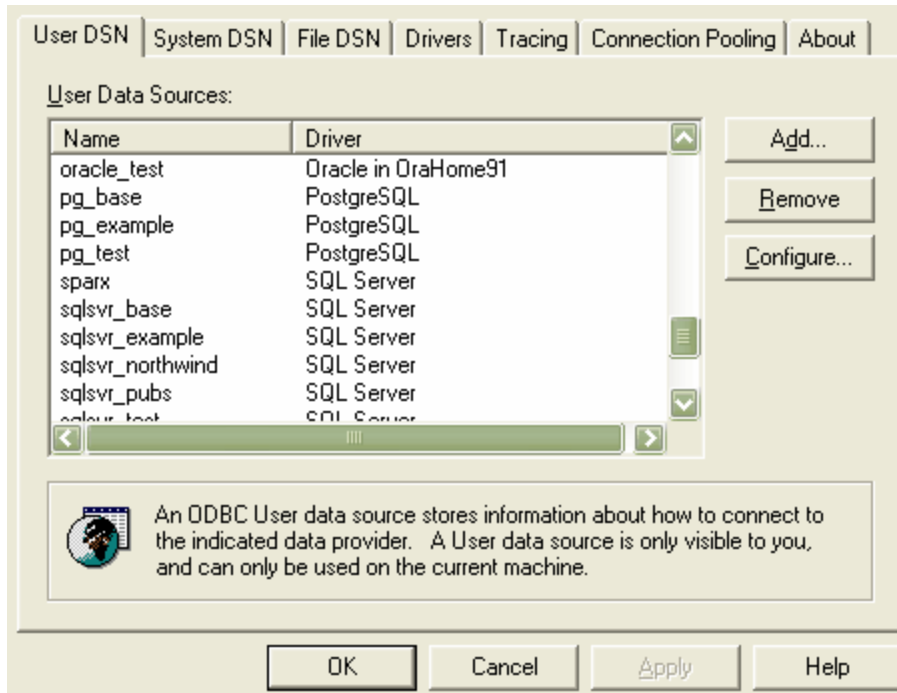
7. Press **OK** to complete the configuration.

Your PostgreSQL connection is now available to use in Enterprise Architect.

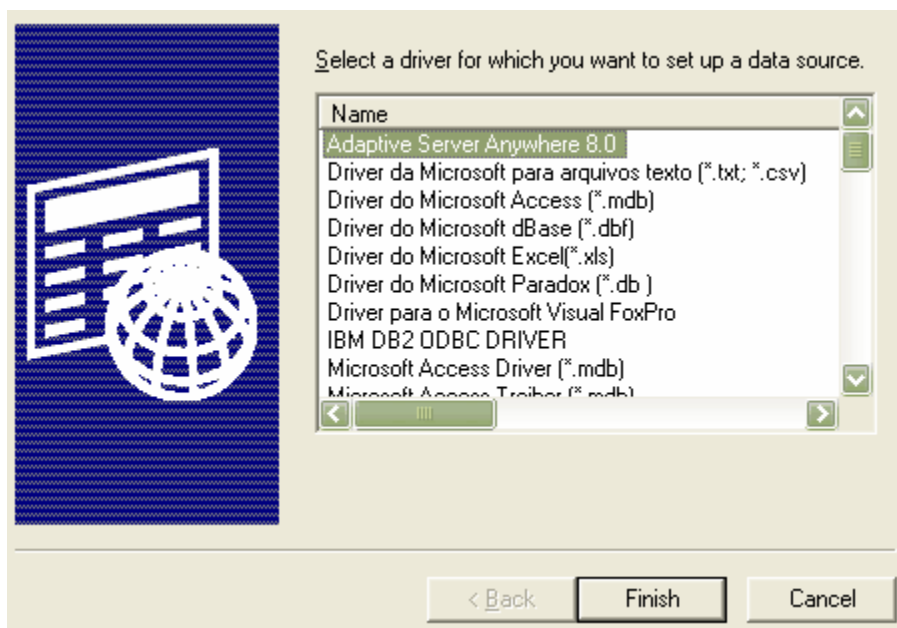
8.2.3.1.3 Setup an Adaptive Server Anywhere ODBC Driver

Before you can connect to a ASA data repository, you must first set up a ASA ODBC driver. To do this, you will require Microsoft MDAC components, the ASA DBMS system and the ASA ODBC driver (installed with the ASA DBMS). Follow the steps below to set up your ASA ODBC Driver:

1. Open the *Control Panel | Administrative Tools | Data Sources (ODBC)* window (below).



2. Press the *Add* button - this will bring up the dialog below, which allows you to add a new DSN.



3. Select "Adaptive Server Anywhere 8.0" from the list.
4. Click the *Finish* button.
5. Enter the following configuration details:

A name for the connection on the ODBC tab.

The screenshot shows the 'Advanced' tab of the ODBC configuration dialog box. The 'Data source name' field contains 'asa_base_model'. The 'Description' field is empty. The 'Isolation level' field is empty. There are five unchecked checkboxes: 'Microsoft applications (Keys in SQLStatistics)', 'Delphi applications', 'Suppress fetch warnings', 'Prevent driver not capable errors', and 'Delay AutoCommit until statement close'. The 'Describe Cursor Behavior' section has three radio buttons: 'Never', 'If required' (which is selected), and 'Always'. The 'Translator' field contains '<No Translator>'. There are buttons for 'Select Translator...', 'Test Connection', 'OK', and 'Cancel'.

The username and password on the Login tab (dba, sql - are the defaults when ASA is installed).

The screenshot shows the 'Login' tab of an ODBC configuration dialog. The 'Database' sub-tab is selected. Two radio buttons are present: 'Use integrated login' (unselected) and 'Supply user ID and password' (selected). Below these, there are two text input fields: 'User ID:' containing 'dba' and 'Password:' containing '***'. A checkbox labeled 'Encrypt password' is located below the password field and is currently unchecked. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

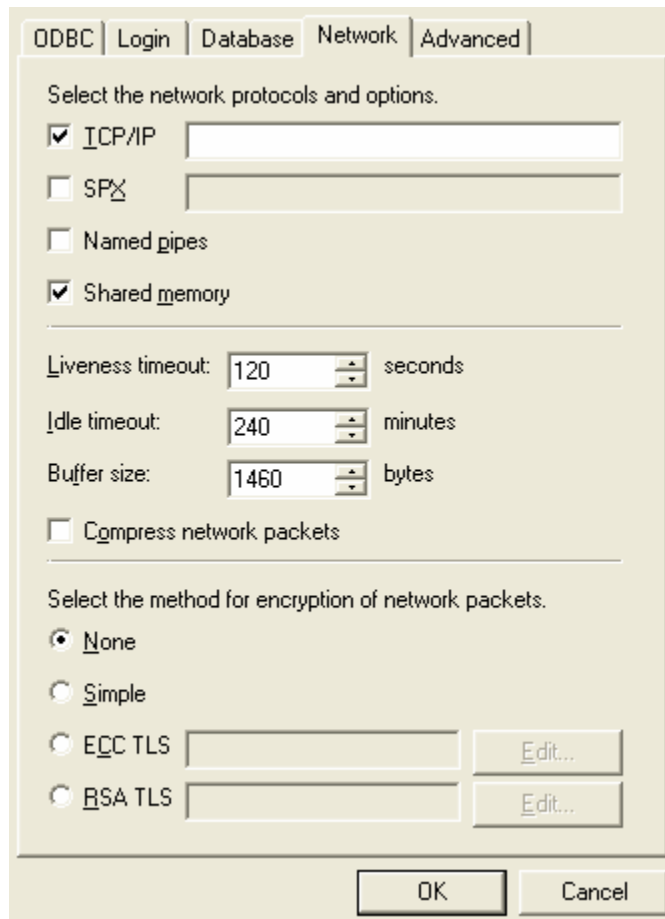
The server name and the path to the database on the Database tab.

The image shows a screenshot of the ODBC configuration dialog box, specifically the Database tab. The dialog has five tabs: ODBC, Login, Database, Network, and Advanced. The Database tab is active. It contains the following fields and options:

- Server name:** asa_localserver
- Start line:** (empty text box)
- Database name:** (empty text box)
- Database file:** \\Server\asa8\ea_base.db
- Browse...** button (next to the Database file field)
- Encryption key:** (empty text box)
- Automatically start the database if it isn't running
- Automatically shut down database after last disconnect

At the bottom of the dialog are two buttons: OK and Cancel.

The network protocol on the Network tab (if the database is on a network location).



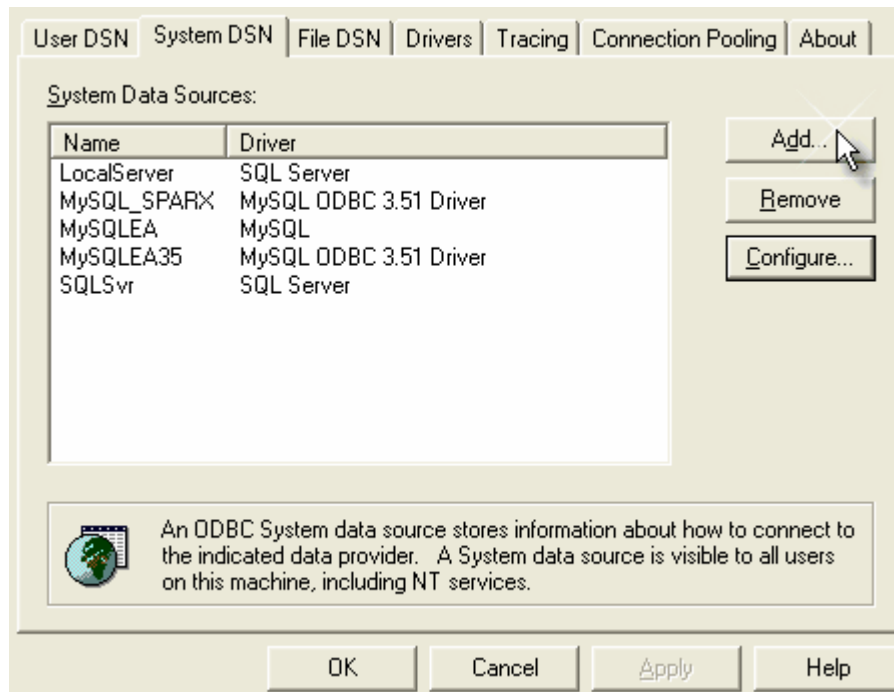
6. You can now return to the ODBC tab and test the connection.
7. Press **OK** to complete the configuration.

Your Adaptive Server Anywhere connection is now available to use in Enterprise Architect.

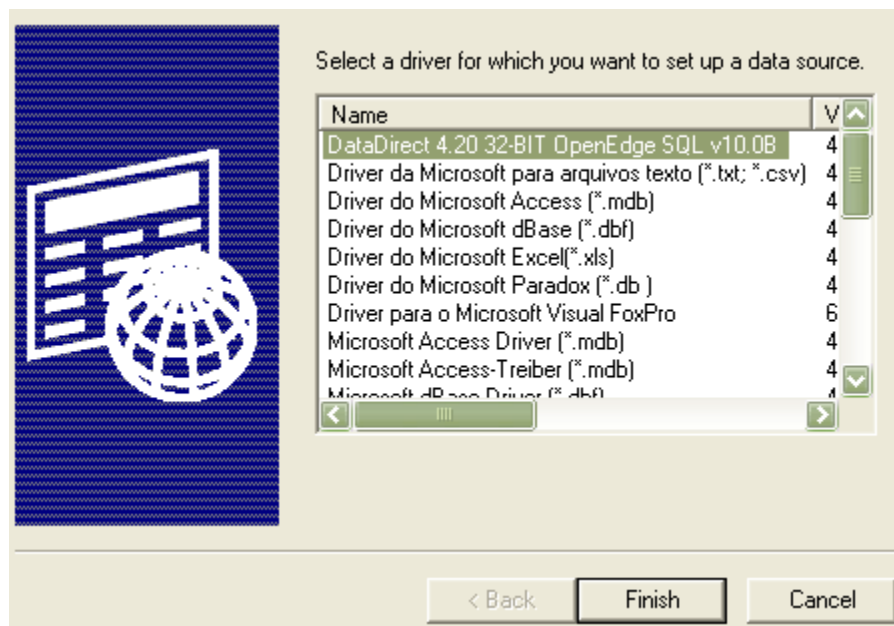
8.2.3.1.4 Setup a Progress OpenEdge ODBC Driver

Before you can connect to a OpenEdge data repository, you must first set up an OpenEdge ODBC driver. To do this, you will require Microsoft MDAC components, OpenEdge DBMS system and DataDirect ODBC driver for OpenEdge (version 4.20 minimum). Follow the steps below to set up the ODBC Driver:

1. Open the **Control Panel | Administrative Tools | Data Sources (ODBC)** window (below).



2. Press **Add** - this will bring up the dialog below, which allows you to add a new DSN.



3. Select the DataDirect/OpenEdge SQL driver from the list.
4. Press **Finish**. This will open the DSN Configuration dialog.
5. Enter the following configuration details:
 - The Data Source Name
 - Description (optional)

- The host address of the DBMS server and Port Number
- The Database Name on the selected server
- The User ID

See the example below:

The screenshot shows a dialog box with three tabs: 'General', 'Advanced', and 'About'. The 'General' tab is active. It contains several input fields and buttons. The 'Data Source Name' field contains 'openedge_users'. The 'Description' field is empty. The 'Host Name' field contains 'dbserver02'. The 'Port Number' field contains '4141'. The 'Database Name' field contains 'ea'. The 'User ID' field contains 'oe_user'. There is a 'Help' button to the right of the 'Data Source Name' field. At the bottom of the dialog, there are four buttons: 'Test Connect', 'OK', 'Cancel', and 'Apply'.

8. Press **Test Connect** to confirm that the details are correct.
9. If the test succeeds, press **OK** to complete the configuration.
10. If the test does not succeed, review your settings.

Your OpenEdge connection is now available to use in Enterprise Architect.

8.2.3.2 Connecting to a Data Repository

This section details how to connect to the following data repositories:

- [MySQL Data Repository](#)
- [SQL Server Data Repository](#)
- [Oracle 9i and 10g Data Repository](#)
- [PostgreSQL Data Repository](#)
- [Adaptive Server Anywhere Data Repository](#)
- [MSDE Server Data Repository](#)
- [Progress OpenEdge](#)

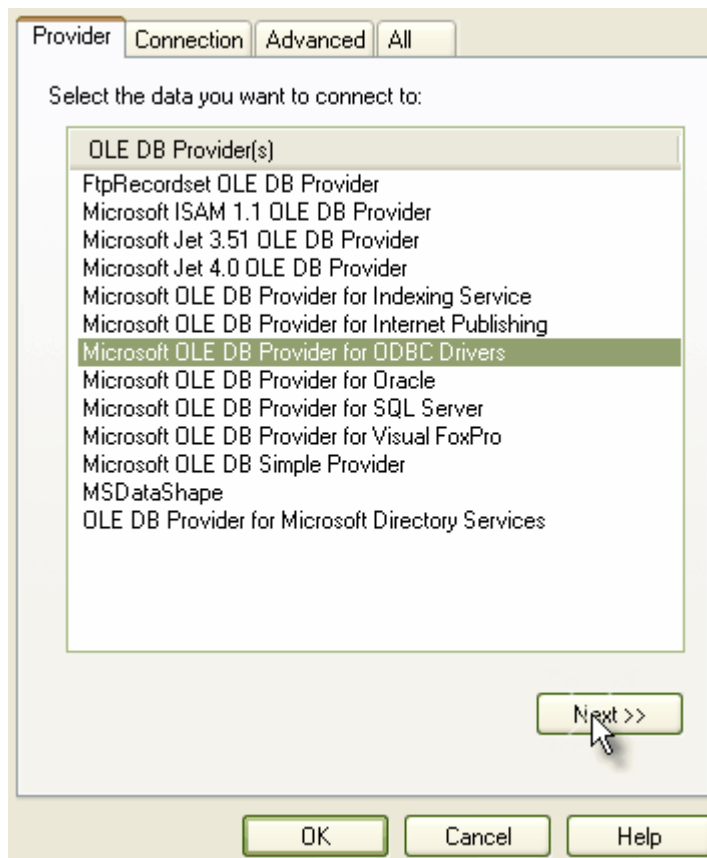
8.2.3.2.1 Connect to a MySQL Data Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

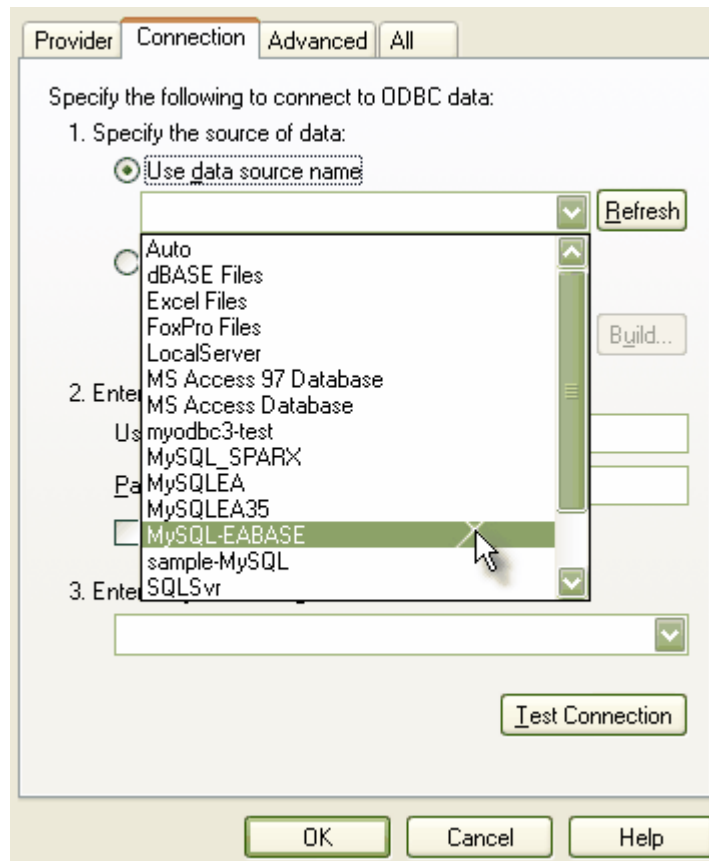
In order to use a MySQL data repository, you must connect to it in EA first. Before connecting to the

repository, you will need to have [set up a MySQL ODBC driver](#). Be aware, there are some [limitations with using MySQL](#) with EA. Follow these steps to connect to a MySQL data repository in EA:

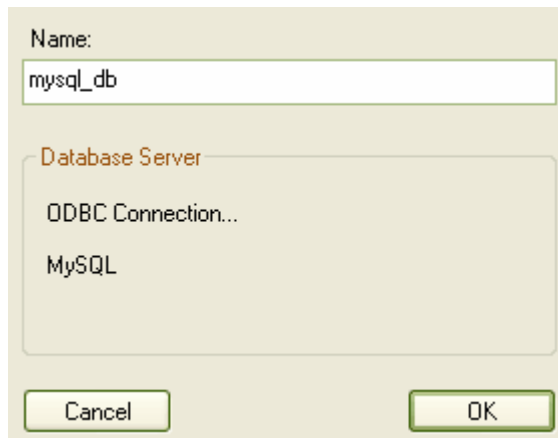
1. In the [Open Project dialog](#), check the *Connect to Server* checkbox.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have *Connect to Server* checked, the *Data Link Properties* dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press *Next*. This will take you to the *Connection* tab (below).



5. Select the ODBC driver you have set up to connect to your MySQL repository from the Use data source name drop down list. In the [setup example](#) the driver title was "MySQL-EABASE".
6. Enter the *User name* and *Password* (these are optional).
7. Enter the initial catalog (this is optional).
8. Press *Test Connection* to confirm that the details are correct.
9. If the test succeeds, press *OK*.
10. If the test does not succeed, revise your settings.
11. After you have pressed OK, the Connection Name dialog will appear (below). Give the connection a suitable name so you will recognize it in the Recently Opened Project on the [open project dialog](#).



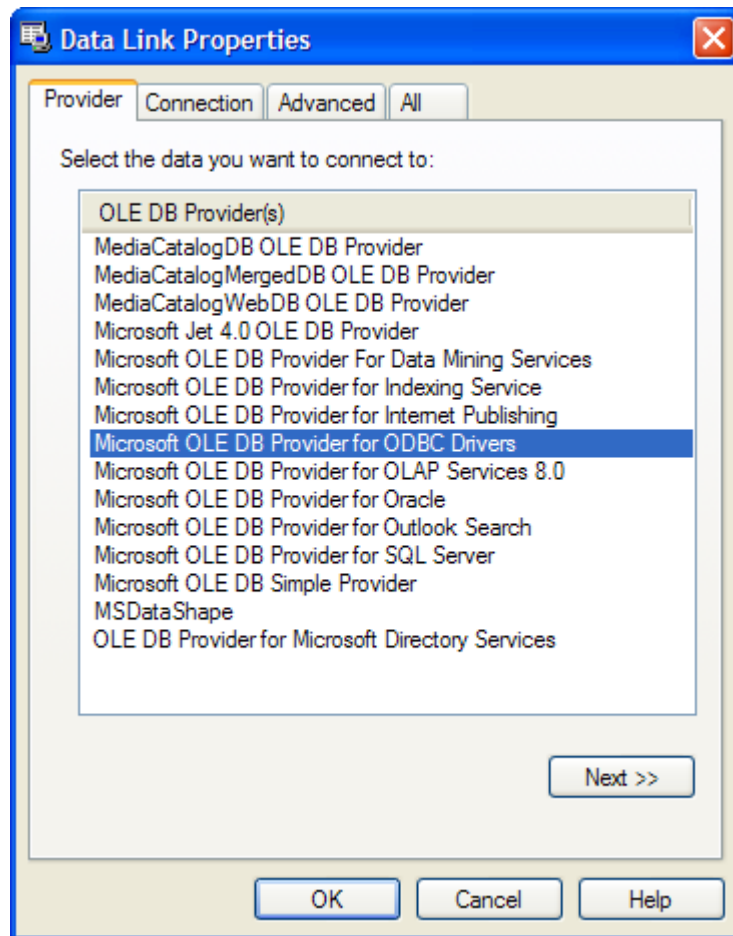
12. Press **OK** to complete the configuration.

8.2.3.2.2 Connect to a SQL Server Data Repository (Corporate Edition only)

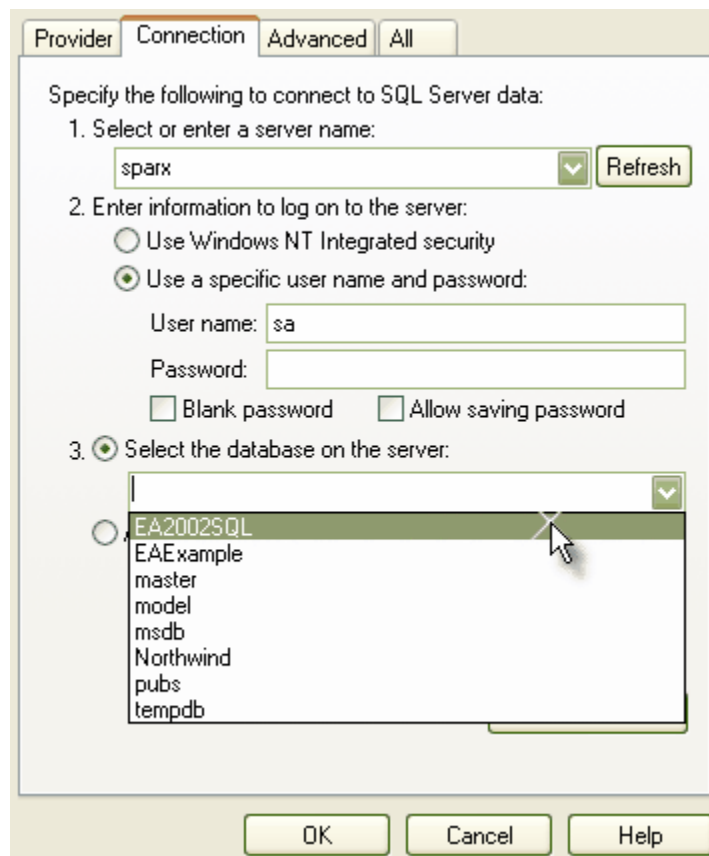
Note: This feature is available in the Corporate Edition only.

In order to use a SQL Server data repository, you must connect to it in EA first. Follow the steps below to connect to your SQL Server data repository in EA:

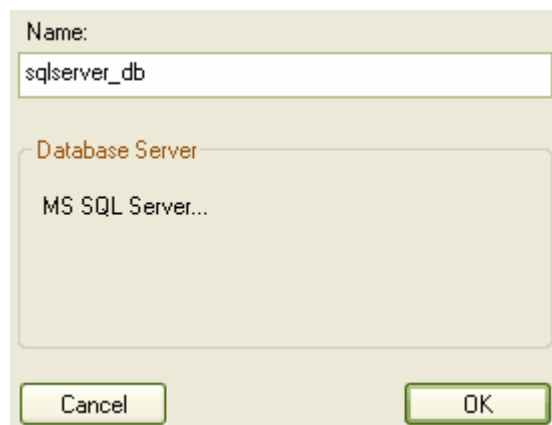
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for SQL Server" from the list.
4. Press *Next*. This will take you to the *Connection* tab (below).



5. Next enter the server details, including *Server Name*, *User Name* and *Password*.
6. From the *Select the database on the server* drop down list, select the model you wish to connect to.
7. Press *Test Connection* to confirm that the details are correct.
8. If the test succeeds, press *OK*.
9. If the test does not succeed, revise your settings.
10. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



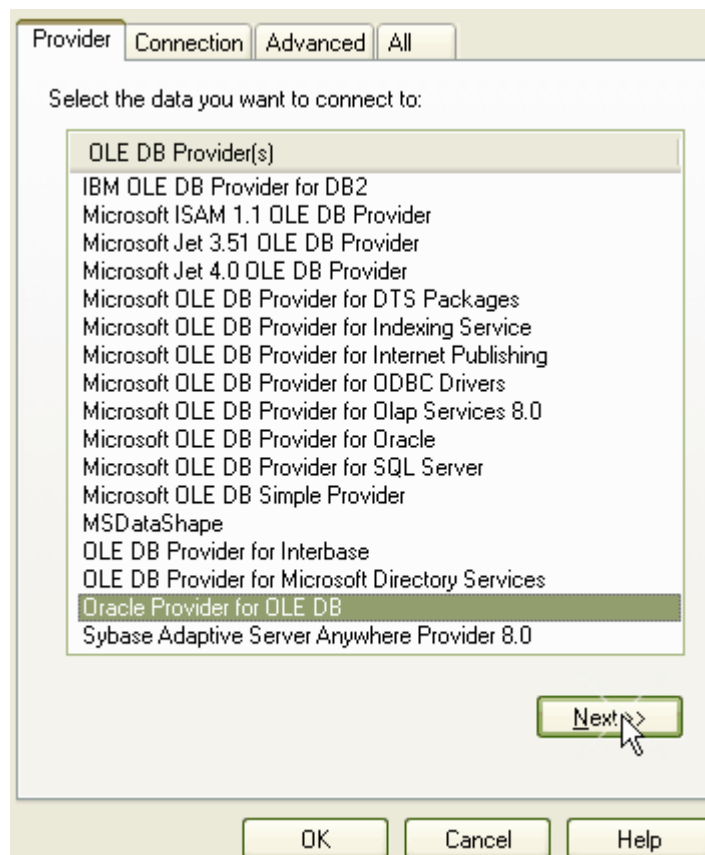
11. Press *OK* to complete the configuration.

8.2.3.2.3 Connect to an Oracle9i Data Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

In order to use a Oracle 9i and 10g data repository, you must connect to it in EA first. Follow the steps below to connect to your Oracle 9i and 10g data repository in EA:

1. In the [Open Project dialog](#), check the **Connect to Server** checkbox.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select the "Oracle Provider for OLE DB" from the list.
4. **Note:** Do not select "Microsoft OLE DB Provider for Oracle" - EA may not work as expected.
5. Press **Next**. This will take you to the **Connection** tab (below).

Specify the following to connect to this data:

1. Enter the data source and/or location of the data:
Data Source: EA
Location:
2. Enter information to log on to the server:
 Use Windows NT Integrated security
 Use a specific user name and password:
User name: scott
Password: ●●●●●●
 Blank password Allow saving password
3. Enter the initial catalog to use:

Test Connection

OK Cancel Help

6. Next enter the *Data Source* details, *User Name* and *Password*.
7. Press *Test Connection* to confirm that the details are correct.
8. If the test succeeds, press *OK*.
9. If the test does not succeed, revise your settings.
10. After you have pressed *OK*, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).

Name:
oracle_db

Database Server
Oracle Si...

Cancel OK

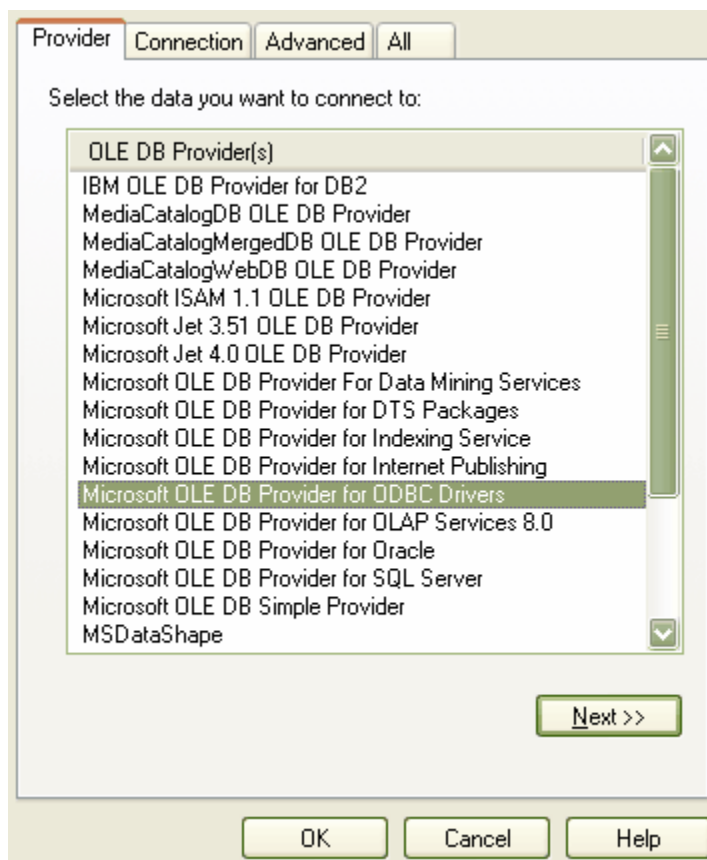
11. Press *OK* to complete the configuration.

8.2.3.2.4 Connect to a PostgreSQL Data Repository (Corporate Edition only)

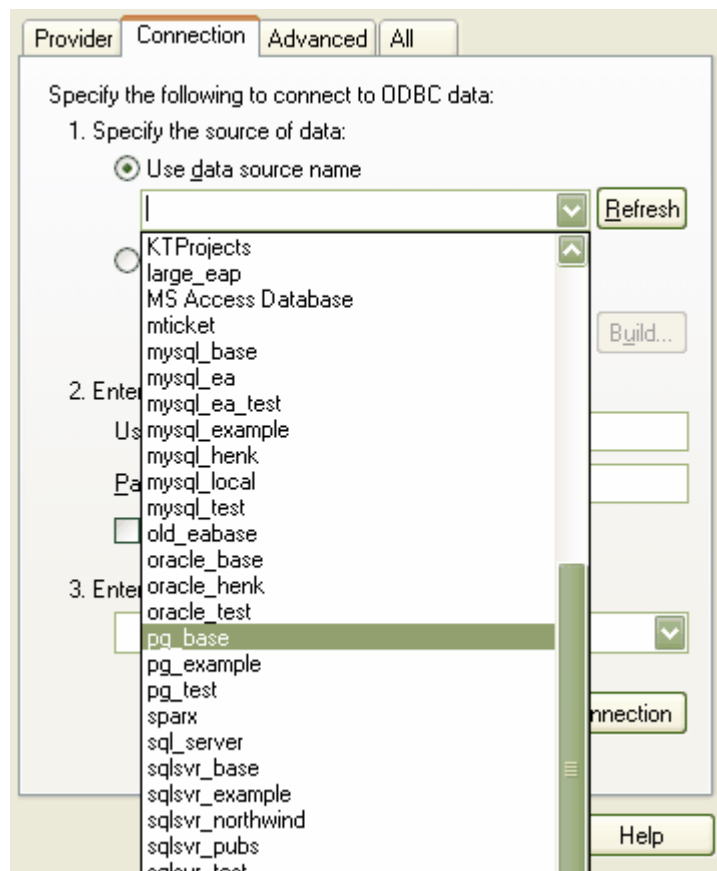
Note: This feature is available in the Corporate Edition only.

In order to use a PostgreSQL data repository, you must connect to it in EA first. Before connecting to the repository, you will need to have [set up a PostgreSQL ODBC driver](#). Follow these steps to connect to a MySQL data repository in EA:

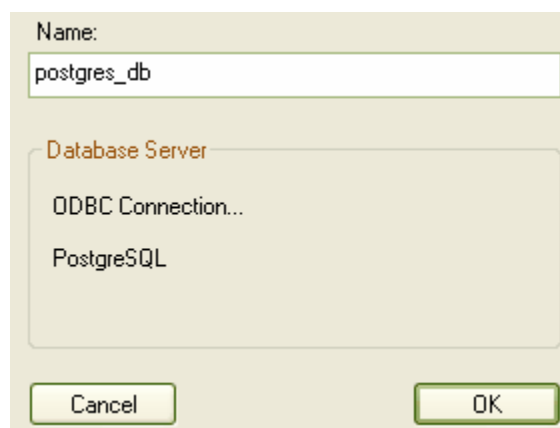
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox, or on the Start Page, click on the **Connect to Server Repository...** link.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press the **Next** button. This will take you to the **Connection** tab (below).



5. Select the ODBC driver you have set up to connect to your PostgreSQL repository from the *Use data source name* drop down list.
6. Press the *Test Connection* button to confirm that the details are correct.
7. If your test succeeded, press the *OK* button.
8. If your test did not succeed, revise your settings.
9. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



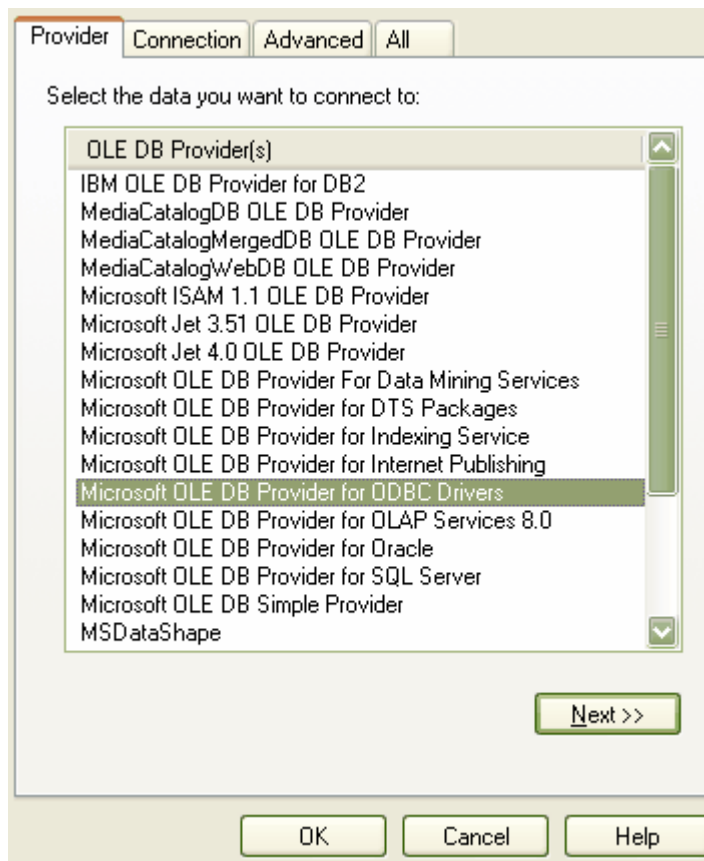
10. Press the **OK** button to complete the configuration.

8.2.3.2.5 Connect to an Adaptive Server Anywhere Data Repository (Corporate Edition only)

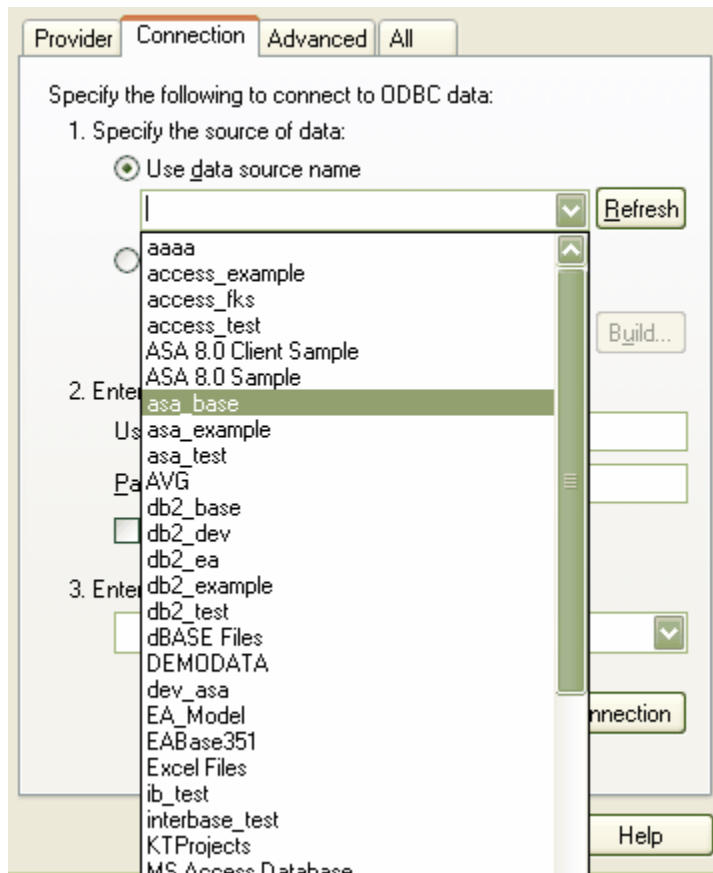
Note: This feature is available in the Corporate Edition only.

In order to use a ASA data repository, you must connect to it in EA first. Before connecting to the repository, you will need to have [set up an ASA ODBC driver](#). Follow these steps to connect to an ASA data repository in EA:

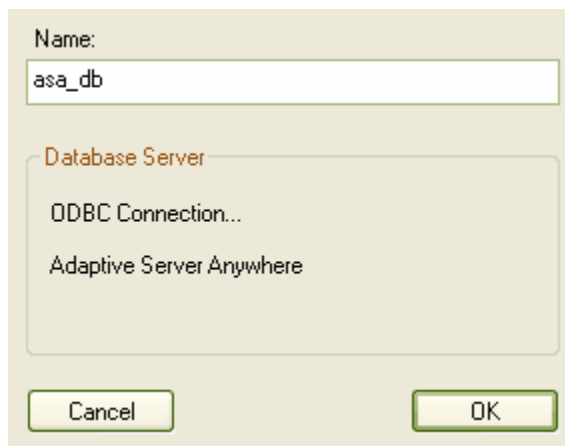
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox, or on the Start Page, click on the **Connect to Server Repository...** link.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press the **Next** button. This will take you to the **Connection** tab (below).



5. Select the ODBC driver you have set up to connect to your ASA repository from the *Use data source name* drop down list.
6. Press the *Test Connection* button to confirm that the details are correct.
7. If your test succeeded, press the *OK* button.
8. If your test did not succeed, revise your settings.
9. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



10. Press the **OK** button to complete the configuration.

8.2.3.2.6 Connect to a MSDE Server Data Repository (Corporate Edition only)

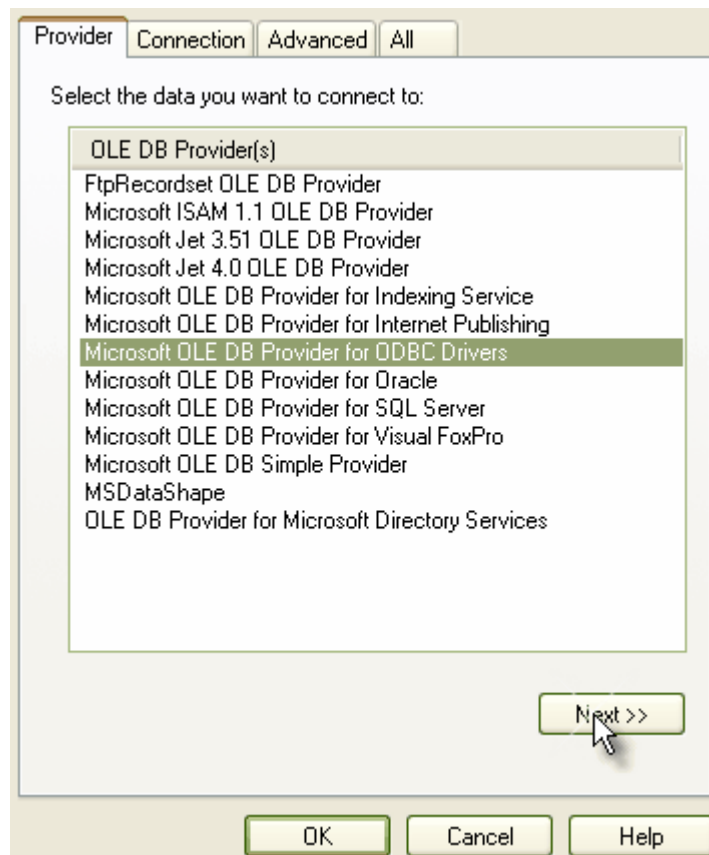
Follow the instructions in [Connect to a SQL Server Repository](#).

8.2.3.2.7 Connect to a Progress OpenEdge Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

In order to use a OpenEdge data repository, you must connect to it in EA first. Follow the steps below to connect to your OpenEdge data repository in EA:

1. In the [Open Project dialog](#), check the **Connect to Server** checkbox.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press **Next**. This will take you to the **Connection** tab (below).

Provider Connection Advanced All

Specify the following to connect to ODBC data:

1. Specify the source of data:

Use data source name

openedge_users Refresh

Use connection string

Connection string: Build...

2. Enter information to log on to the server

User name: oe_user

Password: ●●●●

Blank password Allow saving password

3. Enter the initial catalog to use:

ea

Test Connection

OK Cancel Help

5. Select the ODBC driver you have set up to connect to your OpenEdge repository from the Use data source name drop down list. In the [setup example](#) the driver title was "openedge_users".
6. Enter the *User name* and *Password*.
7. Enter the initial catalog.
8. Open the All tab.
9. Edit the Extended Properties value. As shown below, provide the value of : DefaultSchema=PUB

Property Description

Extended Properties

Property Value

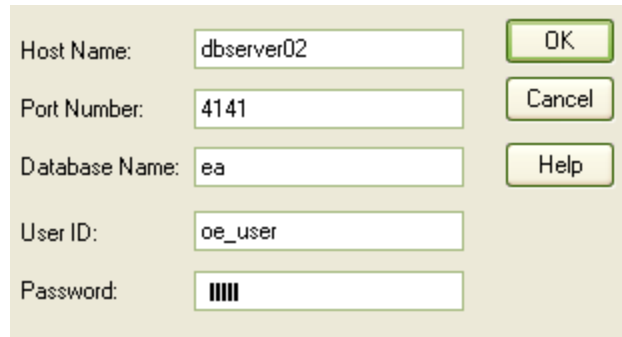
DefaultSchema=PUB

Reset Value OK Cancel

10. Press *Test Connection* to confirm that the details are correct.
11. If the test succeeds, press *OK*.

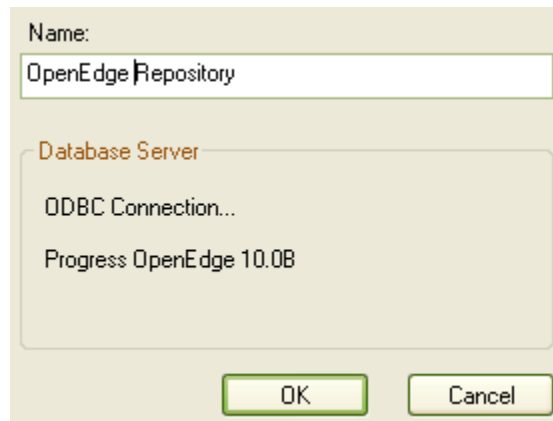
12. If the test does not succeed, revise your settings.

13. After you have pressed OK, the Logon to Progress dialog will appear (below).



A dialog box titled "Logon to Progress" with a light beige background. It contains five input fields and three buttons. The fields are: "Host Name:" with the value "dbserver02", "Port Number:" with the value "4141", "Database Name:" with the value "ea", "User ID:" with the value "oe_user", and "Password:" with masked characters "||||". The buttons are "OK", "Cancel", and "Help", arranged vertically on the right side.

14. Give the connection a suitable name so you will recognize it in the Recently Opened Project on the [open project dialog](#).



A dialog box titled "Name:" with a light beige background. It contains a text input field with the value "OpenEdge Repository". Below the input field is a section titled "Database Server" containing the text "ODBC Connection..." and "Progress OpenEdge 10.0B". At the bottom are "OK" and "Cancel" buttons.

15. Press **OK** to complete the configuration.

8.2.3.3 Creating a Repository

This section details how to create to the following data repositories:

- [MySQL Data Repository](#)
- [SQL Server Data Repository](#)
- [Oracle 9i and 10g Data Repository](#)
- [PostgreSQL Data Repository](#)
- [Adaptive Server Anywhere Data Repository](#)
- [MSDE Server Data Repository](#)

8.2.3.3.1 Create a New MySQL Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

Before creating a MySQL data repository in EA, you will need MySQL and MySQL ODBC drivers set up. For further information on setting these up, refer to [Setup a MySQL ODBC Driver](#).

To create a new MySQL repository, you will need to first create a database into which you will import the table definitions for EA. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

Creating the Data Repository

Once you have created the database and executed the script, you should have an empty EA project to begin working with. You can transfer data from an existing .EAP file or simply start from scratch.

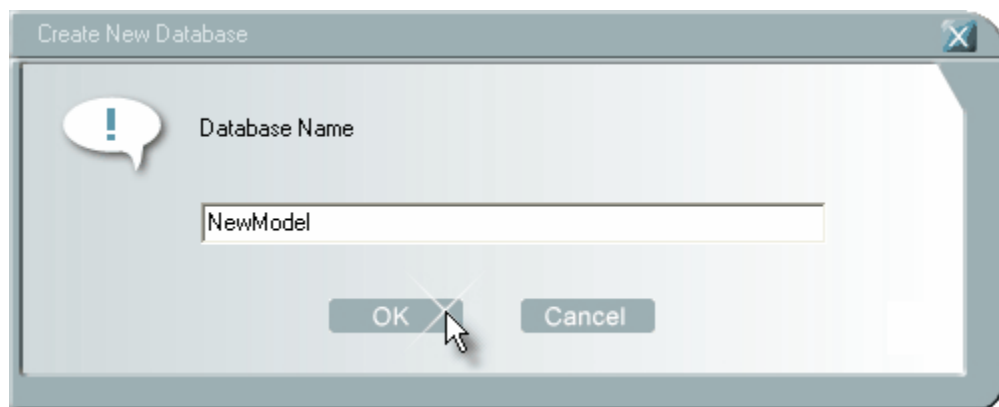
Third Party Tools

If you are unfamiliar with MySQL and DBMS systems in general, you may wish to consider a suitable front end tool.

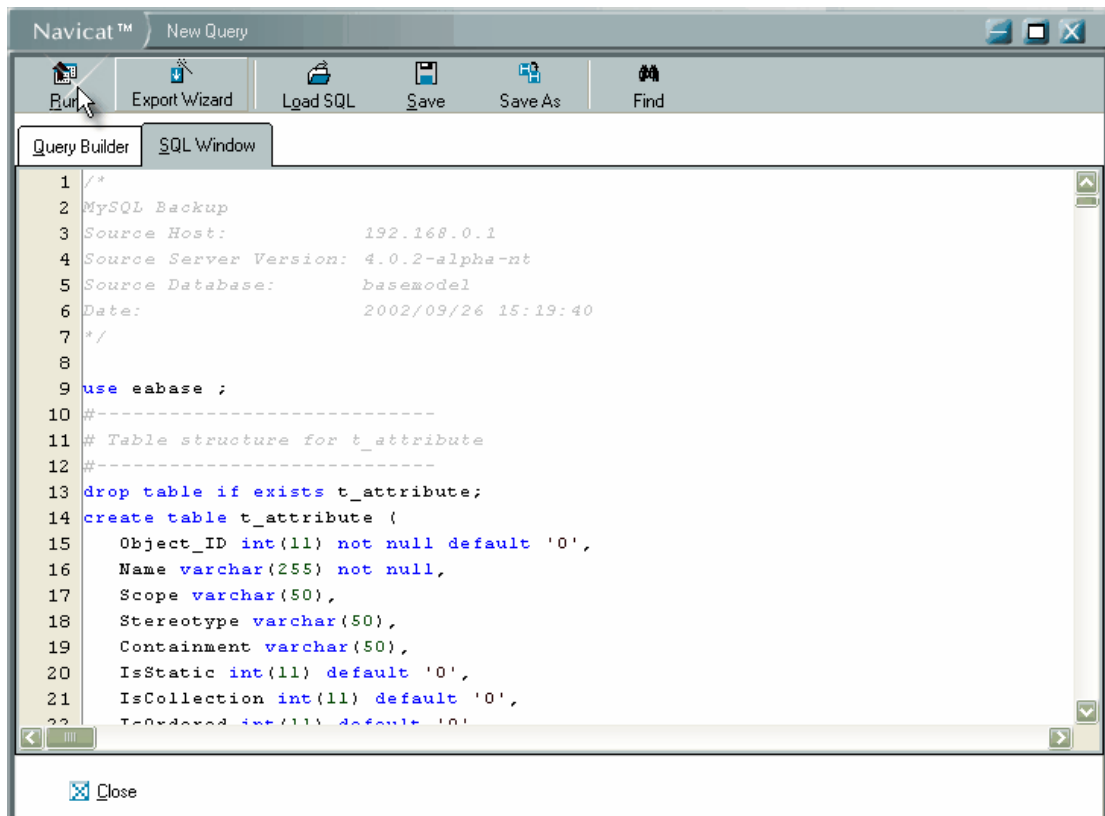
PremiumSoft Navicat (formerly PremiumSoft MySQLStudio) is one such tool, and is available at www.navicat.com. Navicat provides a convenient graphical user interface to enable the creation of databases, execution of scripts, backups, restores and more.

Below are some basic instructions to get you started with Navicat.

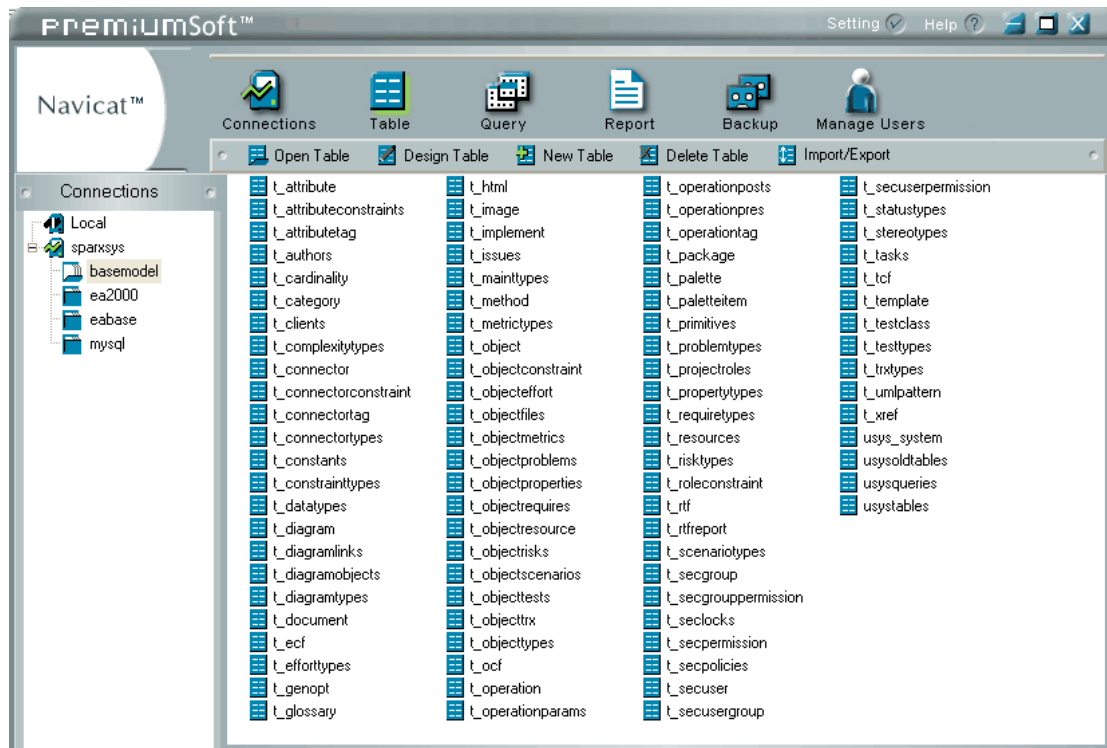
1. Create a new database.



2. Open and execute the MySQL script.



3. Below is an example showing the tables created in the MySQL repository after running the script in Navicat.



8.2.3.3.2 Create a New SQL Server Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

Before creating a SQL Server data repository, you will need SQL Server installed, MDAC 2.6 or 2.7 installed and permission to create a new database. Please note that setting up SQL Server and the issues involved are beyond the scope of this manual. Consult your program's documentation for a guide to this.

Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

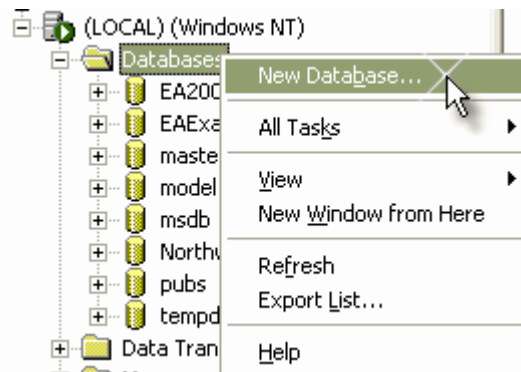
- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

Create a SQL Server Repository

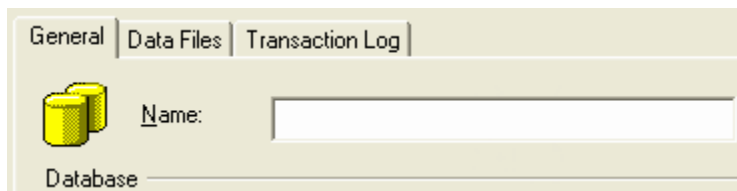
SQL Server repositories are created without any data, so you will need to perform a [data transfer](#) in EA to copy a suitable starter project. If you are starting from scratch, EABase.EAP is a good starting point. If you wish to use an existing .EAP model, you can [upsized](#) it.

You can use SQL Enterprise Manager to create a SQL Server repository by following the below steps:

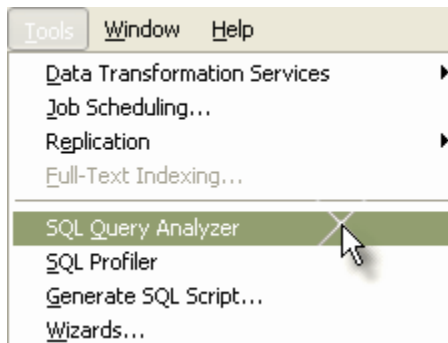
1. In SQL Enterprise Manager, locate the server on which you wish to create your new EA model - in the example below this is (LOCAL).
2. Right click and choose **New Database** from the context menu (below).



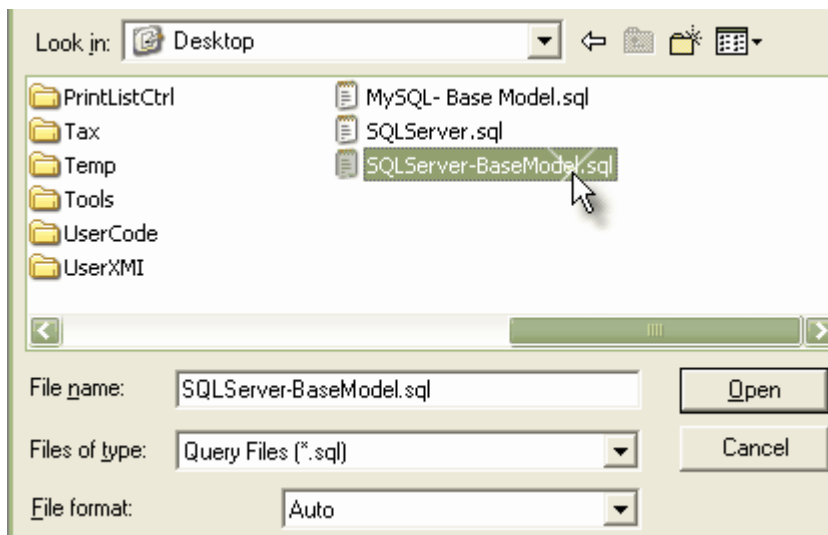
3. Enter a suitable name for the Database. Set any file options etc. as required (below).



4. Click on the database to select it, then go to the *Tools* menu and start *SQL Query Analyzer* (below).



5. In Query Analyzer, use the File Find dialog to locate the supplied EA SQL Server Model script file. Press *Open*.



6. Check that you have selected the correct database to run the script in. In this example the tables are being added to the Base Model database as shown in the drop down menu below.



7. Press **Run** - SQL Server will execute the script which will create the base model for an EA Project.

Tip: Use the [Data Transfer](#) function to upload a basic model into the repository.

8.2.3.3 Create a New Oracle9i Server Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

Before creating a Oracle 9i and 10g data repository, you will need Oracle 9i or 10g installed, MDAC 2.6 or 2.7 installed and permission to create a new database. Please note that setting up Oracle and the issues involved are beyond the scope of this manual. Consult your program's documentation for a guide to this.

Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at http://www.sparxsystems.com.au/registered/reg_ea_oracle_instructions.html.
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

******** **Creating the Data Repository**

Oracle 9i and 10g repositories are created without any data, so you will need to perform a [data transfer](#) in EA to copy a suitable starter project. If you are starting from scratch, [EABase.EAP](#) is a good starting point. If you wish to use an existing .EAP model, you can [upsized](#) it.

1. Create a new database on the Oracle 9i and 10g server.
2. Connect to the newly created database with a program such as *Oracle SQL*Plus* or *SQL Plus Worksheet*.
3. Execute the script Oracle_BaseModel.sql which will create the base model tables and indexes for an EA Project.

Tip: Use the [Data Transfer](#) function to upload a basic model into the repository.

8.2.3.4 Create a New PostgreSQL Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

Before creating a PostgreSQL data repository in EA, you will need PostgreSQL and PostgreSQL ODBC drivers set up. For further information on setting these up, refer to [Setup a PostgreSQL ODBC Driver](#).

To create a new PostgreSQL repository, you will need to first create a database into which you will import the table definitions for EA. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

Creating the Data Repository

After you create the database and execute the script, the result should be an empty EA project to begin working with. You can transfer data from an existing .EAP file or simply start from scratch.

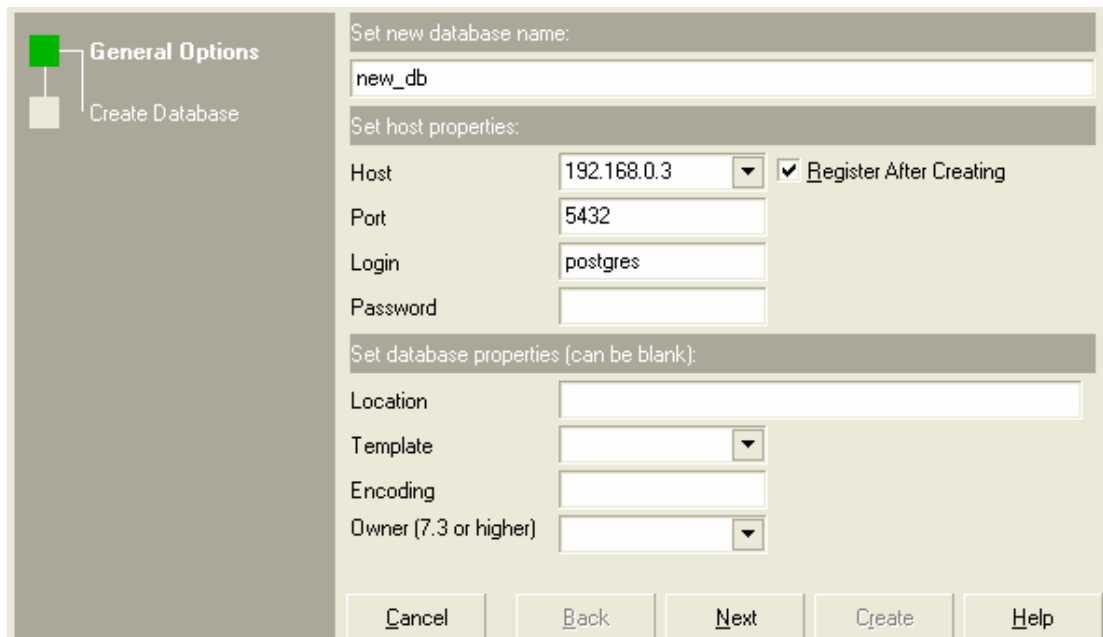
Third Party Tools

If you are unfamiliar with PostgreSQL and DBMS systems in general, you may wish to consider a suitable front end tool.

EMS PostgreSQL Manager is one such tool, and is available at www.sqlmanager.net/products/postgresql/manager. It provides a convenient graphical user interface to enable creation of databases, execution of scripts, restores and more.

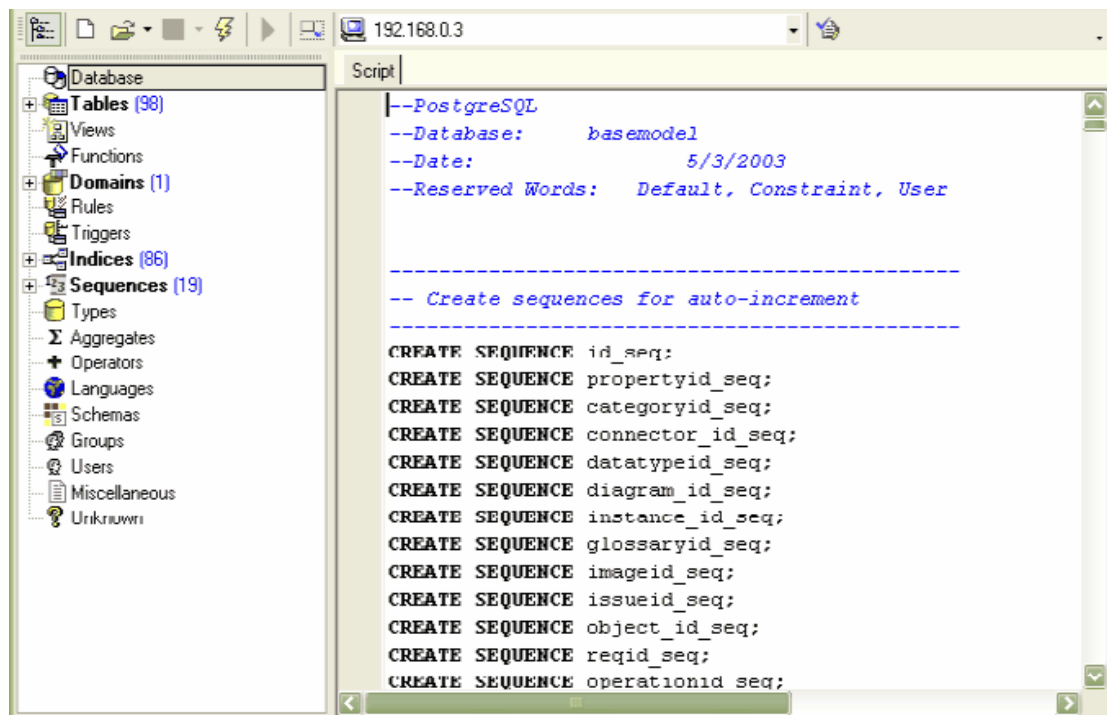
Below are some basic instructions to get you started with EMS PostgreSQL Manager.

1. Create a new database.

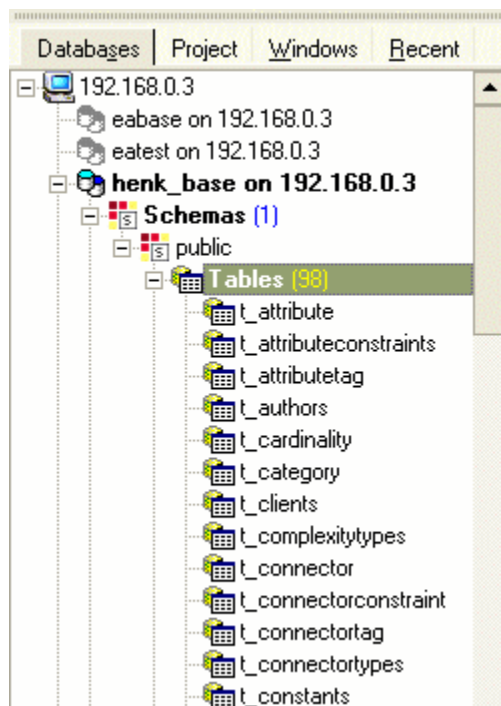


The screenshot shows the 'Create Database' dialog box in EMS PostgreSQL Manager. The dialog is titled 'General Options' and 'Create Database'. It is divided into three sections: 'Set new database name:', 'Set host properties:', and 'Set database properties (can be blank):'. The 'Set new database name:' section has a text box containing 'new_db'. The 'Set host properties:' section includes a 'Host' dropdown menu with '192.168.0.3' selected, a 'Port' text box with '5432', a 'Login' text box with 'postgres', and a 'Password' text box. A checkbox labeled 'Register After Creating' is checked. The 'Set database properties (can be blank):' section includes a 'Location' text box, a 'Template' dropdown menu, an 'Encoding' text box, and an 'Owner (7.3 or higher)' dropdown menu. At the bottom of the dialog are five buttons: 'Cancel', 'Back', 'Next', 'Create', and 'Help'.

2. Open and execute the PostgreSQL sql script.



- Below is an example showing the tables created in the PostgreSQL repository after running the script in EMS PostgreSQL Manager.



8.2.3.3.5 Create a New Adaptive Server Anywhere Repository (Corporate Edition Only)

Note: This feature is available in the Corporate Edition only.

Before creating a ASA data repository in EA, you will need ASA and ASA ODBC drivers set up. For further information on setting these up, refer to [Setup an Adaptive Server Anywhere ODBC Driver](#).

To create a new ASA repository, you will need to first create a database into which you will import the table definitions for EA. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

Creating the Data Repository

After you create the database and execute the script, the result should be an empty EA project to begin working with. You can transfer data from an existing .EAP file or simply start from scratch.

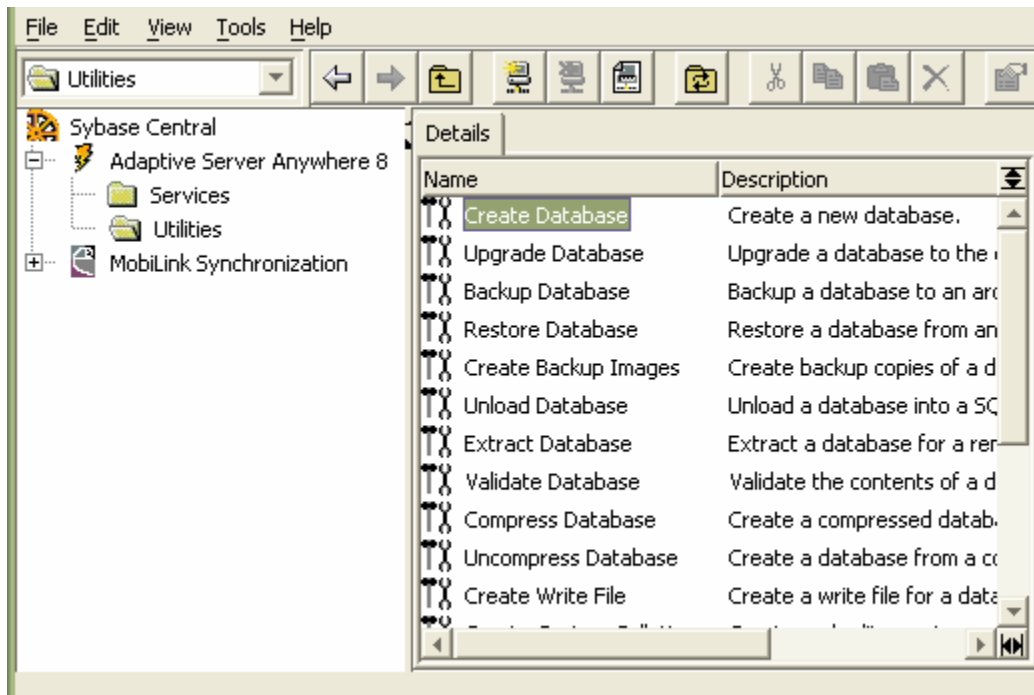
Third Party Tools

If you are unfamiliar with ASA and DBMS systems in general, you may wish to consider a suitable front end tool.

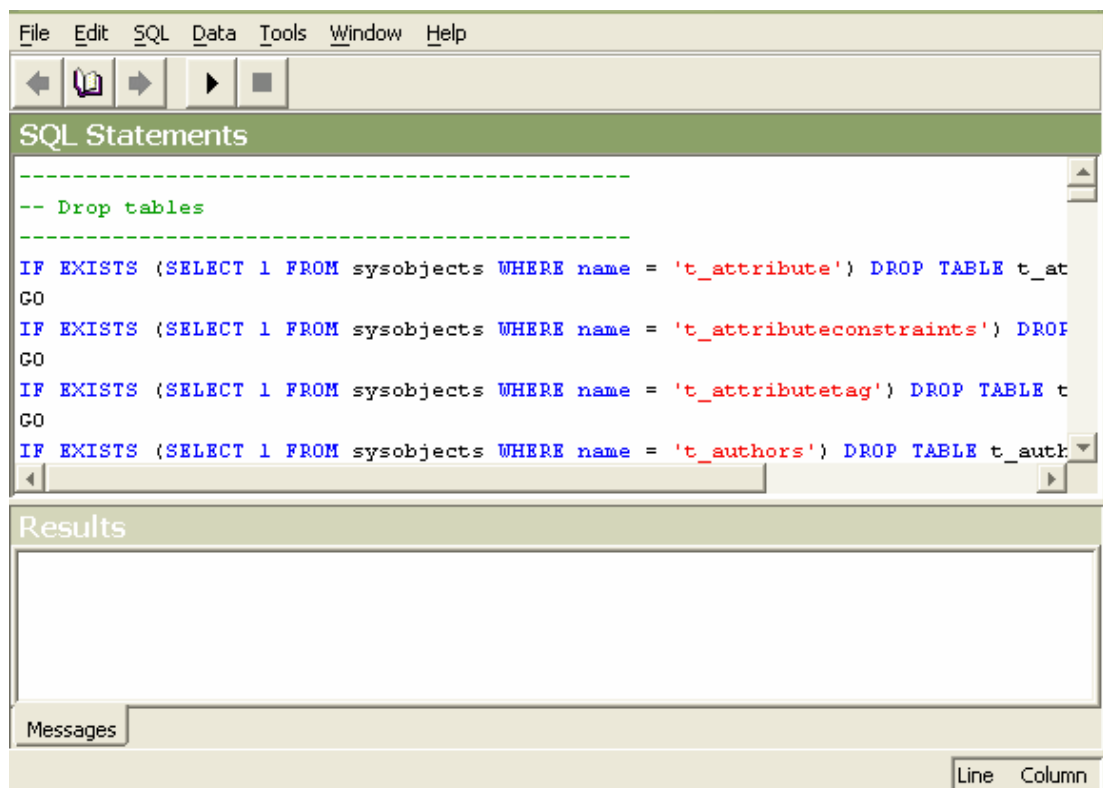
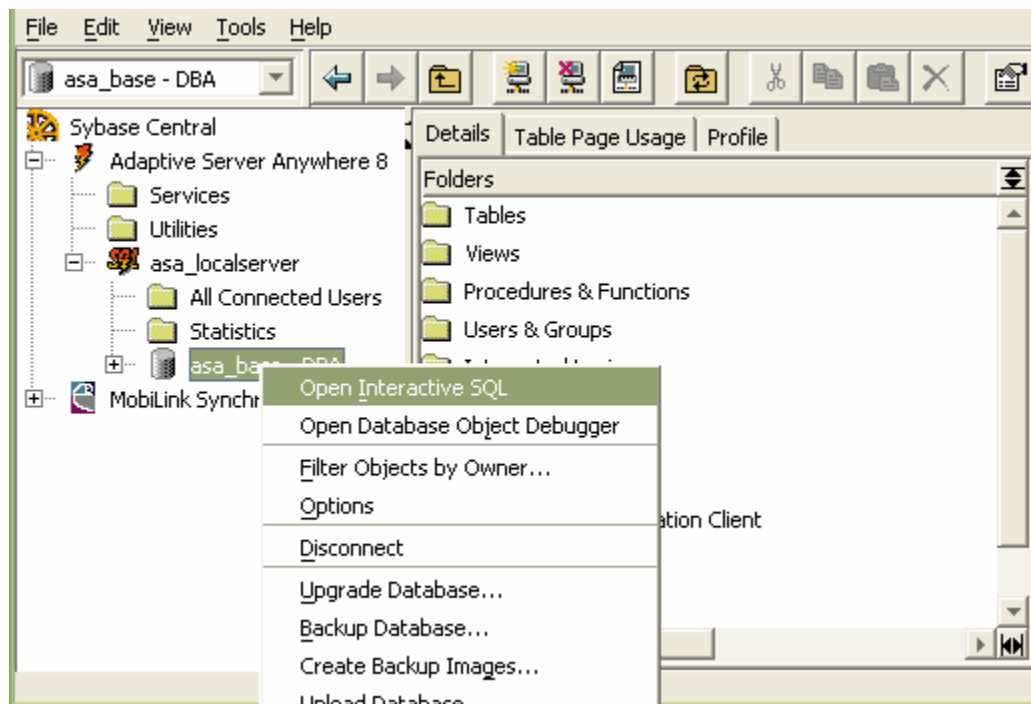
Sybase Central is one such tool, that can be installed along with the DBMS. It provides a convenient graphical user interface to enable creation of databases, execution of scripts, restores and more.

Below are some basic instructions to get you started with Sybase Central.

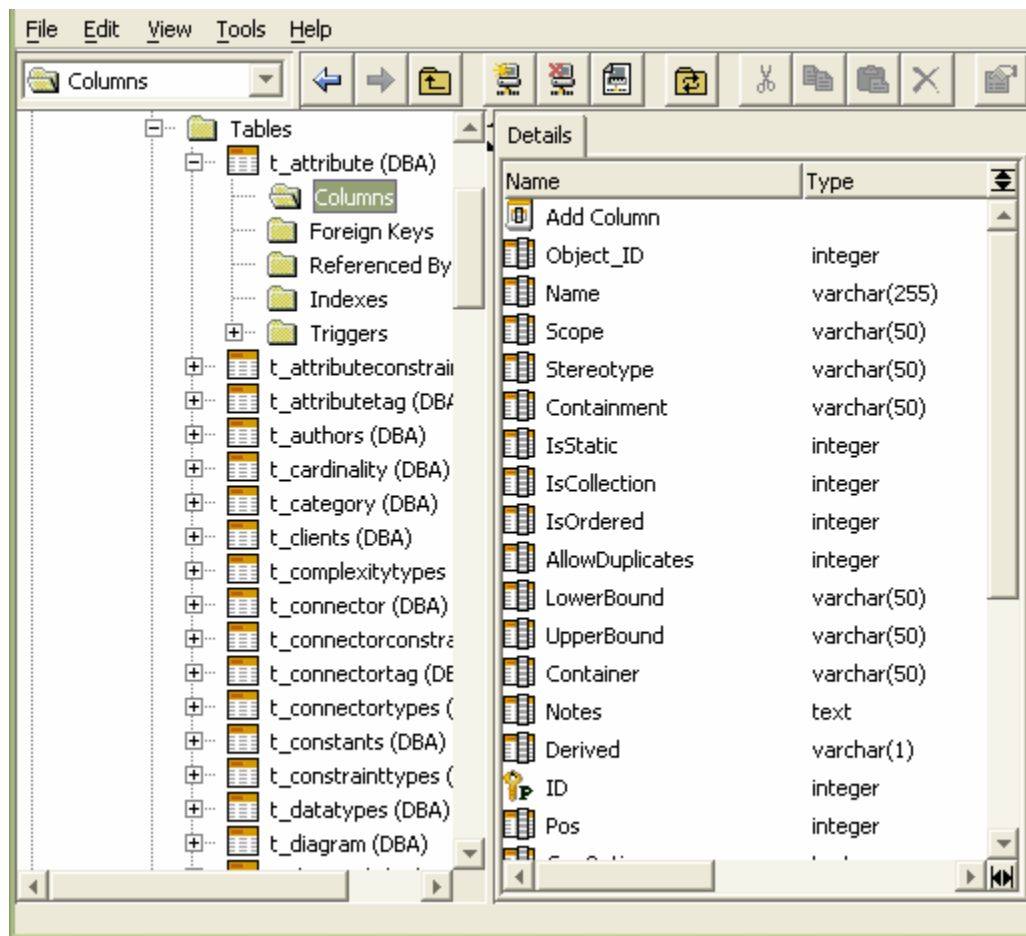
1. Create a new database.



2. Open and execute the ASA SQL script.



3. Below is an example showing the tables created in the ASA repository after running the script in EMS ASA Manager.



8.2.3.3.6 Create a New MSDE Server Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

Before creating a SQL Server MSDE data repository, you will need MSDE Server installed, MDAC 2.6 or 2.7 installed. Please note that setting up MSDE Server and the issues involved are beyond the scope of this manual. Consult your program's documentation for a guide to this.

Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

Use the SQL Server 2000 and 2005 script for MSDE, and follow the instructions to [Create a New SQL Server Data Repository](#).

8.2.3.3.7 Create a New Progress OpenEdge Repository (Corporate Edition only)

Note: This feature is available in the Corporate Edition only.

Before creating a Progress OpenEdge data repository, you will need OpenEdge installed, MDAC 2.6 or 2.7 installed and permission to create a new database. Please note that setting up OpenEdge and the issues involved are beyond the scope of this manual. Consult your OpenEdge documentation for a guide to this.

Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script is up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at

- Registered users can obtain the scripts from the Registered Corporate Edition resources page of the Sparx Systems website at http://www.sparxsystems.com.au/registered/reg_ea_openedge_instructions.html
- Trial users can obtain the scripts from the Corporate Edition Resources page of the Sparx Systems website at <http://www.sparxsystems.com.au/resources/corporate/>

http://www.sparxsystems.com.au/registered/reg_ea_openedge_instructions.html.

Creating the Data Repository

OpenEdge repositories are created without any data, so you will need to perform a [data transfer](#) in EA to copy a suitable starter project. If you are starting from scratch, [EABase.EAP](#) is a good starting point. If you wish to use an existing .EAP model, you can [upsized](#) it.

1. Run proenv from the OpenEdge menu: Start->Programs->OpenEdge->proenv.
2. Create database: prodb <database_name> empty.
3. Start database server: proserve <database_name> -S <port_number>
4. Open Data Administration to add a user:

```
prowin32 -db <database_name> -S <port_number> -p _admin -rx
```

5. Open Admin->Security->Edit User List.
6. Close Data Administration.
7. Open SQL Explorer Tool, connect as 'sysprogress'.
8. Add user:

```
create user 'user', 'password';  
commit;
```

9. Grant privileges:

```
grant dba, resource to <user>;  
commit;
```

Tip: Use the [Data Transfer](#) function to upload a basic model into the repository.

8.2.4 Create and Open Model Files Discussion

MySQL Limitations

Note that use of MySQL has the following limitations:

1. If MyISAM table types are used (default), transactional support is disabled. To enable transactions you will need to set up MySQL to use InnoDB tables and create the database tables as InnoDB type. Sparx provide a suitable script to create InnoDB based repository tables - as well as the more common MyISAM. These are available to registered users on the Corporate Edition resources page of the Sparx website at [www](#).

sparxsystems.com.au/registered/reg_ea_corp_ed.html.

2. Due to some limitations of the SQL query engine, some advanced search capabilities in the Search Dialog are disabled.
3. Only MySQL 4.0.10 or later is supported.
4. MySQL ODBC Driver 3.50 or higher is required. This is the development version, but again the earlier version does not provide sufficient support to run EA.

SQL Scripts

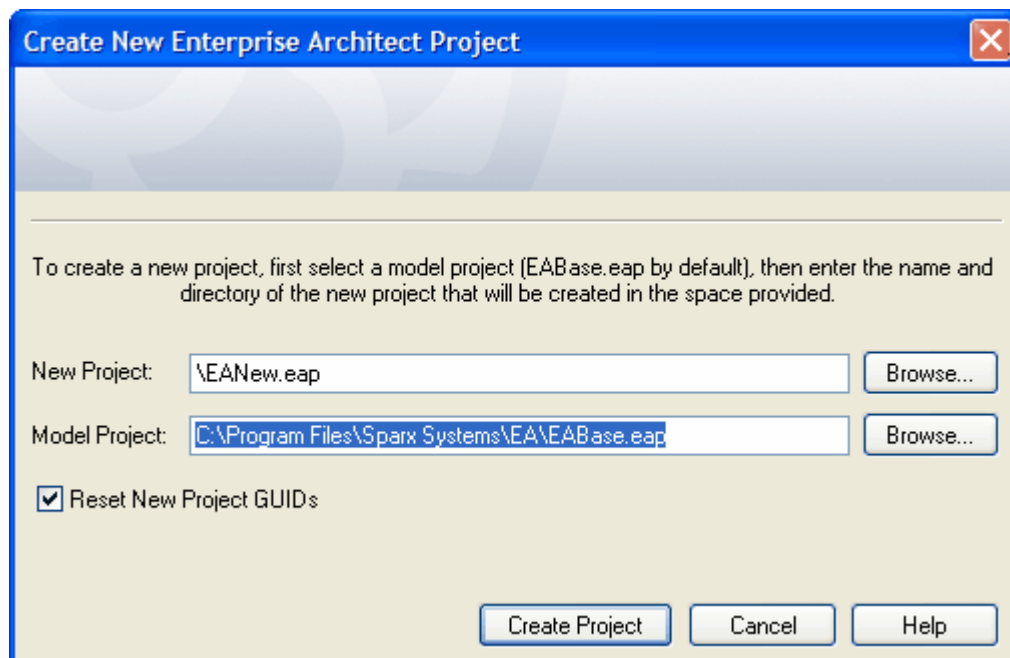
Sparx Systems provide various SQL scripts to assist with creating data repositories. These are available to registered users on the Corporate Edition resources page of the Sparx website at www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.

8.2.5 Copy a Base Project

To copy an existing Base Project

From the [Start Page](#), select *Copy a Base Project...*

The *Create a New Model* dialog will appear (below). To create a new Enterprise Architect project, you need to select a project template that will form the base model for the new project. When you install Enterprise Architect a default model is installed called EABase.EAP



- Use the *New Project Browse...* option to select where to save your project. If this is to be a shared project, place the file on a shared network resource (eg. Network Server or Workgroup Server).
- Use the Reset New Projects GUIDs option to replace all GUIDs in the source model with fresh GUIDs.
Note: *If this new project is based on one that is already under version control, it is recommended that the option be disabled. This will prevent duplication of packages when Get Latest is used.*
- Use the *Model Project Browse...* option to select the [base model for your project](#). EABase.EAP is the default, however you can select any existing model file you wish (see [Designing a Custom Template](#)).

When you have entered the filenames, press **Create Project** to create your project. The **Cancel** button closes the dialog without creating a new project.

Tip: You can copy any Enterprise Architect project using Windows Explorer, and open the copied project as a new project.

8.3 Upgrading Models

The structure of Enterprise Architect project files is occasionally changed to support more features. When this happens, existing project files must be upgraded to the new format to ensure correct operation and to take advantage of all the new features.

Upgrading is a simple and quick process. You can use the [Upgrade Wizard](#), which will alert you to the upgrade need, and take you through the upgrade process. This will bring your project to the current level to support all the latest EA features.

See Also

- [Upgrade Replicas](#)

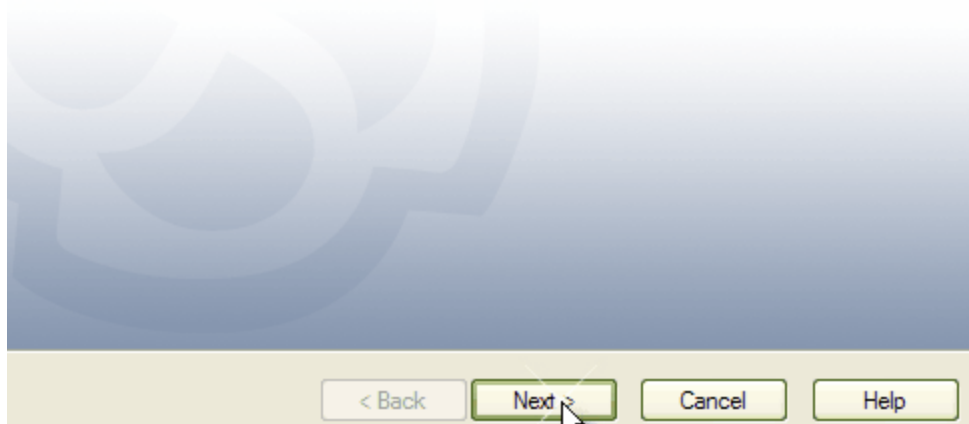
8.3.1 The Upgrade Wizard

When you try to load a project that needs to be updated to the latest format, the upgrade wizard will open and guide you through the required steps. You cannot load a previous EA version model without upgrading.

Upgrading Enterprise Architect

The project you have selected to open was created with an older version of Enterprise Architect. To work correctly it must now be upgraded to the latest version.

This wizard will guide you through the upgrade procedure.



Note: Upgrading is essential when a new major version is released. Once upgraded, the project cannot be opened with earlier versions of EA.

The wizard will:

- Alert you to the need to upgrade.
- Advise you to back up the current project. Backing up before any changes are made is essential.
- Check if the current project can be upgraded. All projects can be upgraded, except replicas which are not Design Masters.
- Perform the upgrade.
- Open the newly converted project.

Note for replicated models: If the wizard detects the project you are opening is a replica and not a design master, a different upgrade path is required.

8.3.2 Upgrade Replicas

Models that have replication features added may need to be handled in a way different from regular projects.

If the model is a [Design Master](#) (the root model of all other replicas) then you can upgrade the model to the current version. After upgrading a Design Master you should re-create the replicas, rather than synchronizing.

If the model is not a Design Master, the model cannot be upgraded in the normal manner. First EA must remove the replication features, then upgrade the project in the normal manner. The [Upgrade Wizard](#) will guide you through these steps.

The Upgrade Wizard should handle the upgrade process for you, detected which upgrade method is the best.

8.4 Data and Model Integrity

In the event of a failed XML import, network crash or other unforeseen event that could disrupt the integrity of information in the model, it is recommended to run the [Data Integrity check function](#). This will examine all database records and ensure there are no 'orphaned' records or inaccurate or unset identifiers. You can run the integrity checker first in report mode to discover if anything needs correcting, and then run it again in repair mode.

When EA checks the model, it will attempt to recover lost packages and elements, and will generate a new package at the model root level called '_recovered_'. Check through any elements that are found and if necessary drag them into the model proper. If they are not required, delete them.

Note: This function does NOT check UML conformance ... only the data relationships and repository structure.

You can select a variety of items to check, and elect to either Report Only on the state of your model - or to try and repair any inconsistencies. The recovery process will try and restore elements where possible, but in some cases, will simply delete the lost records.

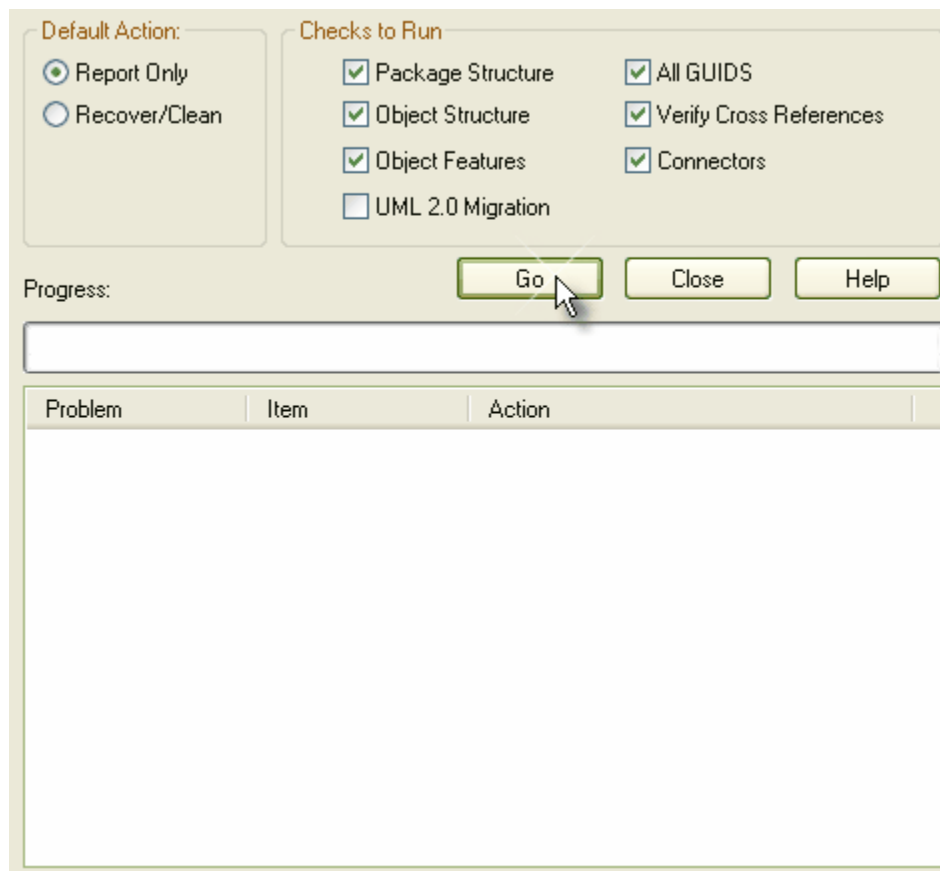
See Also

- [Running SQL Patches](#)

8.4.1 Checking Data Integrity

To check the data integrity of your model, follow these steps:

1. From the **Tools** | **Data Management** submenu, select **Data Integrity**. This will open the **Project Integrity Check** dialog:



2. Select which checks you want to run - the basic checks available are:
 - Package Structure
 - Object Structure
 - Object Features
 - GUIDs
 - Cross References
 - Connectors
 - UML 2.0 Migration
3. Select whether you just want to view a report of the state of your model, or whether you want to go ahead and attempt to recover/clean your project.

Warning: You should back up your project file first if you select the *Recover/Clean* option.

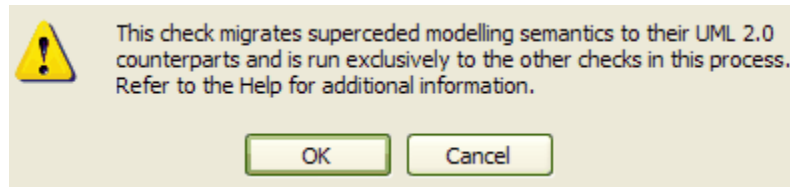
4. Press **Go** to run the check.

UML 2.0 Migration

The UML 2.0 Migration check allows users to migrate the project from the UML 1.3 semantic's to UML 2.0 semantic's. The Migration process currently converts activities that are invocations of operations, into called operation actions as per the UML 2.0 specification. The UML 2.0 Migration option is an exclusive process which does not allow any of the other checks to be selected. To perform the UML 2.0 migration use the following steps:

1. From the *Tools | Data Management* submenu, select *Data Integrity*. This will open the *Project Integrity Check* dialog as shown previously.
2. Check the *UML 2.0 Migration* checkbox and press the **Go** button. This will display the following dialog,

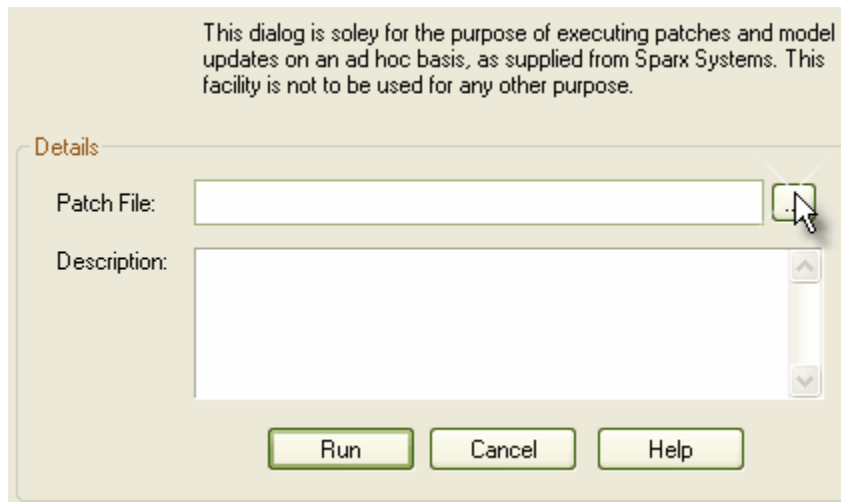
press the **OK** button to proceed or cancel the migration by pressing the **Cancel** button.



2. Press the **Go** button to perform the migration.

8.4.2 Running SQL Patches

Occasionally, Sparx Systems may release a patch to correct a model fault. If a patch is released, you can load it here and run from the **Tools | Run Patch** menu. The patch will generally check how many records will be updated, and report on what will be done.



8.5 Data Transfer

The Corporate Edition of EA supports [SQL Server](#), [MySQL](#) and [Oracle 9i and 10g](#) data repositories. At some point, the need may arise to move a complete model from one repository to another, row by row, table by table.

The data transfer function allows you to perform the following tasks:

- Upload an existing EAP file to SQL Server or MySQL
- Download a repository in MySQL or SQL Server to an EAP file
- Move a repository from SQL Server to MySQL or from one server to another
- Move all records from an EAP file with replication to a model with none (Remove Replication)
- Copy all records from an EAP file to another (recommended after serious network crash or repeated database corruption)
- Copy all records from a JET 3.5 to JET 4 (Access 2000 or XP) - or back the other way

See the [Perform a Data Transfer](#) topic for instructions.

WARNING: ALL RECORDS IN THE TARGET REPOSITORY WILL BE OVERWRITTEN

8.5.1 Perform a Data Transfer

To use the data transfer function, follow these steps:

1. From the **Tools** | **Data Management** submenu, select **Data Transfer**. This will open the **Full Model Data Transfer** dialog:

The dialog box is titled "Full Model Data Transfer". It contains the following elements:

- Data Transfer Type:** Four radio buttons are present. The first is ".EAP to .EAP", the second is "DBMS to .EAP", the third is ".EAP to DBMS" (which is selected), and the fourth is "DBMS to DBMS".
- Source and Target Models:** Two text boxes with browse buttons ("..."). The "Source Model" box contains "C:\Documents and Settings\John\Desktop\EA Models\EA Ne". The "Target Model" box contains "DBType=0;Connect=Provider=MSDASQL.1;Persist Security In".
- Logfile:** A checked checkbox labeled "Logfile" followed by a text box containing "C:\Temp\Example_to_MYSQL.log" and a browse button ("...").
- Caution:** A text block stating: "Caution: All data will be erased from the Target Model prior to data transfer. Please ensure you have backed up target if necessary".
- Buttons:** Three buttons at the bottom: "Help", "Transfer Data", and "Close".
- Progress:** A label "Progress:" followed by an empty progress bar.

2. Select the **Data Transfer Type** - you can choose from:
 - .EAP to .EAP
 - DBMS to .EAP
 - .EAP to DBMS
 - DBMS to DBMS
3. Enter the name or connection string for the **Source** and **Target** models.
4. Press **Transfer Data**.
5. It is good practise to do a [Database Compare](#) after this process to verify all records are written.

WARNING: ALL RECORDS IN THE TARGET REPOSITORY WILL BE OVERWRITTEN

8.5.2 Why Compare Models?

It is sometimes useful to compare the size and row counts of two models. After a database crash, after import from XML or after performing a deletion of model elements - these are all examples of when it may be useful to compare models.

You can compare EAP files to other EAP files or to DBMS based repositories - or compare two DBMS repositories. EA will examine the number of rows in each database and produce a report indicating the total records in each and the difference between the two. No examination is made of the data in the table - just the record count.

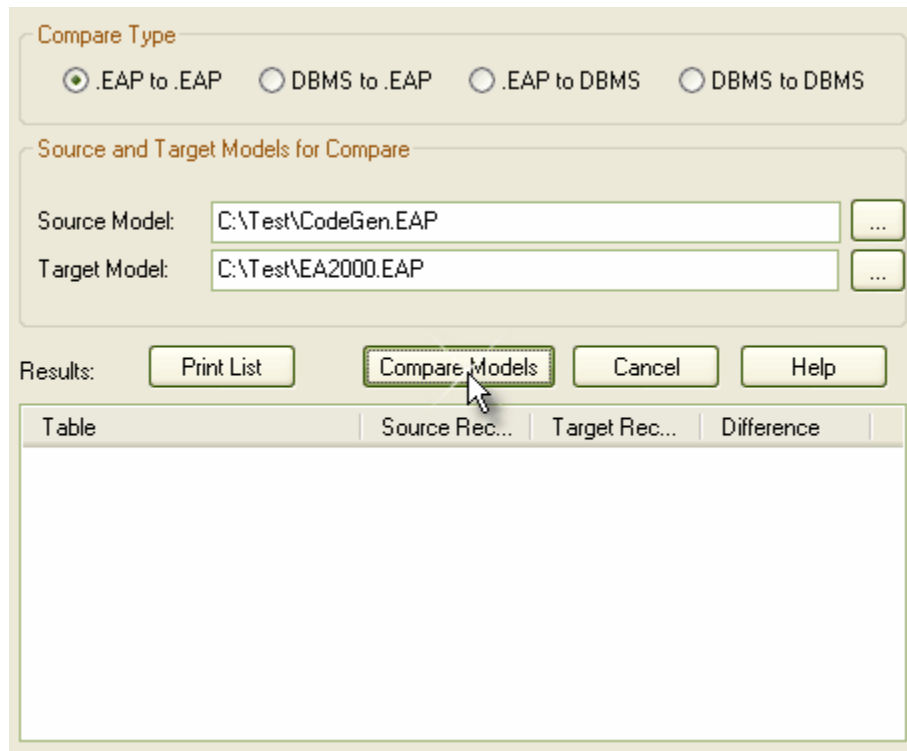
Comparing models this way is a convenient 'sanity check' after restoring a backup or doing a data transfer. If discrepancies are found, you will need to investigate further manually.

See the [Compare Models](#) topic for instructions.

8.5.3 Comparing Models

To compare models, follow the steps below:

1. From the **Tools | Data Management** submenu, select **Data Compare**. This will open the **Compare table record counts** dialog:



2. Select the **Compare Type** - you can choose from:
 - .EAP to .EAP
 - DBMS to .EAP
 - .EAP to DBMS
 - DBMS to DBMS

3. Enter the name or connection string for the *Source* and *Target* models.
4. Press *Compare Models*. The results will appear in the listbox below. You can also print the list by pressing *Print List*.

8.5.4 Copy Packages from One Project to Another

Using the XML import/export capabilities of Enterprise Architect, you can copy and move packages between EA models. This allows for a high level of flexibility in building a model from re-usable parts and from elements produced in widely dispersed geographic regions.

Copy a Package from One EA Model to Another

To copy a package from one EA model to another, follow the steps below:

1. Open the EA model you wish to copy from.
2. In the Project Browser, right click on the package you wish to copy.
3. Select *Export Package to XML File* from the *Import/Export* submenu. This will open the *Export Package to XML* dialog:

Root Package: CORBA

Filename: C:\XML\Corba\Corba.xml

Stylesheet: (Optional stylesheet to post process XML content)

General Options

- Export Diagrams
- Format XML Output
- Write Log file
- Use DTD
- Generate Diagram Images

Format: [Dropdown]

For Export to Other Tools

- Unisys/Rose Format
- XMI 1.0
- XMI 1.2
- Exclude EA Tagged Values

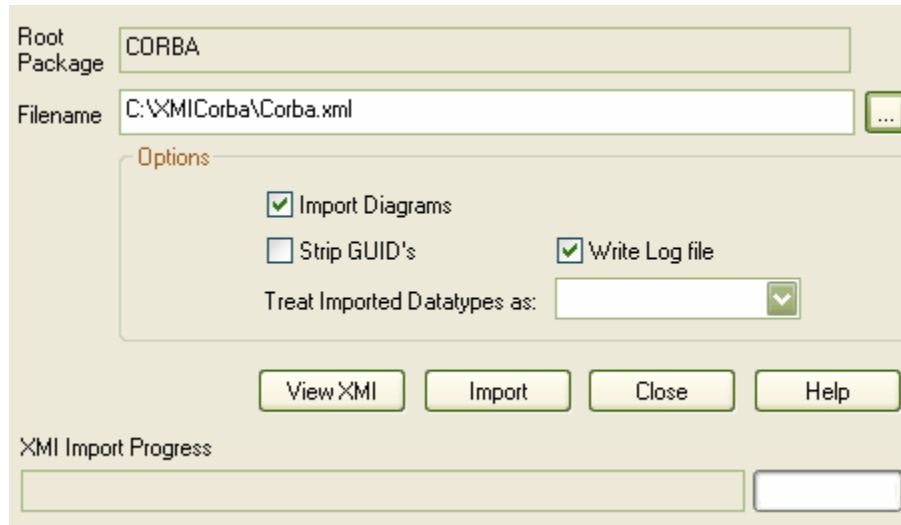
Warning: These options are for exporting EA model elements to other tools only.

Buttons: View XML, Export, Close, Help

Progress: [Progress Bar]

4. Select the appropriate options and filename in the *Export Package to XML* dialog (see [Export Package to XML](#) for further information).
5. Press *Export* to begin the export process.
6. When the export is complete, open the recipient EA model. In the Project Browser, navigate to the location you wish to import the package into.
7. Right click to view the context menu, and select *Import Package from XML File* from the *Import/*

Export submenu. This will open the *Import Package from XML* dialog:



8. [Import Package from XML](#) Select the appropriate options and filename in the *Import Package from XML* dialog (see [for further information](#)).
9. The package has now been copied from the source project to the destination project.

Note: If the package you are importing already exists in the target model (has been imported previously), then you must either import over the existing one, -OR- select to Strip GUIDS - in which case you will get a replica/copy of the original. You may also use this technique to copy and entire package within the same model

8.6 Upsizing Models

The Desktop and Professional versions of Enterprise Architect use an MS JET database as the model repository. If you purchase the Corporate Edition, you have the option to use a [SQL Server](#) 2000 and 2005, [MySQL](#), [Oracle 9i and 10g](#), [PostgreSQL](#), [Adaptive Server Anywhere](#) 8 and 9, [Progress OpenEdge](#) or [MSDE](#) data repository.

The process of upsizing a model is fairly straightforward and involves the following steps:

1. Install the DBMS software and create a database.
2. Run a [script supplied by Sparx Systems](#) to create the required tables.
3. Open EA and use the [Data Transfer](#) function (under the *Tools | Data Management | Data Transfer* menu) to move a model from a .EAP file to the DBMS repository.

This section details how to do this for [MySQL](#), [SQL Server](#), [Oracle 9i and 10g](#), [PostgreSQL](#), [Adaptive Server Anywhere](#), [Progress OpenEdge](#) or [MSDE](#).

8.6.1 Upsizing to MySQL

Before you set up EA for use with MySQL, we recommend you run the [Tools | Data Management | Data Integrity](#) tool on the base project you wish to upsize to MySQL. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for MySQL. Follow them in order:

Stage One: Install MySQL Components

1. Install MySQL - version 4.0.3 or higher.
2. Install MySQL ODBC 3.51 or higher.
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Set up a MySQL ODBC Driver](#).

Stage Two: Select Table Type

1. If you wish to use InnoDB tables, set up the MySQL .ini file as required and run the MySQL - InnoDB BaseModel script.
2. If you wish to use MyISAM tables, set up the MySQL .ini file as required and run the MySQL - MyISAM BaseModel script.

Note: See discussion on [MySQL limitations](#).

Note: The scripts are available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.

Stage Three: Create the Database

1. Create an empty database.

Note: See [Create a New MySQL Repository](#).

2. Now that have an empty database you can use the [Tools | Data Management | Data Transfer](#) menu option in EA to transfer an existing model into the server.

Stage Four: Transfer the Data

1. Open EA (you can press **Cancel** at the **Open Project** screen to open with no project loaded).
2. From the [Tools | Data Management](#) submenu, select **Data Transfer**. This will open the **Full Model Data Transfer** dialog:

3. Select **.EAP to DBMS** as the *Data Transfer Type*.
4. Enter the name of the **.EAP** file to upsize to MySQL as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press *Next*.
7. Select the ODBC Data Source you configured to point to your new database.
Note: See [Connect to a MySQL Data Repository](#) for more information.
8. Press *OK*.
9. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
10. Press *Transfer Data* to begin the data transfer process.

Once the process is completed, you will have upsized your model to MySQL and can now open it from EA.

8.6.2 Upsizing to SQL Server

Before you set up EA for use with SQL Server, we recommend you run the [Tools | Data Management | Data Integrity](#) tool on the base project you wish to upsize to SQL Server. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are three stages to upsizeing your database for SQL Server. Follow them in order:

Stage One: Create an Empty Database

1. Install SQL Server.
2. Create an empty database.

Note: See [Create a New SQL Server Repository](#).

Stage Two: Configure the Database

1. Using a tool such as the SQL Query Analyser, load the SQL Server - Base Model.sql file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
2. Make sure the new database is the currently active database.
3. Run the script to create all required tables/indexes etc.

Note: See [Create a New SQL Server Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the **Tools | Data Management | Data Transfer** menu option in EA to transfer an existing model into the server.

1. Open EA (you can press **Cancel** at the **Open Project** screen to open with no project loaded).
2. From the **Tools | Data Management** submenu, select **Data Transfer**. This will open the **Full Model Data Transfer** dialog.

Data Transfer Type

.EAP to .EAP DBMS to .EAP .EAP to DBMS DBMS to DBMS

Source and Target Models

Source Model: C:\Documents and Settings\John\Desktop\EA Models\EANe ...

Target Model: DBType=0;Connect=Provider=MSDASQL.1;Persist Security In ...

Logfile

Logfile C:\Temp\Example_to_MYSQL.log ...

Caution: All data will be erased from the Target Model prior to data transfer. Please ensure you have backed up target if necessary

Help Transfer Data Close

Progress:

3. Select .EAP to DBMS as the *Data Transfer Type*.
4. Enter the name of the .EAP file to upsize to SQL Server as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Microsoft OLE DB Provider for SQL Server" from the list, then press *Next*.
7. On the *Data Source* details page of the connection dialog, enter the server name, database name, security details as required.

Note: See [Connect to a SQL Server Data Repository](#) for more information.

8. Press *OK*.
9. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
10. Press *Transfer Data* to begin the data transfer process.

Once the process is completed, you will have upsized your model to SQL Server and can now open it from EA.

8.6.3 Upsizing to Oracle

Before you set up EA for use with Oracle, we recommend you run the *Tools | Data Management | Data Integrity* tool on the base project you wish to upsize to Oracle. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are three stages to upsizing your database for Oracle. Follow them in order:

Stage One: Create an Empty Database

1. Install Oracle.
2. Create an empty database.

Note: See [Create a New Oracle Repository](#).

Stage Two: Configure the Database

1. Using a tool such as the *SQL*Plus* or *SQL Plus Worksheet*, load the *Oracle_BaseModel.sql* file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
2. Make sure the new database is selected as the current database.
3. Run the script to create all required tables/indexes etc.

Note: See [Create a New Oracle Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the Project | Manage Data | Data Transfer menu option in EA to transfer an existing model into the server.

1. Open EA (you can press *Cancel* at the *Open Project* screen to open with no project loaded).
2. From the *Tools* | *Data Management* submenu, select *Data Transfer*. This will open the *Full Model Data Transfer* dialog:

The screenshot shows the 'Full Model Data Transfer' dialog box. It is divided into three main sections. The first section, 'Data Transfer Type', contains four radio buttons: '.EAP to .EAP', 'DBMS to .EAP', '.EAP to DBMS' (which is selected), and 'DBMS to DBMS'. The second section, 'Source and Target Models', has two text input fields. The 'Source Model' field contains the path 'C:\Documents and Settings\John\Desktop\EA Models\EANe'. The 'Target Model' field contains the connection string 'DBType=0;Connect=Provider=MSDASQL.1;Persist Security In'. Both fields have a browse button ('...') to their right. The third section, 'Logfile', has a checked checkbox labeled 'Logfile' and a text input field containing 'C:\Temp\Example_to_MYSQL.log', also with a browse button ('...'). Below these sections is a caution message: 'Caution: All data will be erased from the Target Model prior to data transfer. Please ensure you have backed up target if necessary'. At the bottom of the dialog are three buttons: 'Help', 'Transfer Data', and 'Close'. Below the buttons is a 'Progress:' label and an empty progress bar.

3. Select *.EAP to DBMS* as the *Data Transfer Type*.
4. Enter the name of the *.EAP* file to upsize to Oracle as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Oracle Provider for OLE DB" from the list, then press *Next*.
7. On the *Data Source* details page of the connection dialog, enter database name and security details as required.

Note: See [Connect to an Oracle Data Repository](#) for more information.

8. Press *OK*.
9. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
10. Press *Transfer Data* to begin the data transfer process.

Once the process is completed, you will have upsized your model to Oracle and can now open it from EA.

8.6.4 Upsizing to PostgreSQL

Before you set up EA for use with PostgreSQL, we recommend you run the *Tools | Data Management | Data Integrity* tool on the base project you wish to upsize to PostgreSQL. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for PostgreSQL. Follow them in order:

Stage One: Install PostgreSQL Components

1. Install PostgreSQL - version 7.3.2 or higher.
2. Install psqLODBC - version 7.03.01.00 or higher.
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Set up a PostgreSQL ODBC Driver](#).

Stage Two: Configure the Database

1. From the psql command line, or using a tool such as EMS PostgreSQL Manager, load the Postgres_Basemodel.sql file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
2. Run the script to create all required tables/indexes etc.

Note: See [Create a New PostgreSQL Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the *Tools | Data Management | Data Transfer* menu option to transfer an existing model into the server.

1. Open EA.
2. Go to the *Tools | Data Management | Data Transfer* menu option. This will open the *Full Model Transfer Dialog*.

Data Transfer Type

.EAP to .EAP DBMS to .EAP .EAP to DBMS DBMS to DBMS

Source and Target Models

Source Model: ...

Target Model: ...

Logfile

Logfile ...

Caution: All data will be erased from the Target Model prior to data transfer. Please ensure you have backed up target if necessary

Help Transfer Data Close

Progress:

3. Select .EAP to DBMS as the *Data Transfer Type*.
4. Click the Browse [...] button to the right of the *Source Model* and locate the .EAP file to upsize to PostgreSQL.
5. Click the Browse [...] button to the right of the *Target Model*. This will open the *Datalink Properties* Dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press the *Next* button.
7. From the *Use data source name* drop down list, select the ODBC Data Source you configured to point to your new database.

Note: See [Connect to a PostgreSQL Data Repository](#) for more information.

8. Press *OK*.
9. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
10. Press the *Transfer Data* button to begin the data transfer process.

Once the process is completed, you will have upsized your model to PostgreSQL and can now open it from EA

8.6.5 Upsizing to Sybase Adaptive Server Anywhere (ASA)

Before you set up EA for use with ASA, we recommend you run the [Tools | Data Management | Data Integrity](#) tool on the base project you wish to upsize to ASA. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for ASA. Follow them in order:

Stage One: Install ASA Components

1. Install Adaptive Server Anywhere - SQL Anywhere Studio 8 or higher. This will also install the ASA ODBC driver.
2. Create a new database for the EA repository using Sybase Central.
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Set up an ASA ODBC Driver](#).

Stage Two: Configure the Database

From Sybase Central, right click on the newly created database and open Interactive SQL. Load the ASA_BaseModel.sql file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.

Run the script to create all required tables/indexes etc.

Note: See [Create a New ASA Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the Tools | Data Management | Data Transfer menu option to transfer an existing model into the server.

1. Open EA.
2. Go to the **Tools | Data Management | Data Transfer** menu option. This will open the **Full Model Transfer Dialog**.

Data Transfer Type

.EAP to .EAP DBMS to .EAP .EAP to DBMS DBMS to DBMS

Source and Target Models

Source Model: ...

Target Model: ...

Logfile

Logfile ...

Caution: All data will be erased from the Target Model prior to data transfer. Please ensure you have backed up target if necessary

Help Transfer Data Close

Progress:

3. Select .EAP to DBMS as the *Data Transfer Type*.
4. Click the Browse [...] button to the right of the *Source Model* and locate the .EAP file to upsize to ASA.
5. Click the Browse [...] button to the right of the *Target Model*. This will open the *Datalink Properties* Dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press the *Next* button.
7. From the *Use data source name* drop down list, select the ODBC Data Source you configured to point to your new database.

Note: See [Connect to a ASA Data Repository](#) for more information.

8. Press *OK*.
9. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
10. Press the *Transfer Data* button to begin the data transfer process.

Once the process is completed, you will have upsized your model to Adaptive Server Anywhere and can now open it from Enterprise Architect.

8.6.6 Upsizing to SQL Server Desktop Engine (MSDE)

Before you set up EA for use with MSDE, we recommend you run the [Tools | Data Management | Data Integrity](#) tool on the base project you wish to upsize to MSDE. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

Follow the steps in [Upsizing to SQL Server](#) to upsize your model to MSDE.

8.6.7 Upsizing to Progress OpenEdge

Before you set up EA for use with OpenEdge, we recommend you run the [Tools | Data Management | Data Integrity](#) tool on the base project you wish to upsize to OpenEdge. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for OpenEdge. Follow them in order:

Stage One: Install OpenEdge Components

1. Install OpenEdge - version 10.0B3 or higher.
2. Install OpenEdge ODBC 10.0B or higher driver
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Setup a Progress OpenEdge ODBC Driver](#)

Stage Two: Configure the Database

1. Create an empty OpenEdge database, using the scripts *OpenEdge_BaseModel.sql* file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.html.
2. Make sure the new database is selected as the current database.
3. Run the script to create all required tables/indexes etc.

Note: See [Create a New OpenEdge Repository](#)

Stage Three: Transfer the Data

1. Open EA (you can press *Cancel* at the *Open Project* screen to open with no project loaded).
2. From the *Tools | Data Management* submenu, select *Data Transfer*. This will open the *Full Model Data Transfer* dialog:

The screenshot shows the 'Full Model Data Transfer' dialog box. It is divided into three main sections:

- Data Transfer Type:** Contains four radio buttons. The third option, '.EAP to DBMS', is selected.
- Source and Target Models:** Contains two text boxes. The 'Source Model' field contains the path 'C:\Documents and Settings\John\Desktop\EA Models\EAne'. The 'Target Model' field contains the connection string 'DBType=0;Connect=Provider=MSDASQL.1;Persist Security In'. Both fields have a browse button ('...') to the right.
- Logfile:** Contains a checked checkbox labeled 'Logfile' and a text box containing the path 'C:\Temp\Example_to_MYSQL.log'. There is also a browse button ('...') to the right.

At the bottom of the dialog, there is a caution message: 'Caution: All data will be erased from the Target Model prior to data transfer. Please ensure you have backed up target if necessary'. Below this are three buttons: 'Help', 'Transfer Data', and 'Close'. At the very bottom, there is a 'Progress:' label followed by an empty progress bar.

3. Select *.EAP to DBMS* as the *Data Transfer Type*.
4. Enter the name of the *.EAP* file to upsize to OpenEdge as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press *Next*.
7. Select the ODBC Data Source you configured to point to your new database.

Note: See [Connect to a OpenEdge Data Repository](#) for more information.

8. Press **OK**.
9. If desired, check the **Logfile** checkbox and enter a path for the data transfer log file.
10. Press **Transfer Data** to begin the data transfer process.

Once the process is completed, you will have upsized your model to OpenEdge and can now open it from EA.

8.7 Model Maintenance

This section highlights some of the administrative functions you may need to carry out to maintain your model.

See Also

- [Rename a Project](#)
- [Compact a Project](#)
- [Repair a Project](#)

8.7.1 Rename a Project

Important: *The only way to rename an Enterprise Architect project is at the Windows file system level.*

To rename an EA project, follow these steps.

1. If you have the project open, shut it down.
2. Ensure no other users have the file open.
3. Open Windows Explorer and navigate to the project.
4. Rename the project file using Windows Explorer.
5. You should *keep the ".EAP" extension the same* to preserve compatibility with the default project type as installed in the registry at installation time.

See Also

- [Model Maintenance](#)
- [Compact a Project](#)
- [Repair a Project](#)

8.7.2 Compact a Project

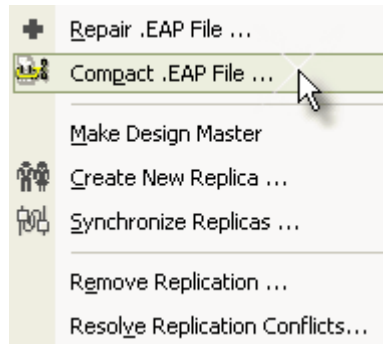
After some time, a project may benefit from *compacting* to conserve space.

Note: *Compacting will shuffle the contents of the model around, eliminating unused space and generally reducing the size of your model file.*

Compact a Project

To compact a project, follow these steps:

1. Ensure that no users have the target project open.
2. Select the *Tools | Manage .EAP File | Compact .EAP File* menu option to compact the selected project.



3. Follow the on-screen instructions to complete the process.

Warning: Always compact and repair projects on a *local* drive, never on a network drive.

See Also

- [Model Maintenance](#)
- [Rename a Project](#)
- [Repair a Project](#)

8.7.3 Repair a Project

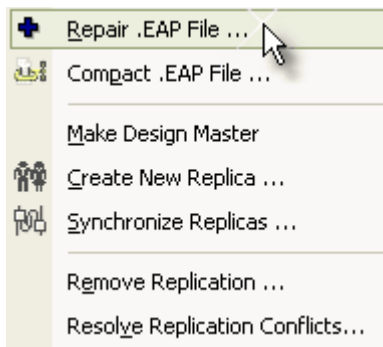
If a project has not been closed properly, often caused by system or network outages, on rare occasions the .EAP file will not open correctly. In this case you will get a message informing you the model is of an unknown database format or is not a database file.

Note: Poor network connections may also cause this symptom.

Repair a Project that has not closed correctly

To repair a project which was not closed properly, follow these steps:

1. Copy the project file to a local drive on your PC.
2. Start Enterprise Architect and open a place holder model to enable access to the "Repair .EAP File" facility.
NOTE: This is NOT the model you intend to repair.
3. Select *Tools | Manage .EAP File | Repair .EAP File* from the main menu.
4. Follow the on-screen instructions.



Note: All users must be logged off the project you are attempting to repair.

Warning: Never attempt to repair a project over a network connection: copy it to a local drive first.

Ensuring the Integrity of the Repaired Project

An additional step which you can use to ensure the integrity of your model is use the "Remove Replication" feature.

1. Open Enterprise Architect, but when you are prompted to open a project, press Cancel.
2. Select **Tools | Manage .EAP File | Remove Replication** from the main menu.
3. Follow the prompts, when you are prompted for the Replica Project browser for your problem project, you may be given a warning about the project not being the "Design Master", accept this warning . Click Next.
4. Browse for the clean project (i.e. EABase.eap). Click Next.
5. Enter the path and name of the new project to created, then click Next.
6. Click Run and the removal process will be run.

Once the removal process has been completed, open the project and do a check of the project contents. If the data is intact, backup the old project and replace it with the new version.

See Also

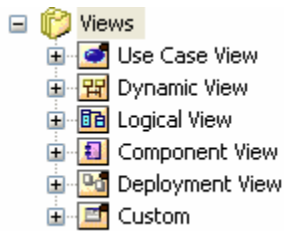
- [Model Maintenance](#)
- [Rename a Project](#)
- [Compact a Project](#)

8.8 Manage Views

The top level packages in EA are referred to as **Views**. This terminology is used simply to designate that the package is at the top level and may be used to subdivide a project into partitions such as Business Process, Logical Model, Dynamic View, etc.

There are 6 main views:

- [Use Case View](#) - eg. [Use Case diagram](#), [Analysis diagram](#), [Robustness diagram](#)
- [Dynamic View](#) - eg. [Activity diagram](#), [Communication diagram](#), [Sequence diagram](#), [State diagram](#)
- [Logical View](#) - eg. [Class Model](#), [Code Engineering](#), [Data Model](#)
- [Component View](#) - eg. [Component diagram](#)
- [Deployment View](#) - eg. [Deployment diagram](#)
- [Custom View](#)



You may wish to rename these views, move them into a different order, or delete the provided views and create your own. Do this by right clicking the mouse on the selected View to open the context menu, and choose whether to rename, move or delete the view.

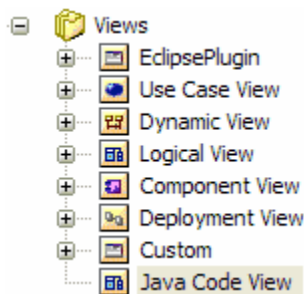
See Also

- [Add Additional Views](#)
- [Rename Views](#)
- [Delete Views](#)

8.8.1 Add Additional Views

You may add additional packages at the **View** level. These will appear at the end of the standard views and be displayed in creation order. The icon for user created views is the same as for the Custom view. User views are a good way to extend the standard model on a per project basis depending on specific requirements and modeling techniques.

The example below shows an additional view called "Java Code View" which has been appended to the main Views list.

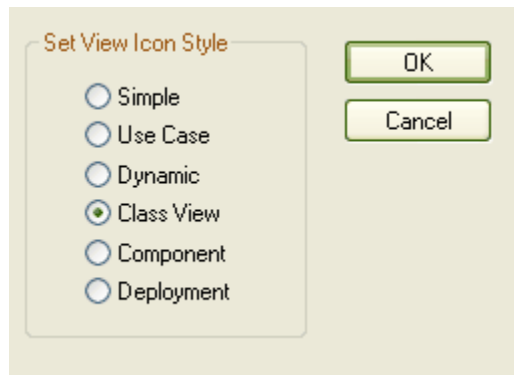


Note: You can delete a user created view. If you do so, ALL contents of the view are deleted as well, so take care.

Create a User View

To create a user view, follow these steps:

1. Right click on the root element of the Project Browser (named **Views**).
2. Select the **New View** option from the menu.
3. Give your view a name and press **OK**.
4. The **Set View Icon** dialog will then allow the user to specify the view icon, select the most appropriate one from the list and then press the **OK** button.



8.8.2 Rename Views

You can rename a view if you wish.

Rename a View

To rename a view, follow these steps:

1. Right click on the view in the Project Browser. Select the *Package Properties* option.



2. Enter the new name and press *OK*.

General Require Constraints Link Scenario Files

Name: Business Modeling View

Stereotype: Abstract

Author: Status: Proposed

Scope: Public Complexity: Easy

Alias: Language: C++

Keywords:

Phase: 1.0 Version: 1.0

Note:

Apply OK Cancel Help

8.8.3 Delete Views

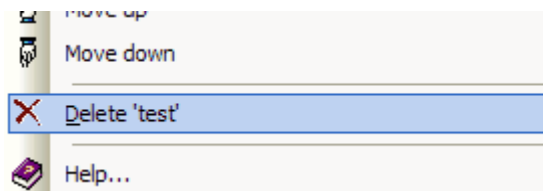
You can delete a view if you wish.

Warning: If you delete a view, all its contents will be deleted at the same time. It CANNOT be restored.

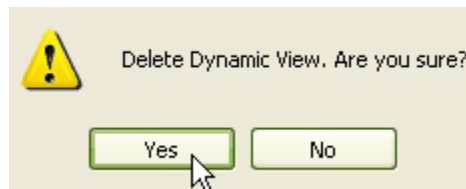
Delete a View

Follow these steps if you wish to delete a view.

1. Right click on the view you wish to delete in the *Project Browser*.
2. Select *Delete* from the context menu.



3. You will see the following warning:



4. If you wish to delete the view and its contents, press **Yes**. If not, press **No**.

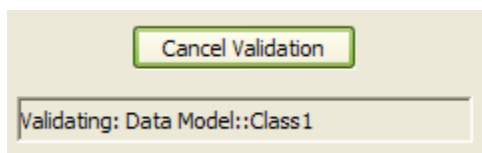
8.9 Model Validation

Model Validation is used to validate UML models against known UML rules, as well as any constraints defined within the model using the Object Constraint Language (OCL) . Model Validation can be run against a single UML element a diagram or an entire package. The details of the Model Validation process, against the item being validated, is defined below:

- Validating a UML element will validate the element and its children, features (attributes & operations) and relationships (connectors & links).
- Validating a UML diagram will validate the diagram itself (for correctness) as well as any elements and connectors within it.
- Validating a UML package will validate the package, subpackages and all elements, connectors and diagrams within it.

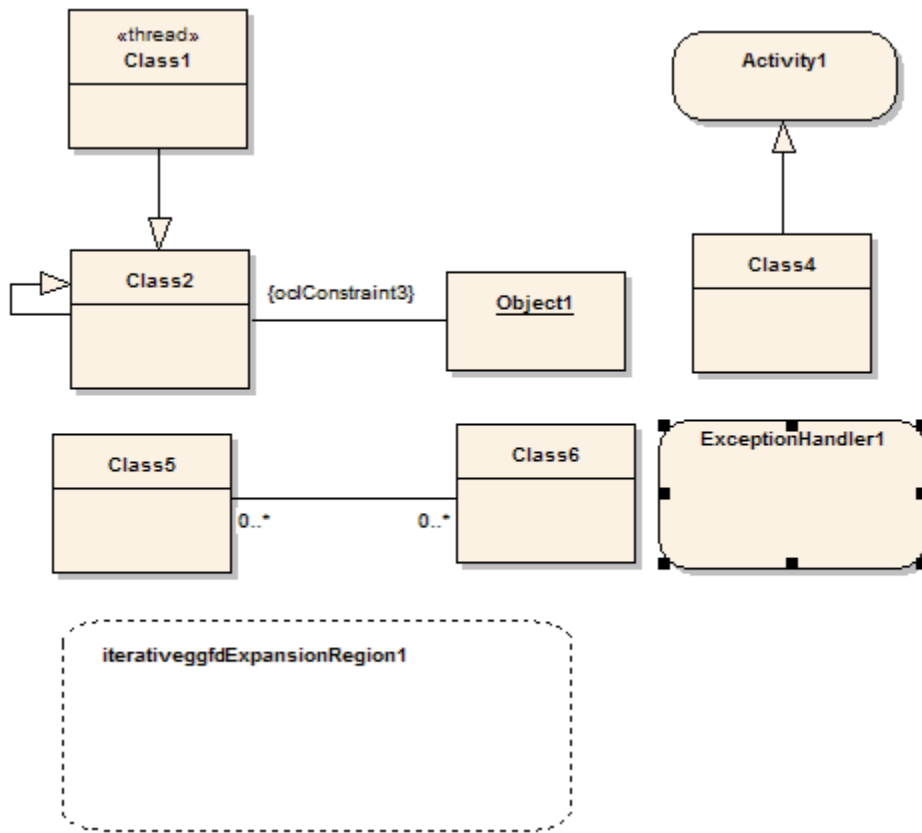
To use the Model Validation use the following steps:

1. Select the Package, Diagram or Element either from **Project View**, or within an open diagram.
2. Select **Project | Model Validation | Validate Selected** from the main menu. [Ctrl+Alt+V]
3. EA will perform the validation process. Results are displayed in the Output window (the output window can be accessed using the **View | Output** menu command).
A progress window is also displayed with a cancel button, allowing you to cancel the validation process at any time, additionally you can also use the **Project | Model Validation | Cancel Validation** command.
- 4.

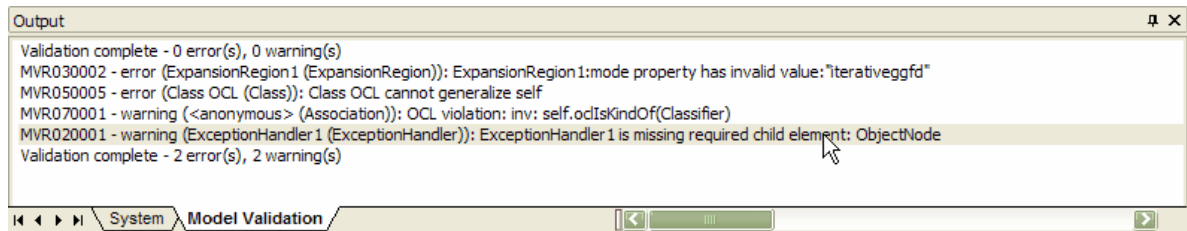


Example Model Violations.

The following diagram demonstrates a UML diagram containing several basic violations.



Running the Model Validation process on this diagram reveals the following violations in the Output Window:



From the output listing you can see that the diagram:

- Contains a UML ExpansionRegion (ExpansionRegion1) with an invalid value for its "mode" property (in this case, the valid values are "iterative, parallel or stream")
- Contains an invalid self-generalization on Class2 (UML Elements cannot be self-generalized)
- Contains an OCL violation for the anonymous association (between Class2 and Object4)
- Contains a UML ExceptionHandler (ExceptionHandler1) that is missing its child input ObjectNode

Note: By double clicking on an error in the Output window, the element in the diagram will become selected.

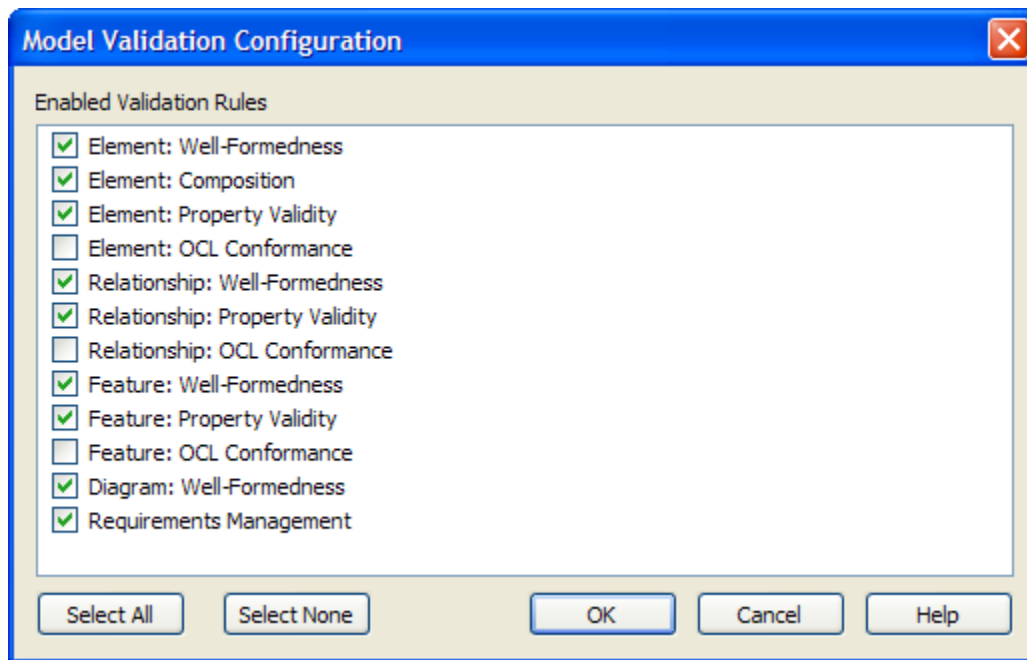
See Also

- [Configuring Model Validation](#)
- [Rules Reference](#)

8.9.1 Configuring Model Validation

The Model Validation Configuration dialog is used to enable and disable the rules that are run with the model validator. Additional rules may be defined in this dialog from any additional addins that may be installed alongside EA. Additionally, specific Model Validation settings can be assigned to a Perspective, for more information regarding Perspectives and their use please refer to the [Configure Perspectives](#) topic.

The Model Validation Configuration dialog can be accessed via the main menu by selecting *Project | Model Validation | Configure*.



Tips & Tricks: If you want to disable UML syntax ("The requested connection is not UML compliant"), this option can be disabled from *Tools | Options | Diagram | General | Strict UML Syntax*

See Also

- [Model Validation](#)
- [Rules Reference](#)

8.9.2 Rules Reference

The Model Validation feature works against a set of validation rules arranged in the following groups:

- [\(Element, Relationship, Feature, Diagram\): Well-Formedness](#)
Responsible for checking whether or not an Element, Relationship, Feature or Diagram is well-formed. This group of rules includes checks such as whether item is a valid UML item and whether a diagram contains valid elements within it
- [Element: Composition](#)
Responsible for checking whether or not a UML Element contains valid children, whether it contains the right number of valid children, and whether or not the element is missing any required children.
- [\(Element, Relationship, Feature\): Property Validity](#)
Responsible for checking whether or not the item in question has the correct UML properties defined for it and whether they contain incorrect or conflicting values. For more information about these properties refer to the Custom Properties section.
- [\(Element, Relationship, Feature\): OCL Conformance](#)
Responsible for validating an item against any defined constraints in OCL.

8.9.2.1 Well-Formedness (Element, Relationship, Feature, Diagram)

This group of rules is responsible for checking whether or not an Element, Relationship, Feature or Diagram is well-formed. This group of rules includes checks such as whether item is a valid UML item and whether a diagram contains valid elements within it. Reported violations include:

| Violation ID | Description | Information |
|---------------------|---|---|
| MVR010001 | <<Element>> is not a valid UML Element. | The element is not a recognized UML 2.0 element |
| MVR050001 | <<Relationship>> is not a valid UML Relationship | The relationship is not a recognized UML 2.0 relationship |
| MVR050002 | <<Relationship>> is not legal for <<Start Element>> --> <<End Element>> | The relationship between the given start and end elements is not valid for those elements. |
| MVR050003 | <<Parent Element>>:isLeaf=true and cannot be generalized by <<Child Element>> | The generalization relationship cannot exist between parent and child elements because the parent element is defined as a Leaf element. |
| MVR050004 | <<Child Element>>:isRoot=true and cannot generalize <<Parent Element>> | The generalization relationship cannot exist between parent and child elements because the child element is defined as a Root element. |
| MVR050005 | <<Element>> cannot generalize self | The element cannot be self-generalized |
| MVR0B0001 | Statechart violation: <<extended information>> | The state diagram contains a UML violation, refer to the extended information for more information about the detected violation |

See Also

- [Model Validation](#)
- [Configuring Model Validation](#)
- [Rules Reference](#)

8.9.2.2 Element Composition

This group of rules is responsible for checking whether or not a UML Element contains valid children, whether it contains the right number of valid children, and whether or not the element is missing any required children.

| Error ID | Description | Information |
|-----------------|---|---|
| MVR020001 | <<Element>> is missing required child element <<Child Element>> | The element is missing a child element of type "Child Element" |
| MVR020002 | Invalid UML package child. | The element cannot be a direct package child and must be a child of another element (for example: Ports must be children of other elements, and not direct UML package members) |
| MVR020003 | Invalid child <<Child Element name>> (<<Child Element Type>>). | The child element is invalid on the tested parent element |

See Also

- [Model Validation](#)
- [Configuring Model Validation](#)
- [Rules Reference](#)

8.9.2.3 Property Validity (Element, Relationship, Feature)

This group is responsible for checking whether or not the item in question has the correct UML properties defined for it and whether they contain incorrect or conflicting values. For more information about these properties refer to the [Custom Properties](#) section.

| Error ID | Description | Information |
|-----------|---|---|
| MVR030001 | <<Element>>:<<Property>> property is undefined | The element property contains no value |
| MVR030002 | <<Element>>:<<Property>> property has invalid value: "<<Value>>" | The element property contains an invalid value |
| MVR030003 | <<Element>>:isLeaf=true and cannot be abstract | The element's isLeaf and isAbstract properties are both set to true, which is invalid |
| MVR060001 | <<Relationship>>:<<Property>> property is undefined | The relationship property contains no value |
| MVR060002 | <<Relationship>>:<<Property>> property has invalid value: "<<Value>>" | The relationship property contains an invalid value |

See Also

- [Model Validation](#)
- [Configuring Model Validation](#)
- [Rules Reference](#)

8.9.2.4 OCL Conformance (Element, Relationship, Feature)

This group is responsible for validating an item against any defined constraints in OCL. The Object Constraint Language is used to describe expressions on UML Models. OCL is used to express side-effect free constraints. OCL constraints may be added to any element, relationship or attribute in EA.

| Error ID | Description | Information |
|-----------|---------------------------------|---|
| MVR040001 | OCL violation: <<violated OCL>> | The element violates the OCL constraint specified. |
| MVR070001 | OCL violation: <<violated OCL>> | The relationship violates the OCL constraint specified. |
| MVR0A0001 | OCL violation: <<violated OCL>> | The attribute violates the OCL constraint specified. |

Defining OCL Constraints for an Element

OCL constraints may be added to an element via the *Properties* dialog. Click on the *Constraints* tab and select *OCL* from the *Type* drop down list. It is important to note that in order to have a valid OCL constraint requires that the syntax be correctly formed, if the expression is not correct EA will inform the user that the OCL constraint is not valid. To perform an OCL Validation, ensure that Constraint (OCL) Rules is selected in the Model Validation Configuration Dialog, OCL violations are recorded in the Model Validation output window.

Constraint:

oclConstraint Type: OCL Status: Approved

inv: self.oclIsKindOf(DirectedRelationship)

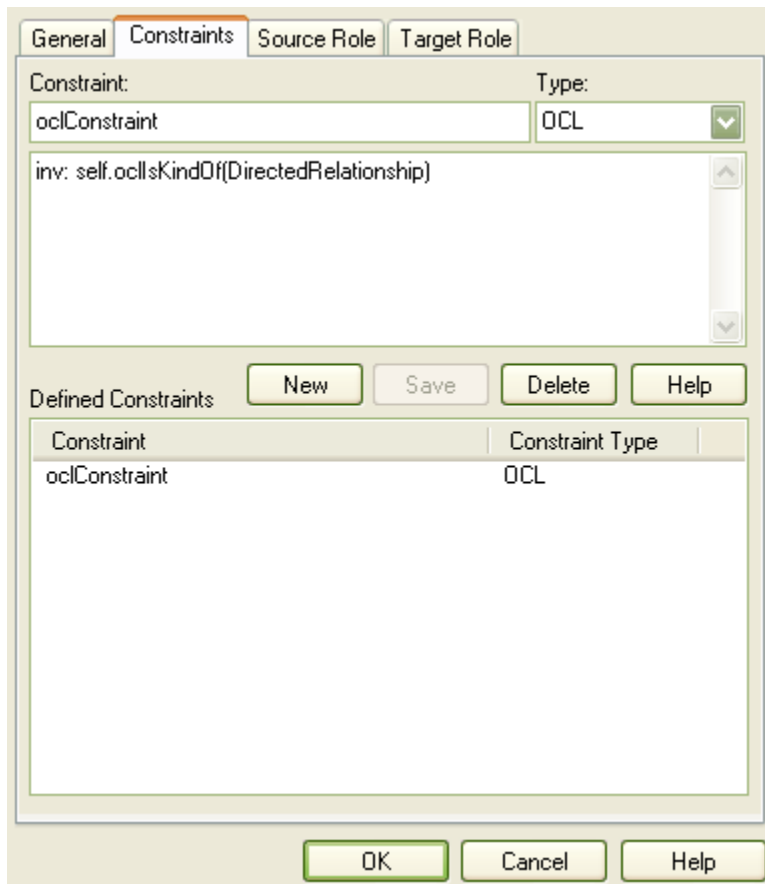
Defined Constraints New Save Delete

| Constraint | Type | Status |
|---------------|------|----------|
| oclConstraint | OCL | Approved |

Apply OK Cancel Help

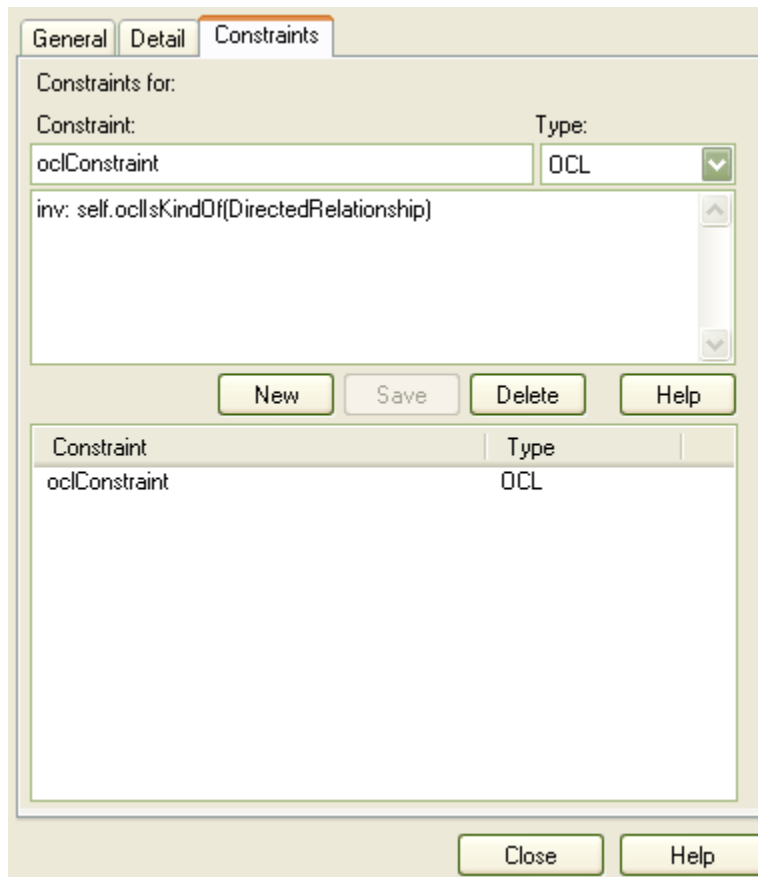
Defining OCL Constraints for a Relationship

OCL constraints may be added to a relationship via the *Properties* dialog. Click on the *Constraints* tab and select *OCL* from the *Type* drop down list. It is important to note that in order to have a valid OCL constraint requires that the syntax be correctly formed, if the expression is not correct EA will inform the user that the OCL constraint is not valid. To perform an OCL Validation, ensure that Constraint (OCL) Rules is selected in the Model Validation Configuration Dialog, OCL violations are recorded in the Model Validation output window.



Defining OCL Constraints for a Feature (Attribute)

OCL constraints may be added to a feature via the *Properties* dialog. Click on the *Constraints* tab and select *OCL* from the *Type* drop down list. It is important to note that in order to have a valid OCL constraint requires that the syntax be correctly formed, if the expression is not correct EA will inform the user that the OCL constraint is not valid. To perform an OCL Validation, ensure that Constraint (OCL) Rules is selected in the Model Validation Configuration Dialog, OCL violations are recorded in the Model Validation output window.



See Also

- [Model Validation](#)
- [Configuring Model Validation](#)
- [Rules Reference](#)

8.10 MOF

EA offers support for exporting packages to XML under the MOF 1.3 and MOF 1.4 standards. MOF models are created by assigning the stereotype *metamodel* to the package. MOF models may be exported to MOF 1.3 or MOF 1.4 XML file specification.

Background Knowledge

MOF, (Meta-Object Facility) is an Object Management Group (OMG) standard. MOF originated in the (UML), when the OMG was in need of a Meta-Modeling architecture to define the UML. MOF is designed as a four-layered architecture, for more details see the diagram below.

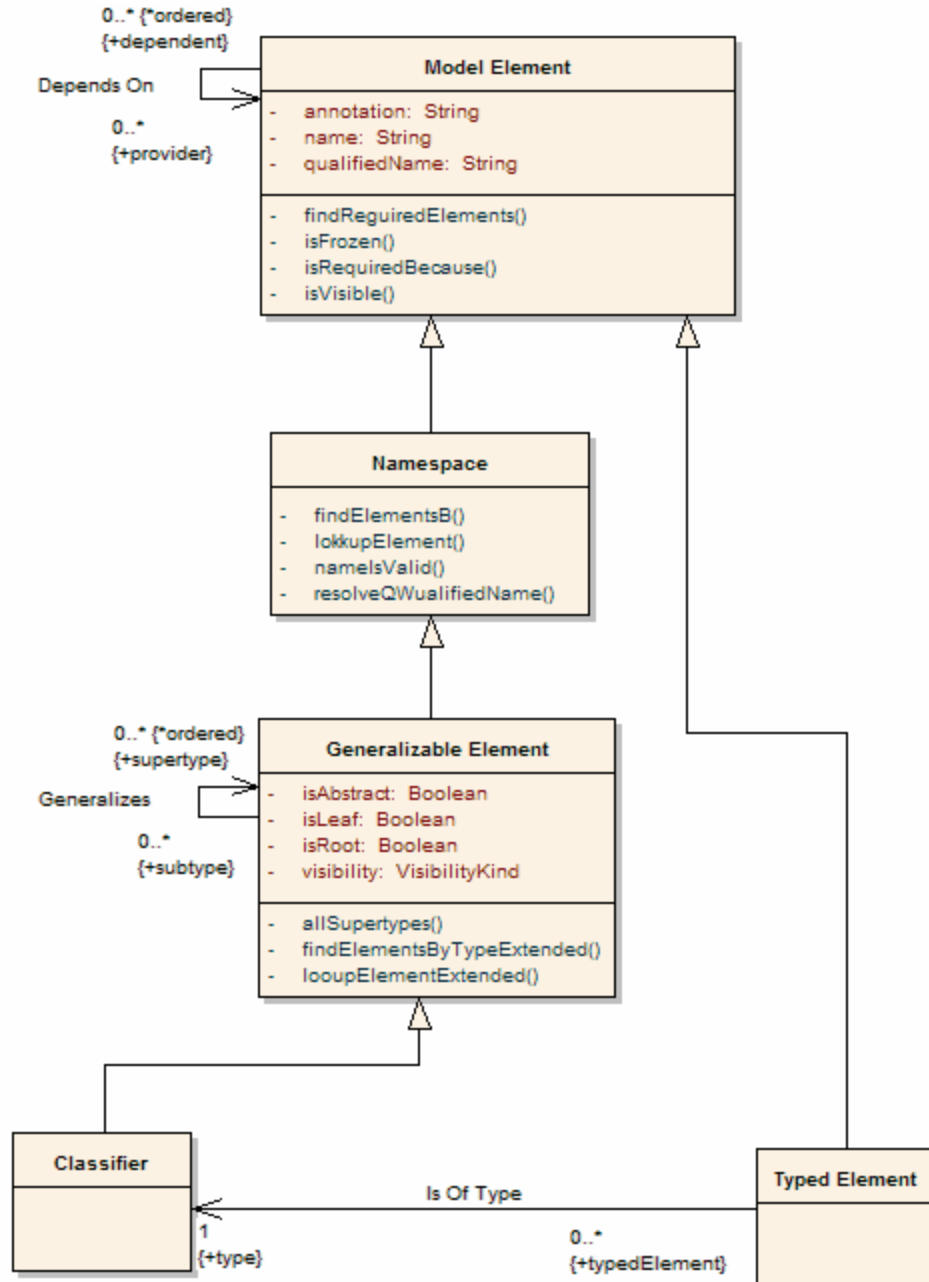
Because of the similarities between the MOF-model and UML structure models, MOF meta-models are usually modeled as UML class diagrams. A supporting standard of MOF is XML, which defines an XML-based exchange format.

MOF is a closed meta-modeling architecture. It is a strict meta-modeling architecture; every model element on every layer is strictly an instance of a model element of the layer above. MOF only provides a means to define the structure, or abstract syntax of a languages or of data.

Simplified, MOF uses the notion of classes, as known from object orientation, to define concepts (model elements) on a meta-layer. These classes (concepts) can then be instantiated through objects (instances) of the model layer below. Due to the fact that an element on the M2 layer is an object (instance of an M3 model

element) as well as a class (it is an M2 layer concept) the notion of a clabject is used. Clabject is a merge of the words class and object.

Another related standard is OCL which describes a formal language that can be used to define model constraints by means of predicate logic.



See Also

- [Getting Started](#)
- [Export MOF to XMI](#)
- [Importing MOF to XMI](#)
- [Toolbox-Metamodel](#)

8.10.1 Getting Started

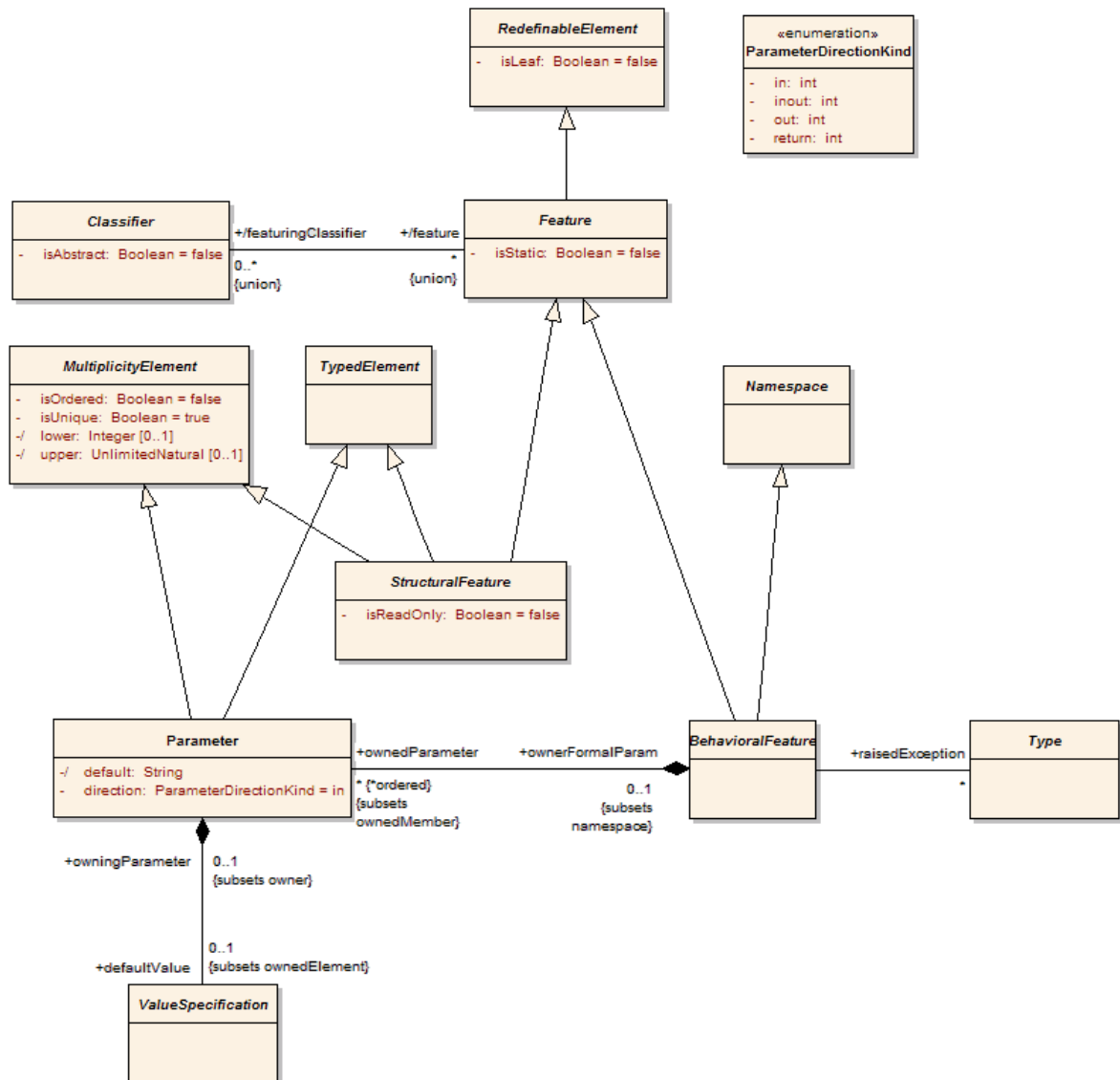
MOF diagrams are class diagrams that are contained in packages with a metamodel stereotype. To create a MOF diagram use the steps detailed below.

1. Create a package that will contain your MOF elements, then right click on the package in the Project View to bring up its context menu, select *Package Properties*. Navigate to the General Tab of the Package dialog and in the Stereotype field enter the value metamodel.
- 2.

The screenshot shows the 'Package Properties' dialog box with the 'General' tab selected. The 'Name' field contains 'Kernel'. The 'Stereotype' dropdown menu is open, showing 'metamodel' selected. The 'Abstract' checkbox is unchecked. The 'Author' dropdown is 'Neil Capey', 'Status' is 'Proposed', 'Scope' is 'Public', 'Complexity' is '*Easy', and 'Language' is 'Java'. The 'Phase' is '1.0' and 'Version' is empty. The 'Notes' field is empty. At the bottom, there are buttons for 'Apply', 'OK', 'Cancel', and 'Help'.

3. Create a class diagram. To do this, right click on the package in the Project View and select *New Diagram*, the default diagram will be a class diagram. Give your MOF diagram a name.
4. Add MOF elements from the [Metamodel](#) section of the UML Toolbox.

The following is an example of a MOF diagram. A MOF diagram can contain Packages, Classes, Enumerations, Primitives, Generalizations, Associations, Composite and Aggregates.

**See Also**

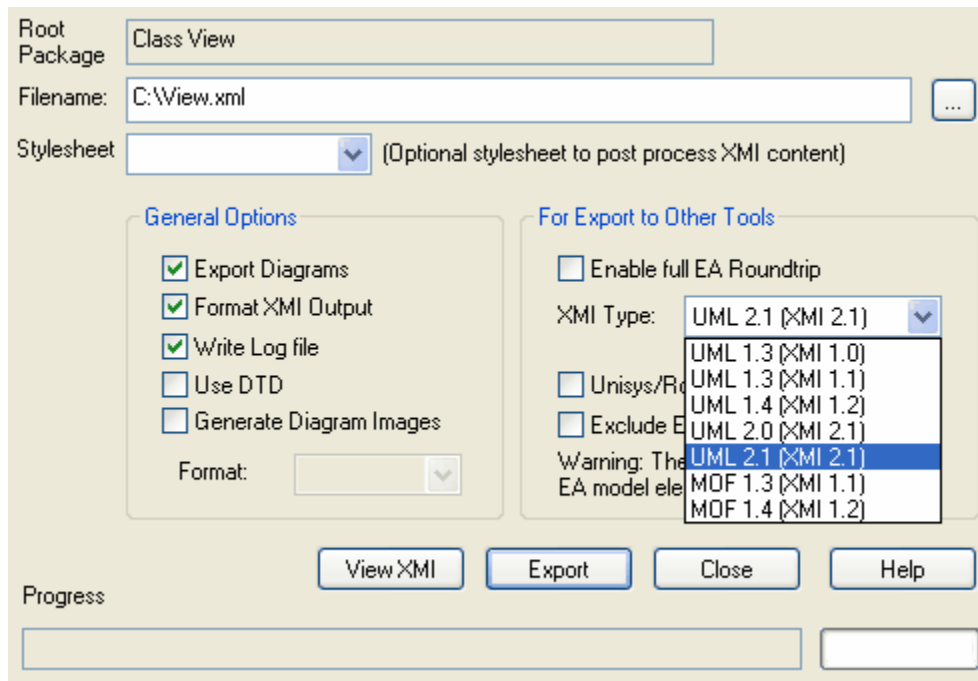
- [MOF](#)
- [Export MOF to XMI](#)
- [Importing MOF to XMI](#)
- [Toolbox-Metamodel](#)

8.10.2 Export MOF to XMI

Once you have created your MOF diagram, you will be able to export the diagram to XMI specifying the MOF 1.3 or MOF 1.4 standard.

1. Open the **Export Package to XMI** dialogue. To do this, right click on the package in the project view and select **Import/Export | Export Package to XMI file...**

The follow dialog will appear. De-select **Enable full EA Roundtrip** and select **MOF 1.3** or **MOF 1.4** from the pull down menu, don't forget to give your XMI file a name.



2. Click the **Export** button and wait until the Progress bar reads 100%.
3. Once your file has been created, you will be able to view the XMI file by pressing the **View XMI** button.

See Also

- [MOF](#)
- [Getting Started](#)
- [Importing MOF to XMI](#)
- [Toolbox-Metamodel](#)

8.10.3 Import MOF from XMI

MOF diagrams exported to XMI are capable of being imported via the regular import XMI features of EA. See [Import from XMI](#)

See Also

- [MOF](#)
- [Getting Started](#)
- [Export MOF to XMI](#)
- [Toolbox-Metamodel](#)

8.11 XMI Import and Export

What is XMI?

XML Metadata Interchange (XMI) is an open standard file format, intended to allow the interchange of model information between models and tools. XMI is based on XML, and is defined by the OMG. Enterprise Architect uses XMI as a method of importing and exporting model specifications between different UML packages, EA projects, and other tools that support XMI.

Enterprise Architect supports the XMI 1.1, 1.2 and 2.1 specifications, but does not fully support the older 1.0 specification. When importing/exporting to XMI 1.0, some loss of data will occur due to the limitations of XMI 1.0. XMI Version 1.1 has support for UML 1.3 whereas XMI 2.1 has support for UML 2.0 and UML 2.1.

With XMI the transfer of model details may be exchanged between different UML tools and other tools that are

capable of using XMI. Limited support for export to Rational Rose is provided using the Rose version of the XMI 1.1 specification, as implemented by Unisys for Rational products. Packages may be [exported](#) from and [imported](#) into EA models. This greatly improves the flexibility and robustness of EA models by allowing Analysts and Modelers to externalize model elements in XMI for [version control](#), distributed development, post processing and transferring packages between models. When performing EA-to-EA transfers, ensure that the XMI version selected is 1.1 or 2.1.

See Also

- [XML Options](#) configure the XML options. XMI import/export and package control all rely on saving and loading XML files. There are some options you can set to streamline this process.
- [Export a package](#) to XMI in XMI 1.0, XMI 1.1, XMI 1.2 and XMI 2.1
- [Import from XMI](#) with support for XMI 1.0, XMI 1.1, XMI 1.2 and XMI 2.1
- [XMI controlled packages](#)
- [Manually control a package](#) by linking it to an XMI file
- [Batch export](#) controlled packages
- [Batch import](#) controlled packages
- [Limitations of XMI](#)
- [The UML DTD](#)

Important Note: The UML_EA.DTD file MUST be present in the output directory where XML files are written when the *Use DTD* option has been selected when performing an [XMI export](#) to XMI 1.1. If it is absent an error will occur on trying to read or write an XML file.

Important Note: When you import an XML file over an existing package - ALL information in the current package is deleted first. Please make sure you do not have important changes you do not want to lose before you do this.

For further information on XMI, including specifications, consult the OMG's [XML/XMI Technology page](#)

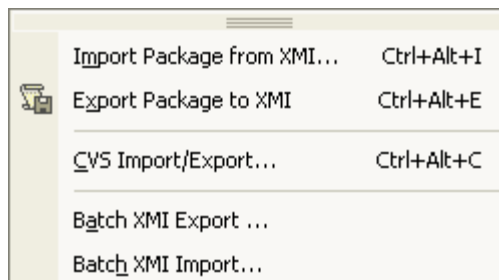
8.11.1 Export to XMI

A Package may be exported to an XMI (XML based) file. This allows EA Model elements to be moved between models, for [distributed development](#), [manual version control](#) and other benefits. It also allows limited export of EA model elements to Rational Rose and other tools which implement the UML 2.1 XMI 2.1 or the UML1.3 XMI 1.2 / XMI 1.1 / XMI 1.0 standard. For more information regarding the limitations of XMI exporting read the [Limitations of XMI](#) topic.

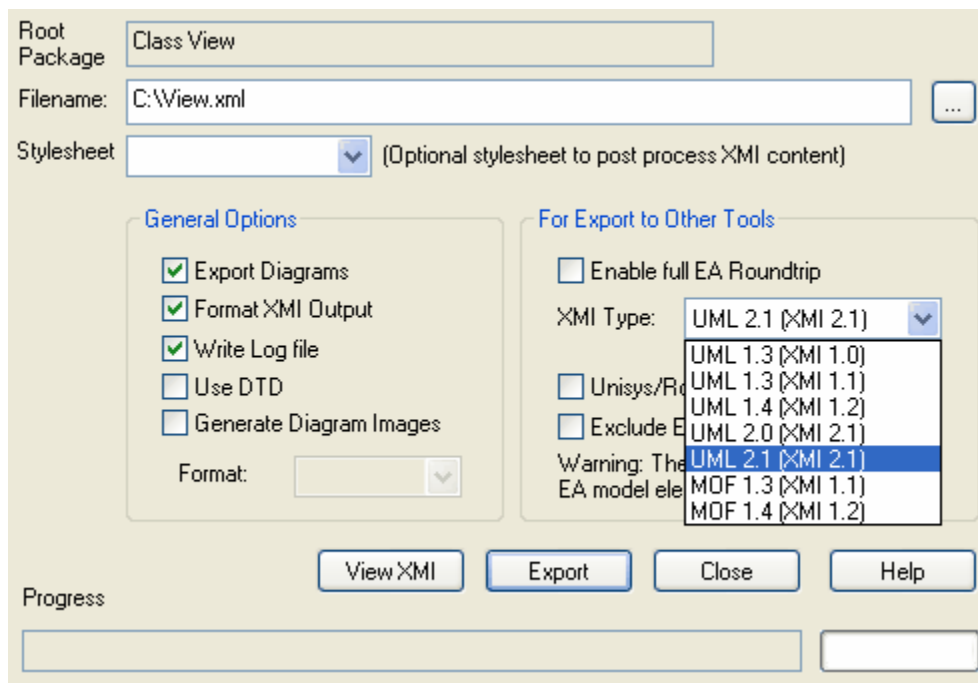
Exporting a Package to XML

To export a package to XML, follow these steps:

1. In the Project Browser, select the package you want to export.
2. Right click and select the *Import/Export* menu OR go to the *Project | Import/Export* submenu. Select the *Export Package to XMI* option.



3. The *Export Package to XMI* dialog will appear.



4. Set the required options - you can configure the following:
 - Filename - where to output the XML file - must be a valid directory/path name.
 - Stylesheet - select a stylesheet to post-process XML content before saving to file.
 - Export diagrams yes/no - check to export diagrams.
 - Use Unisys Rose Format - check to export in Rose UML 1.3, XMI 1.1 format.
 - Format XML output - yes/no - check to format output into readable XML (takes a few additional seconds at end of run).
 - Write log file - check to write a log of export activity (recommended). The log file will be saved in the same directory exported to.
 - Use DTD - check to use the UML1.3 DTD (recommended). Using this option will validate the correctness of the model and that no syntactical errors have occurred. For more information regarding the use of DTDs read the [UML DTD](#) topic.
 - XMI 1.0 - Select this to generate output in XMI 1.0 format.
 - XMI 1.2 - Select this to generate output in XMI 1.2 format.
 - XMI 1.1 - Select this to generate output in XMI 1.1 format.
 - XMI 2.1 - Select this to generate output in XMI 2.1 format.
 - Exclude EA Tagged Values - excludes EA specific information from the export to other tools.
5. Press **Export**.

Important Note: When exporting and importing with XMI 1.0 with EA some loss of data will occur due to the limitations of XMI 1.0.

Important Note: The UML_EA.DTD file MUST be present in the output directory where XML files are written when the Use DTD option has been selected when performing an XMI export using XMI 1.1. If it is absent an error will occur on trying to read or write an XML file.

8.11.2 Import from XMI

A Package may be imported from an XMI (XML based) file. This allows EA Model elements to be moved between models, for distributed development, [manual version control](#) and other benefits.

The following formats can be imported.

- UML 1.3 (XMI 1.0)

- UML 1.3 (XMI 1.1)
- UML 1.4 (XMI 1.2)
- UML 2.0 (XMI 2.1)
- UML 2.1 (XMI 2.1)
- MOF 1.3 (XMI 1.1)
- MOF 1.4(XMI 1.2)

In addition to the above the formats, EA can also import the RSA XMI file (*.uml2).

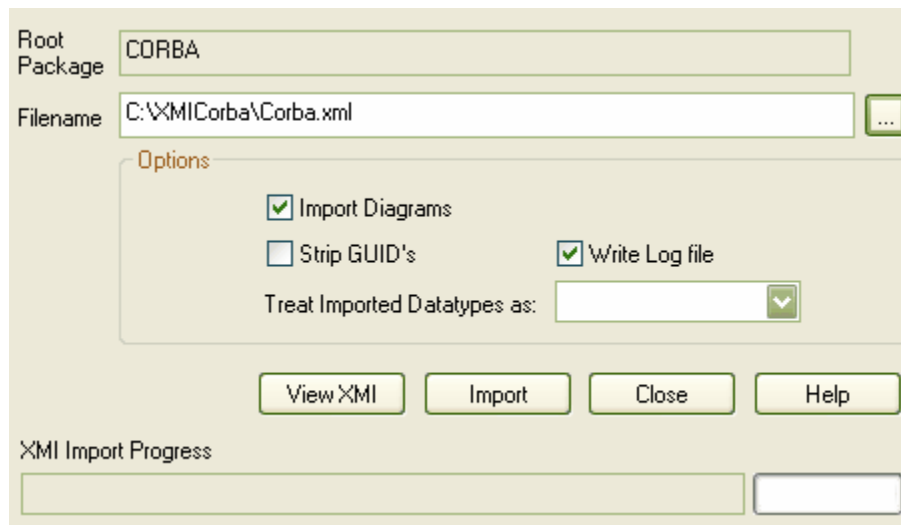
Import From XML

To import a package from XML, follow the steps below:

1. In the Project Browser, select the package you want to import.
2. Right click and select the *Import/Export* menu OR go to the *Project | Import/Export* menu. Select the *Import Package from XML* option.



3. The *Import Package from XML* dialog will appear.



4. Set the required options:
 - Filename - where to load the XML file from - must be a valid directory/path name.
 - Import diagrams - check to import diagrams.
 - Strip GUIDS - check this to remove Universal Identifier information from the file on import. This permits the import of a package twice into the same model - the second import will require new GUIDS to avoid element collisions
 - Write log file - check to write a log of import activity (recommended). The logfile will be saved in

the same directory imported from.

5. Press *Import*.

Important Note: When you import an XML file over an existing package - ALL information in the current package is deleted first. Please make sure you do not have important changes you do not want to lose before you do this.

8.11.3 Limitations of XMI

While XMI is a valuable means of specifying a UML model in a common format, it is relatively limited in the amount of additional information it will tolerate using the standard syntax. A lot of information from an EA Model must be converted to what are called 'Tagged Values' - which will import into other modeling systems as additional information or be ignored completely.

EA can both generate and read XMI 1.0, 1.1 and 1.2, using UML 1.3 format and XMI 2.1, using UML 2.0 and UML 2.1 format. Note that round tripping model elements using XMI (for example to version control or for controlled package) is only possible using XMI 1.1/UML 1.3 - EA format which uses the additional tagged values to store the UML 2.0 information.

Notes for Exporting to Rose and other tools

- There are also discrepancies in the Unisys/Rose implementation with regard to spelling mistakes and slightly different syntax to the official XMI 1.1 specification - so problems may occur.
- The way packages are arranged in different models may impact the successful import into other systems, experimentation is the only work around for this problem.
- Some parts of the XMI import/export process do not work as expected in products like Rational Rose - for example, note links are not supported, state operations import but do not appear in diagrams and some other issues. In addition, Rational Rose only supports import of a full project - not a single package
- It is recommended that for best results you keep the model elements you wish to export to Rose simple and conforming as closely as possible to the UML 1.3 specification.

8.11.4 The UML DTD

When you import or export EA packages to XML, the import or export process may be validated using a DTD or Data Type Definition. This document is used by the XML parser to validate the correctness of the model and that no syntactical errors have occurred. It is always best to use a DTD when moving packages between EA models as it ensures correctness of the XML output, and prevents attempted imports of incorrect XML.

Several DTDs for XMI/UML exist. The OMG defines a standard UML1.3 DTD for use in XMI 1.1. Enterprise Architect uses an extension to this with some additional element extensions for non-standard UML types - such as testing details.

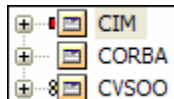
Whenever you write or read an XML file, the XML parser looks in the current directory for the DTD specified in the DOCTYPE element in the XML file itself. If the DTD cannot be found, an error occurs and processing aborts. You must ensure the UML_EA.DTD file is in the current XML output path (generated by default).

8.11.5 Controlled Packages

Controlled packages are a powerful means of 'externalizing' parts of an EA model. Using controlled packages you can:

- Support widely distributed development by having team members submit packages in the form of XML for import into a central EA repository
- Support version control by writing model elements in XML text files, suitable for version controlling using standard version control software. Using XMI in this manner allows users to manually connect to 3rd party version control software outside of the EA environment, EA internally supports the configuration of version control via SCC and CVS.
- Support import/export of model elements between different models - for example a class library may be re-used in many models, and kept up to date in target models using controlled packages - reloading packages as required when new versions of the class model become available.
- Package XML is standard XMI compliant output which may be loaded into any XML viewer, or used by any XML based tool to perform manipulations and extracts - such as document or code generators.

Controlled packages appear in the browser with a small red rectangle to the left of the package icon as in the example below:

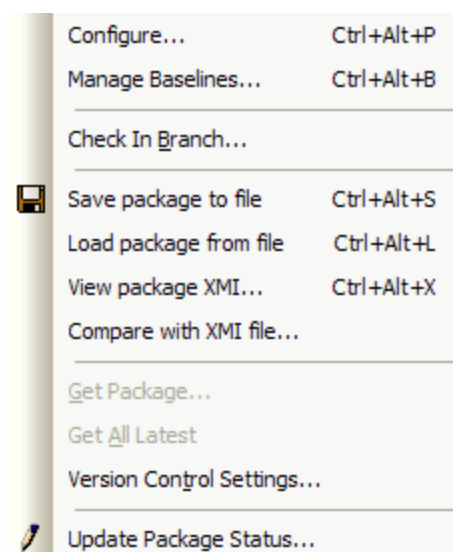


A controlled package is one configured to save and load in XML format to a named file. The XML output is UML1.3 compliant XMI, with Enterprise Architect extensions to support diagrams and additional model elements.

Important Note: The `UML_EA.DTD` file **MUST** be present in the output directory where XML files are written when the **Use DTD** option has been selected when performing an **XMI export**. If it is absent an error will occur on trying to read or write an XML file. This file contains the definition for UML1.3 XMI output based on UML models.

8.11.5.1 Controlled Package Menu

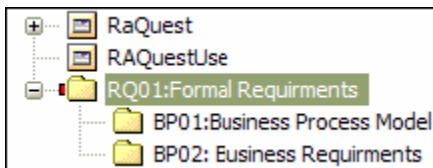
The **Controlled Package** menu is found by right clicking on a version controlled package in the Project Browser and selecting **Package Control**.



| Menu Item | Functionality |
|-------------------------|---|
| Configure | Shows the package control dialog which allows you to specify whether this package (and its children) is controlled, and which file it is controlled through. |
| Save Package to file | Saves a controlled package to an XMI file. |
| Load package from file | Loads a previously saved XMI file. |
| View package XMI.. | Allows the user to view the package XMI after the package has been exported to XMI. |
| Get Package | Allows the user to gain access from packages in the version controlled repository that is in currently available in the users model. |
| Get All Latest | Retrieves the latest version of the package from the repository. Available only for packages which are checked in. Get Latest is not intended for sharing .EAP files and should only be used when people have their own individual databases. |
| Version Control Options | Displays the Version Control Options dialog . |
| Update Package Status | Allows the user to provide a bulk update on the status of a package, this includes status options such as Proposed, Validate, Mandatory etc. |
| Set as Namespace Root | Sets the namespace root for languages which support namespaces, for more information see the Namespaces section. |

8.11.5.2 Configure Packages

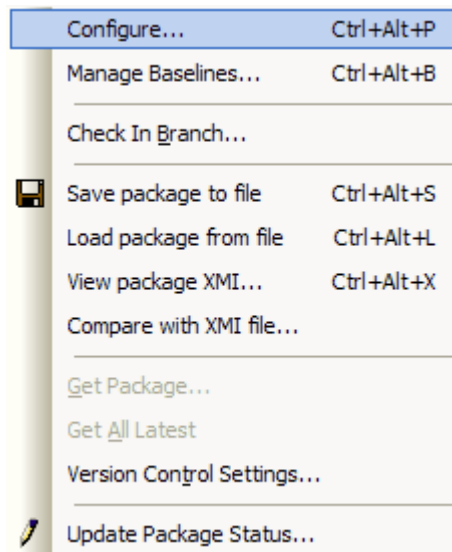
Before you can use a Controlled Package, you must configure it with some options. These include the filename to save to/load from, the type of export, version number, etc. Once a package is configured and marked as controlled, it will appear in the Project Browser with a small red rectangle next to the package icon - indicating it is a controlled package. The image below shows one controlled package called "RQ01: Formal Requirements":



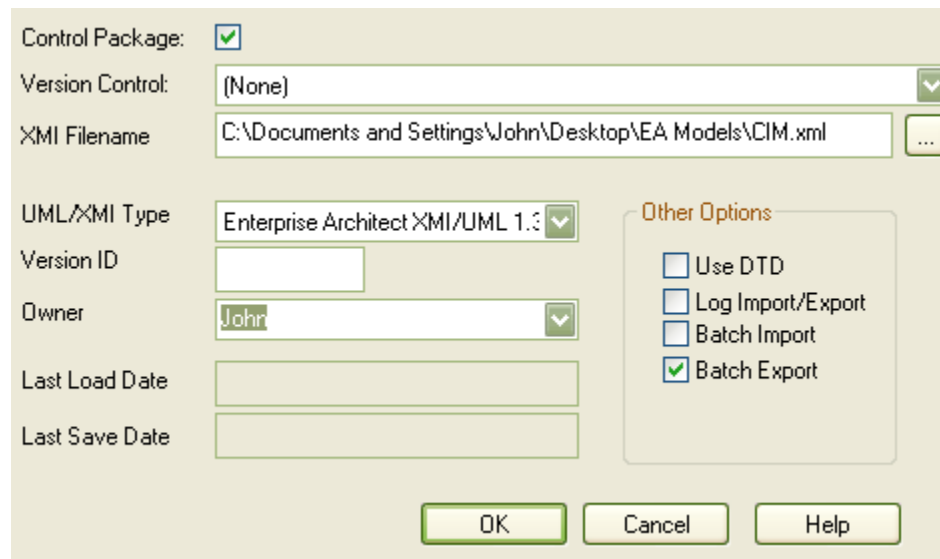
Configure a Controlled Package

To configure a controlled package, follow the steps below:

1. In the Project Browser, right click on the package you wish to control or configure.
2. From the *Package Control* submenu, select *Configure*.



3. The *Package Control Options* dialog will be displayed.



4. Set the required options - you can configure the following:
- Tick the *Control Package* checkbox to indicate this is a controlled package
 - Select the *Version Control* repositories from the *Version Control* dropdown list, this connects this package to a specific version control configuration.
 - Set the *XMI Filename* for import/export
 - Set the *UML/XMI Type* to determine the type of XMI generated, this includes Enterprise Architect XMI/UML 1.3, Rational Rose/Unisys UML 1.3 and Generic XMI 1.0/UML 1.3. Currently only Enterprise Architect UML1.3 is supported for complete import/export round tripping of packages
 - Set the *Version ID* (manual process)
 - Set the package *Owner*
 - Set whether to *Use a DTD*; and
 - Set whether to *Log Import/Export* activity to a log file
 - Whether the package is marked as a *Batch Import* package
 - Whether the package is marked as a *Batch Export* package

Note: When you load a controlled package from file - ALL the current model contents are first deleted, before the file contents are loaded in their place. Please make sure you are certain you wish to revert to a previous version before continuing.

Note: For batch import, the file date of the XMI file is stored and the batch import can be bypassed if the file date of the last import matches that of the current file (ie. there is no change).

8.11.5.3 Disconnect Controlled Package.

It is possible to remove the control from a package. Before removing the package control the package must first be [checked in](#) if it is being used for version control. .

Remove Package Control

1. In the *Project Browser*, right click on the controlled package.
2. From the *Package Control* submenu, select *Configure* or press the *Ctrl + Alt + P* hotkey combination.
3. Uncheck the *Control Package* checkbox.

Control Package:

Version Control: (None) [v]

XMI Filename: CVSPackage.xml [...]

UML/XMI Type: Enterprise Architect XMI/UML 1.3 [v]

Version ID: []

Owner: John [v]

Last Load Date: []

Last Save Date: []

Other Options

Use DTD

Log Import/Export

Batch Import

Batch Export

OK Cancel Help

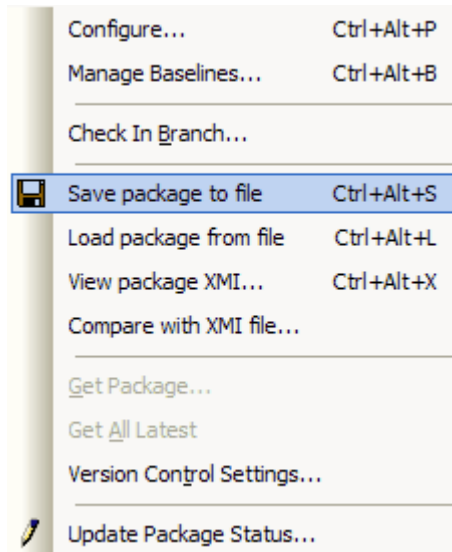
4. Press *OK* to remove package control.
5. The package control for the selected package will now be removed.

8.11.5.4 Save a Package

You can Save a controlled package to an XMI file. Once the package has been correctly [configured](#), follow the steps below:

1. In the Project Browser, right click on the package you wish to save.

2. From the *Package Control* submenu, select *Save Package to File* .



3. The export process will occur automatically according to your configured preferences, overwriting any existing file.

Note: *If you are using a version control package in conjunction with the exported package files, you will need to check out the XMI file first to allow EA to overwrite the existing version.*

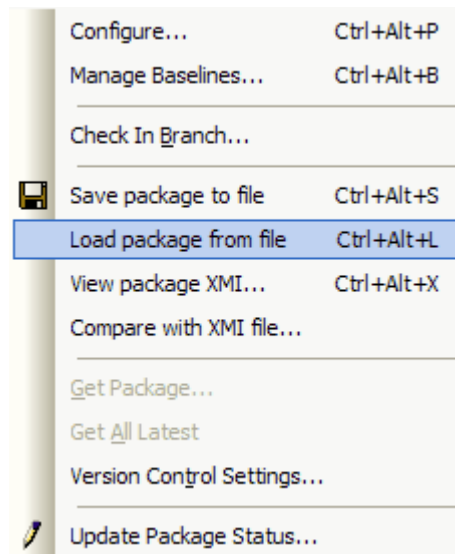
8.11.5.5 Load a Package

Using the Controlled Packages feature you can Save and Load packages to a named file. If a package has been marked for control it will appear with a red rectangle in the left of the package icon in the project view. If you have previously saved a controlled package, you may reload it using the Load Package feature.

Load a Controlled Package

To load a controlled package, follow the steps below:

1. In the Project Browser, right click on the package you wish to load.
2. From the *Package Control* submenu, select *Load Package from File*.



3. If you have configured the package control details you will be asked to confirm the import.

Warning: *Importing deletes the current package entirely from the model, and the action cannot be undone - so care must be taken not to lose any current changes or information.*

4. Press **Yes** to confirm the import. This will delete the current package and proceed with importing the saved package.

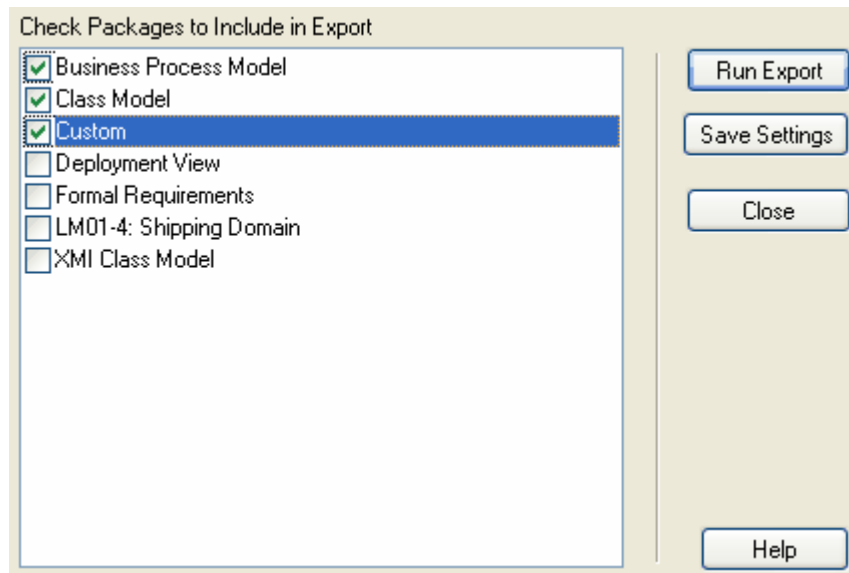
8.11.5.6 Batch XMI Export

A group of controlled packages can be exported in one step, using the Batch XMI Export function.

Batch XMI Export

To export a group of controlled packages, follow the steps below:

1. From the **Project | Import/Export** submenu, select **Batch XMI Export**.



2. In the *Batch Export* dialog, check the packages to include in this export run.
3. Press *Save Settings* if you wish to save this configuration as the default.
4. Press *Run Export*.

EA will cycle through each checked package and export using the options specified in the [Controlled Package](#) dialog. As long as a valid filename exists, EA will export the package to XML and proceed to the next.

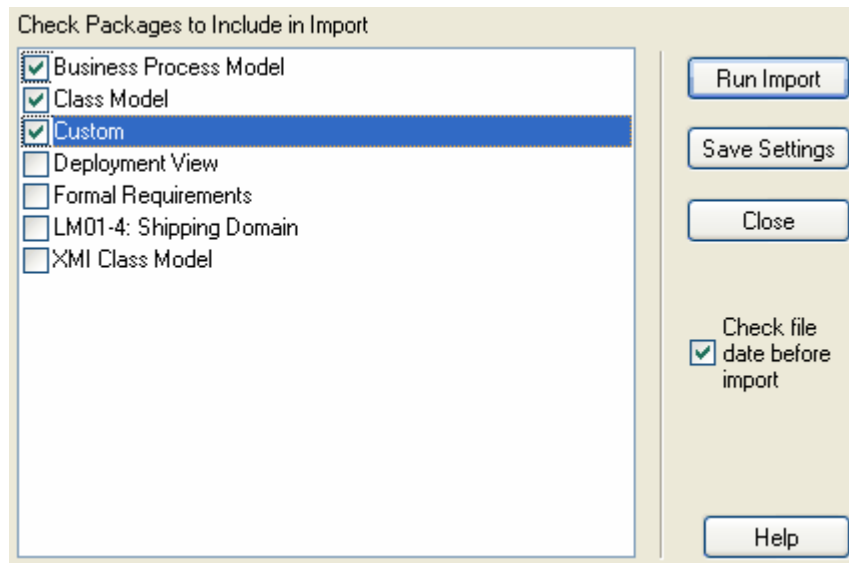
8.11.5.7 Batch XMI Import

A group of controlled packages can be imported in one step, using the Batch XMI Import function.

Batch XMI Export

To export a group of controlled packages, follow the steps below:

1. From the *Project | Import/Export* submenu, select *Batch XMI Import*.



2. In the *Batch Import* dialog, check the packages to include in this import run.

*Tip: If the **Check file date before import** checkbox is ticked, EA will not import a file if the last import file date matches that of the one currently on disk (thus avoiding re-importing the same module multiple times).*

3. Press *Save Settings* if you wish to save this configuration as the default.
4. Press *Run Import*.

8.11.5.8 Manual Version Control with XMI

XMI may be used to support version control by writing model elements in XML text files, suitable for version controlling using standard version control software. Using XMI in this manner allows users to manually connect to 3rd party version control software outside of the EA environment. EA internally supports the configuration of version control via SCC and CVS configurations. To use XMI for version control, the following conditions must be met:

1. Select suitable packages in the Project Browser that will be marked as controlled packages.
2. Configure these with filenames that are visible to a version control system of your choice.
3. Save the controlled packages to establish a model base and check these into the version control system.

When Versioning is Required

Continue working on a package until versioning is required then follow these steps:

1. Check out the package XMI file from the version control system.
2. Save the relevant package using the controlled package support.
3. Check the package back into the version control system

Recover an Earlier Version

To recover an earlier version, follow these steps:

1. Save the current version first if required (important - because the package will be completely deleted during the import process) - and manually update version control system if necessary.
2. Get the required package version from the version control system.
3. Select the package to reload.
4. Use the Package Control/Load Package menu option to import the previous version.
5. EA will delete the controlled package and restore the previous version.

8.12 CSV Import and Export

You can [import](#) and [export](#) information about EA elements in CSV format. You will need to define [CSV specifications](#) to do this.

8.12.1 CSV Specifications

To [import](#) and [export](#) element data from EA using CSV files, you need to first setup one or more file specifications. A file specification lists the fields in the order they will be imported or exported, the filename (optional) and the delimiter between columns. Once you have defined one or more specifications, it can be selected in the CSV import/export dialog as the current specification to apply during an import or export action. CSV will import and only objects and their properties, items such as class attributes cannot be imported or exported through this mechanism. [XML](#) provides a solution to this limitation as does use of the [Automation Interface](#).

To define a specification, open the specification dialog by selecting *Project -> Import/Export -> CSV Import/Export Specifications* from the main menu.

Specification Name: Basic Spec.csv Delimiter: ,

Notes: Basic specification

Default Filename: C:\Temp\BasicSpec.csv

Default Direction: Export

Default Types: class,interface,requirement

Available Fields

Available Element Field

- Type
- Notes
- Priority
- Stereotype
- Language
- Scope

File Specification Up Down Add Field Remove Field

Select Element Field

- Version
- Name
- Phase
- GUID
- Is Leaf
- Author

New Save Save As Delete Cancel Help

The *CSV Import/Export File Specification* dialog provides the following functionality:

| Element | Description |
|--------------------|---|
| Specification Name | Unique name to apply to this specification. Will be used to select a specification from the drop list in the import/export dialog. |
| Delimiter | Character delimiter to use between record fields. Note that if a field contains an instance of the delimiter, the field will be exported wrapped in " (quotation marks) ... and all instances of " in the field will be doubled (ie. " becomes ""). |
| Notes | Description of the specification - only used in this dialog for descriptive purposes. |
| Default Filename | Default filename. |
| Default Direction | Set for Import or for Export. A specification may be used in either direction, but this allows you to set the default type. |
| Default Types | Limit the element types being exported by entering a comma separated list here: eg. class,requirement, component,node,object. |
| Available fields | List of possible record fields, not yet allocated. |

| | |
|--------------------|---|
| File Specification | List of record fields (in order) already assigned. |
| Add Field | Move all selected fields in top list to bottom list. |
| Remove Field | Move all selected fields in bottom list back to available list. |
| New | Create a new specification. |
| Save | Save changes to the currently selected specification. |
| Save As | Save the current specification with a new name. |
| Delete | Delete the current specification. |
| Cancel | Close this dialog. |

8.12.2 CSV Export

It is possible to export information about EA elements in CSV format. Once you have defined a CSV [export specification](#) it is possible to write out major element attributes to a CSV text file.

Export Data in CSV Format

To export data in CSV format, follow these steps:

1. In the Project Browser, right click on the package containing the elements to export.
2. Select the *Import/Export* sub-menu.
3. Select the *CSV Import/Export* menu item.
4. The following dialog will appear.

Package: XMI Class Model

Specification: Basic Spec.csv Edit/New...

File: C:\Temp\BasicSpec.csv ...

Types: class,interface,requirement

Action

Import Export

Include Subfolders

Export Column Names

Progress:

Results

--- Export Complete ---

Print Results View File Run Close Help

5. Set the required options - the *Import/Export Using Delimited File* dialog provides the following functionality:

| Element | Description |
|---------------------|--|
| Package | Name of the current selected package. |
| Specification | Name of export specification to use. |
| Edit/New | Press to edit the export specification or create a new one. |
| File | The filename to export to. |
| Type | List of types to export - leave blank for all, or enter a comma separated list of types. |
| Action | Check Export to export to file, Import to import from file. |
| Include Subfolders | Include all elements in all child packages when exporting. |
| Export column names | If ticked, EA will write out the column names as the first row. |
| Print Results | Print out the result list. |
| View File | View CSV file with default windows application for CSV files. |
| Run | Perform the export. |
| Close | Exit this dialog. |

8.12.3 CSV Import

It is possible to import information about EA elements in CSV format. Once you have defined a CSV [import specification](#) it is possible to read in major element attributes from a CSV text file.

When importing, EA will check the specification to see if there is a GUID field included. If there is, EA will attempt to locate the element identified by the GUID, and if successful will perform an update on the current element, rather than creating a new one. If no GUID field is defined, or EA cannot locate the identified element, a new element is created and placed in the current package. Note that during import, "Type" is a mandatory field and must match one of the legal EA element types.

Import Data in CSV Format

To import data in CSV format, follow these steps:

1. In the Project Browser, right click on the package you wish to import to
2. Select the *Import/Export* sub-menu.
3. Select the *CSV Import/Export* menu item.
4. The following dialog will appear.

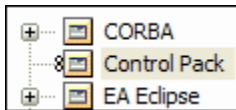
5. Set the required options - the *CSV Import/Export* dialog provides the following functionality:

| Element | Description |
|---------------|---|
| Package | Name of the current selected package. |
| Specification | Name of import specification to use. |
| Edit/New | Press to edit the import specification or create a new one. |
| File | The filename to import from. |
| Types | Not used for import. |
| Action | Check Import to import from file. |
| Print Results | Print out the result list. |
| View File | View CSV file with default windows application for CSV files. |
| Run | Perform the import. |
| Close | Exit this dialog. |

8.13 Version Control

Enterprise Architect supports version control of packages and their component sub-packages to a central repository. This repository is maintained by third-party version control applications that control access and record revisions. Version controlled packages are packages that have been configured for use with version control software. Version Control products that are supported are CVS, SCC, TFS or Subversion.

Controlled packages configured for version control appear in the project browser with a small figure eight "8" to the left of the package icon as in the example below:



See Also

- [Version Control Overview](#)
- [Version Control Setup](#)
- [Using Version Control](#)
- [Version Control Reference](#)
- [Offline Version Control](#)

8.13.1 Version Control Overview

Features

The Version Control feature of Enterprise Architect provides two key facilities:

- Coordinating the sharing of packages between users.
- Saving a history of changes to EA packages, including the ability to retrieve previous versions.

System Requirements

To use version control in EA, a third-party source-code control application is required. EA supports the following version control applications:

- Subversion, which is available from <http://subversion.tigris.org/>
- CVS, which is available from <http://www.wincvs.org/>
- MS Team Foundation Server
- Any version control product that complies with the Microsoft Common Source Code Control standard, version 1.1 or higher.

Set-Up

Before using EA's version control facility, your version control software must be installed on each machine where it is intended to be used.

Typically there will be:

- A server component that manages a version control repository.
- Client components on the workstations that EA uses to communicate with the server.

A version control client must be installed on every machine where you run EA and want to access your version control system. Once the version control software has been installed and configured, you will need to define a Version Control Configuration within EA, to use your installed version control product. For more information on how to do this, see [Version Control Set-up](#).

Note: If you are using the Corporate version of EA with security enabled, you will also need to set up permissions to configure and use version control. See [List of Available Permissions](#) for further information.

Usage

There are four basic ways in which the version control facility might be employed:

| Use | Description |
|---------------------|--|
| Single Shared model | Users share an EA model, stored in a central EAP file or DBMS repository. This configuration allows users to see other users' packages without explicitly having to retrieve them. <ul style="list-style-type: none"> • Version control regulates access to packages, and maintains package revision history. |

| | |
|-------------------------|--|
| Multiple Private models | <p>An EA model is created by a single user who configures it for version control. The model file is then distributed to other users, with each user storing their own private copy of the model.</p> <ul style="list-style-type: none"> • Users update their model's packages through version control. • Version control regulates access to packages, and maintains package revision history. • Other users' new packages are retrieved using the Get Package command. |
| Shared packages | <p>Individual users create separate EA models but share one or more packages.</p> <ul style="list-style-type: none"> • Users share packages through version control. |
| Standard packages | <p>A company may have a standard set of packages which are broadly shared (on a read-only basis).</p> <ul style="list-style-type: none"> • Individual users retrieve packages with the Get Package command. |

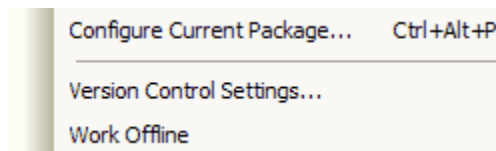
See Also

- [Version Control](#)
- [Version Control Setup](#)
- [Using Version Control](#)
- [Version Control Reference](#)

8.13.2 Version Control Setup

Version control may be assigned to individual packages in EA, each package can only be linked to one version control configuration at a time, although it is possible to connect multiple control configurations for each model. The Version Control Configurations dialog may be used to connect to an SCC provider, CVS configuration, MS Team Foundation Server or to a Subversion configuration.

The Version Control Setup Menu is available under *Project | Version Control* from the main menu. To set the version control configurations choose the *Version Control Options...* item from the menu.

**See Also**

- [Version Control Options SCC](#)
- [CVS with Remote Repositories](#)
- [CVS with Local Repositories](#)
- [Version Control with Subversion](#)
- [Version Control with TFS](#)
- [Configure Package for Version Control](#)

8.13.3 Using Version Control

Links to the most common activities using the version control features of Enterprise Architect as listed below.

General Notes

- The export/import facility is somewhat slow and submitting packages containing many sub-nodes to version control should be avoided. It is recommended version control is applied on an individual package basis. See [Using Nested Version Control Packages](#) for more information.
- Replication should not be combined with version controlled packages.

See Also

- [Version Control Options SCC](#)
- [Version Control Options CVS](#)
- [Version Control with Subversion](#)
- [Version Control with TFS](#)
- [Configuring Package for Version Control](#)
- [Checking In and Checking out Packages](#)
- [Offline Version Control](#)
- [Review Package History](#)
- [Remove Package from Version Control](#)
- [Add a Previously defined Version Control Configuration to a package](#)
- [Include Other Users Packages](#)
- [SCC Version Control Upgrade for 4.5](#)
- [Specifying Private or Shared Models](#)
- [Using Nested Version Control Packages](#)

8.13.4 Version Control Reference

The following references are available for Version Control.

Menus

- [Version Control Setup Menu](#)
- [Version Control Menu](#)

Dialogs

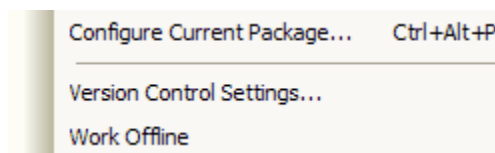
- [Version Control Providers Dialog](#)
- [Version Control Options Dialog](#)

See Also

- [Version Control Options SCC](#)
- [Version Control with CVS](#)
- [Version Control with Subversion](#)
- [Version Control with TFS](#)

8.13.4.1 Version Control Setup Menu

The Version Control Setup Menu is available under *Project* | *Version Control* in the main menu.



| Menu Item | Functionality |
|-----------------------------|--|
| Package Control | Opens the package control dialog which allows you to specify whether this package (and its children) is controlled, and which file it is controlled through. |
| Set Version Control Options | Displays the Version Control Options dialog . |

| | |
|--------------------------------|--|
| Start Version Control Explorer | Asks the SCC provider to display an interface allowing you to review all SCC projects and their contents directly. |
|--------------------------------|--|

8.13.4.2 Version Control Options Dialog

The Version Control Configurations dialog allows you to specify the information required to create a Version Control Configuration, which can then be used to establish a connection to a Version Control provider. EA supports version control through MS Team Foundation Server, Subversion, CVS or any SCC compliant version control product.

It is possible to use multiple version control configurations in the same EA model. It is also possible to use the same version control configuration across different models, to facilitate sharing of "standard" packages between those models, through the version control system.

Setting Up Version Control

When the Version Control Configurations dialog is opened for the first time in any given model, it will appear as shown below.

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configuration Details:

Unique ID:

Type: SCC CVS Subversion TFS

Defined Configurations:

| Unique ID | Type | Files | Location |
|-----------|------|-------|----------|
|-----------|------|-------|----------|

To begin defining a new version control configuration;

1. Click the **New** button.
2. Enter a suitable name in the **Unique ID** field.
3. Click on the **Type** radio button corresponding to the type of version control product you want to connect to.

At this point, the middle section of the dialog will change, to display a collection of fields relating to the type of version control configuration being defined.

For details on configuring a particular type of version control, choose the relevant page from the links below.

To "import" a previously defined configuration for use in the current model;

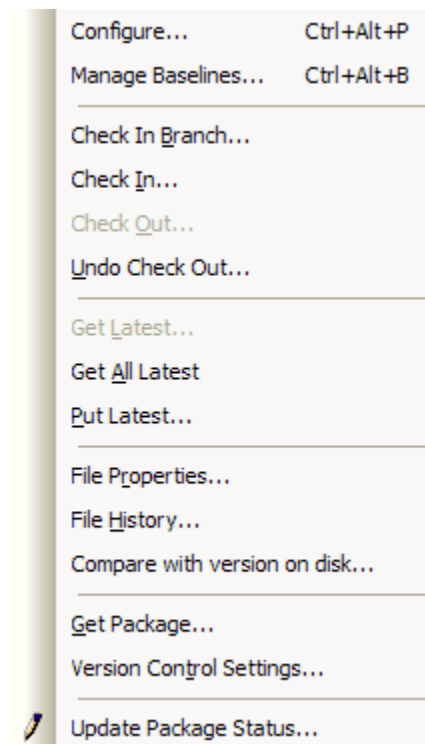
1. Click the *New* button.
2. From the *Unique ID* drop down list, select one of the previously defined version control configurations.
3. Press the *Save* button to save the selected version control configuration in this model.

See Also

- [Version Control Options SCC](#)
- [Version Control Options CVS](#)
- [Version Control Subversion](#)
- [Version Control with TFS](#)
- [Configure Package for Version Control](#)

8.13.4.3 Version Control Menu

The *Version Control* menu is found by right clicking on a version controlled package in the *Project Browser* and selecting *Package Control*.



| Menu Item | Functionality |
|-----------------|--|
| Configure | Shows the package control dialog which allows you to specify whether this package (and its children) is controlled, and which file it is controlled through. |
| Check In Branch | For the selected branch of the model, (i.e. the selected package and all of its child packages) displays a dialog, listing all version controlled packages within that branch that are checked out to the current user. The user may then select packages in the displayed list, to be submitted for check-in. |

| | |
|-------------------------|---|
| Check In | Submits the currently selected package and all sub-packages to the central repository. You will be prompted to enter optional comments describing changes to the packages. |
| Check Out | Retrieves the latest version of the currently selected package and sub-packages from the central repository, overwriting the current packages. After check out the packages are available for editing. |
| Undo Check Out | Cancels all changes you have made to the currently selected package and sub-packages. Restores the model to the state it was in before package was checked out, leaving the select package and sub-packages locked. |
| Get Latest | Retrieves the latest revision of the package from the repository. Available only for packages that are checked in. Available only on Private Models. |
| Get All Latest | Retrieves the latest revision of the all version controlled packages in the project. Only retrieves packages that are checked in. Available only on Private Models. |
| Put Latest | Updates the central repository with the currently selected package (which you have checked out), while retaining checkout status on the package. This is equivalent to checking a package in and immediately checking it back out again. |
| File Properties | Asks the version control provider to show the version control properties associated with the XML export file pertaining to the currently selected package. |
| File History | Where the controlling package has been configured by an SCC provider, this provider will show a change history for the package. Refer to your provider's documentation for details on how to use the control. Otherwise if the version control is CVS the history will be shown via EA's internal CVS history menu. |
| Get Package | Allows the user to gain access from packages in the version control repository that is not currently available in the users model. |
| Version Control Options | Displays the Version Control Options dialog . |
| Update Package Status | Allows the user to provide a bulk update on the status of a package, this includes status options such as Proposed, Validate, Mandatory etc. Note: this option is a generic package element not specific to version control. |
| Set as Namespace Root | Sets the namespace root for languages which support namespaces, for more information see the Namespaces section. Note: this option is a generic package element not specific to version control. |

8.13.4.4 Version Control Options SCC

To set up SCC version control configuration use the following steps:

Set Up the Source Code Control Provider with SCC

To setup the third-party source code control provider, refer to the documentation provided with that application. A repository must be set up using the SCC provider and access to that repository must be available to all intended users.

Connect an EA Model to Version Control with SCC

1. Open/Create the EA model you wish to place under version control.
2. From the *Project* | *Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button, enter a suitable name in the *Unique ID* field, then click on the *Type* radio

button to select **SCC**.

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configuration Details:

Unique ID:

Type: SCC CVS Subversion TFS

Local Project path:

Current User:

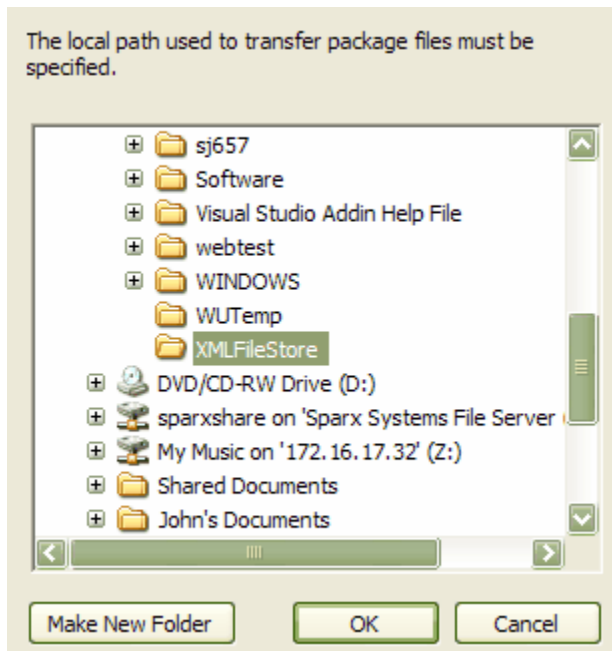
SCC Provider:

SCC Project:

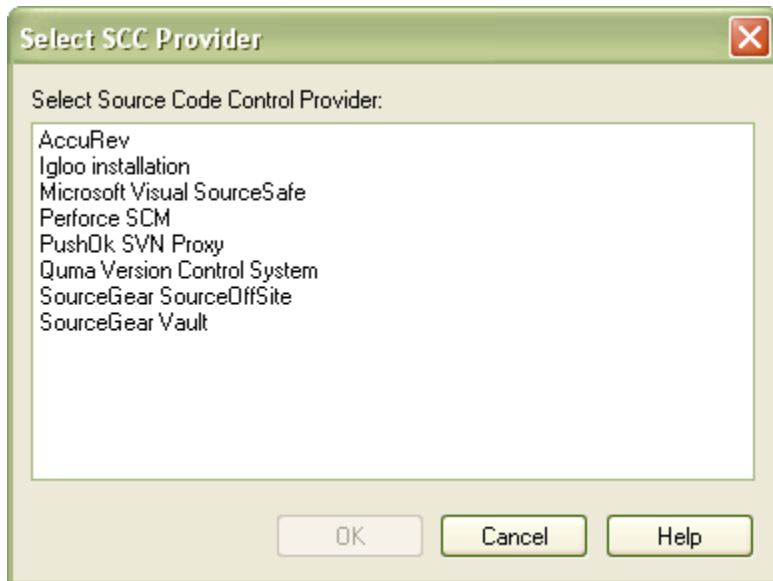
Defined Configurations:

| Unique ID | Type | Files | Location |
|---------------|------|-------|-----------------|
| TFS_Test2 | TFS | 0 | %TFS_Test2% |
| AccuRev4-test | SCC | 0 | %AccuRev4-test% |
| CVS-tester | CVS | 0 | %CVS-tester% |
| SVN_Test | SVN | 0 | %SVN_Test% |

4. The **Local Project path** field is specified by clicking on the **Select Path...** button. Select the local folder which will be used to keep local working copies of the XML files to be stored in the Version Control repository.



5. Select an SCC provider from the dialog which opens, and press the **OK** button.



6. Click the **Save** button to save the configuration you have defined.
 7. The SCC provider is likely to prompt you for various details including the name of the project you want to connect to, and perhaps the user name you want to use when you log in.
 8. The new configuration is added to the list of **Defined Configurations**.
- NOTE: A new entry is also created in the **Local Paths** list, with the same ID as the new version control configuration. The Local Path entry records the Local Project path, for use in subsequent path substitutions.
9. Press the **Close** button when you have finished defining your version control configurations.

| Dialog Item | Functionality |
|---|--|
| This model is private | This setting controls whether the Get Latest and Get All Latest commands are enabled. For an explanation of why this is so, see Specifying Private or Shared Models . |
| Save nested version controlled packages to stubs only | For a full explanation of this option, see Using Nested Version Control Packages . |
| Unique ID | Specify a configuration name that is will readily distinguish it from other configurations. The Unique ID will appear as a selection in list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in the current model. |
| Local Project Path | The folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every PC using version control should have its own local SCC project folder - this should not be a shared network folder. Particularly bear this in mind if you are creating an EAP file which is to be shared (eg. A SQL database). |
| Current User | Read only. Shows the user name of the user currently logged into the SCC provider. |
| SCC Provider | Read only. This is the name of the provider specified in the database. |
| SCC Project | Read only. This is the project selected during the initial setup of the connection to the SCC provider. |

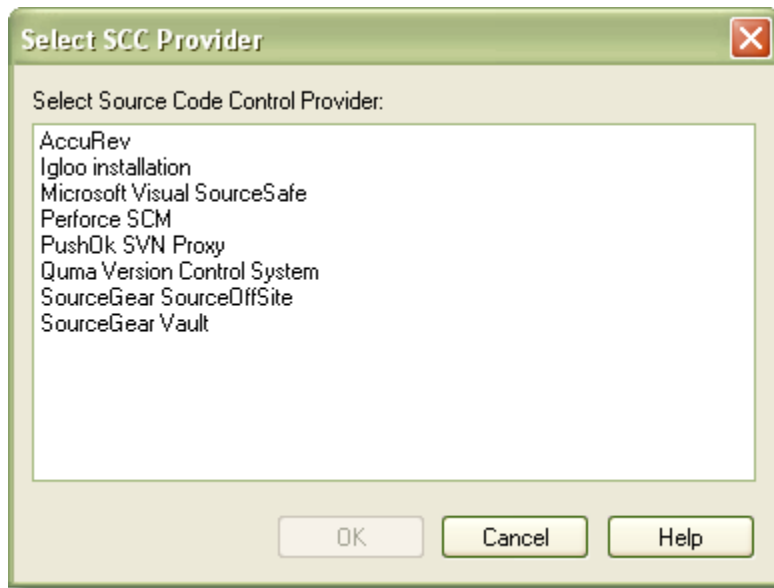
See Also

- [SCC Providers Dialog](#)
- [SCC Version Control User Options](#)

8.13.4.4.1 SCC Providers Dialog

When using a SCC version control, the *Select SCC Provider* dialog allows the selection from a list of the installed SCC configuration providers that may be used for this package.

Note: All users of the shared database are required to specify the same SCC provider.

**See Also**

- [Version Control Options SCC](#)
- [SCC Version Control User Options](#)

8.13.4.4.2 SCC Version Control User Options

Change User

It may be necessary to change the SCC user name. While the following procedure allows this to happen, it is not generally recommended as it may result in a confused checkout states for packages.

1. From the *Project | Version Control* submenu, select *Set Version Control Options*.
2. Select *Change User*.

Users of SCC

Some SCC providers require you to specify a user name, in which the following points apply:

- The version control user name has no relationship to the users set up as part of EA security.
- A single user can be logged into multiple machines simultaneously.
- It is not a "roaming" facility - logging into a PC does not grant access rights to packages checked out on other machines. Each machine has its own record of which packages are checked out and will only allow access to those.
- User name is stored per PC and is stored across EA sessions. If you want to change to a different user press *Change User* in the version control screen.

See Also

- [Version Control Options SCC](#)
- [SCC Providers Dialog](#)

8.13.4.5 Version Control with CVS

CVS is used to manage files and directories and is an open source, version control system. In order to use CVS version control with EA, you must install version control software on your local machine. Also you need to create a working directory before you can configure EA. Use your version control software to create a working

directory. You can have as many working directories as you like on your local machine.

You will also need to connect to a repository, this repository can either be a remote repository or local, located to your machine. If your repository is local, it will need to be created with your version control software.

Each working folder you create will contain connection information to a repository. This connection information includes the path to the repository, either local or remote, the user name and password in order to make a connection.

See Also

- [CVS with Remote Repositories](#)
- [CVS with Local Repositories](#)

8.13.4.5.1 CVS with Remote Repositories

Before you can connect to a remote repository, you must...

- Have version control setup on your local machine.
- Have version control setup on a remote server.
- Have a *working directory* on your local machine that points to the repository on the server.

To set up CVS version control with a remote repository use the following steps:

1. Firstly, have your system administrator install CVS and create a remote repository with a module that you can use to control your EA package files. Your administrator will have to create a username and password for you before you can make a connection.
2. Open a command prompt window and navigate to, or create, a suitable directory, which will hold your CVS working copy directory, e.g.:

```
C:\> cd myCVSWorkSpace
```

3. Connect to the remote CVS repository. An example connection command is shown below.

```
C:\myCVSWorkSpace> cvs -d :pserver:myUserID@ServerName:/repositoryFolder login
```

Note: replace myUserID with your CVS username, replace ServerName with the name of your CVS server and replace repositoryFolder with the path to the repository on the server. You will be prompted to enter your password.

4. Create a local CVS workspace, derived from the remote repository. An example command is shown below.

```
c:\myCVSWorkSpace> cvs -d :pserver:myUserID@ServerName:/cvs checkout moduleName
```

Note:

- The above command will create a subdirectory in your current working directory, called **moduleName**. (Replace **moduleName** with the name of the module created by your system administrator).
- It creates local copies of all files contained in the CVS module found at ServerName:/cvs
- It also creates a subdirectory beneath **moduleName**, called CVS. This subdirectory contains a file called Root, that contains your CVS connection information. EA uses this file to obtain your CVS userID.

5. Verify that your CVS installation is working correctly.

Change directory to the one you specified as the working copy, in the "cvs checkout" command above, i.e. C:\myCVSWorkSpace\moduleName

Now create a test file, e.g. Test.txt, containing the text "CVS Test". You can do this with the command

```
echo CVS Test > Test.txt
```

Execute the following CVS commands:

```
cvs add Test.txt
cvs commit -m"Commit comment" Test.txt
cvs update Test.txt
cvs edit Test.txt
cvs editors Test.txt
```

The "editors" command should produce output like the following:

```
Test1.txt myUserID Tue Aug 9 10:08:43 2009 GMT myComputer C:\myCVSWorkspace\moduleName
```

Take note of the userID that follows the filename. EA must find and use this userID when you create your Version Control Configuration. (see example dialog below).

6. Launch EA and open or create the model whose packages you wish to place under version control.
7. From the *Project | Version Control* submenu, select *Set Version Control Options*.
8. Press the *New* button, enter a suitable name in the *Unique ID* field, then click on the *Type* radio button to select *CVS*.

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configuration Details:

Unique ID:

Type: SCC CVS Subversion TFS

Working Copy path:

Current User:

Workstation Settings:

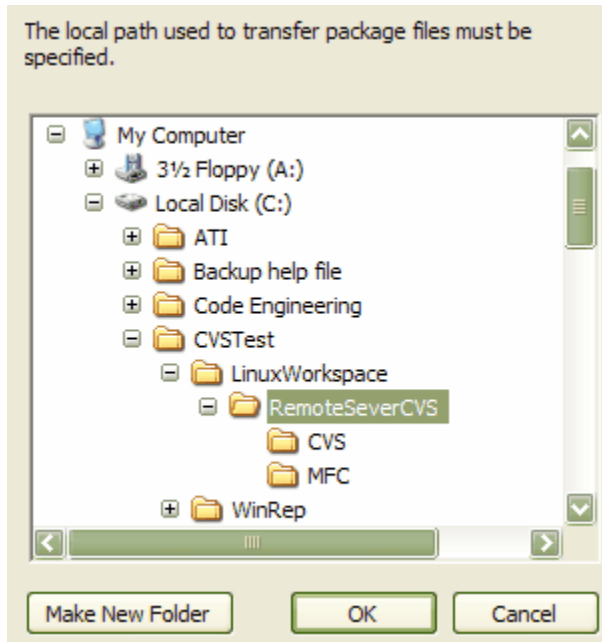
CVS Exe Path:

Defined Configurations:

| Unique ID | Type | Files | Location |
|---------------|------|-------|-----------------|
| TFS_Test2 | TFS | 0 | %TFS_Test2% |
| AccuRev4-test | SCC | 0 | %AccuRev4-test% |
| CVS-tester | CVS | 0 | %CVS-tester% |
| SVN_Test | SVN | 0 | %SVN_Test% |

9. The *Working Copy path* field is specified by clicking on the *Select Path...* button. Select the Local Folder that will be used to keep local working copies of the XML files to be stored in the Version Control repository. Having specified the Working Copy path, the name in the *Current User* field should reflect the user name that was used to log into the remote CVS repository. If this does not happen, it indicates that EA cannot extract the user name from the file "...WorkingCopyPath\CVS\Root" and the configuration will not work correctly.

10. If necessary, set the *Client Exe Path*, by clicking *Select Path...* and browsing to the file path for the file 'cvs.exe', the CVS executable.



11. Click the **Save** button to save the configuration you have defined.
12. The new configuration is added to the list of **Defined Configurations**.

Note: A new entry is also created in the Local Paths list, with the same ID as the new version control configuration. The Local Path entry records the Local Project path, for use in subsequent path substitutions.

| Dialog Item | Functionality |
|---|---|
| This model is Private | This setting controls whether the Get Latest and Get All Latest commands are enabled. For an explanation of why this is so, see the Specifying Private or Shared Models topic. |
| Save nested version controlled packages to stubs only | For a full explanation of this option, see Using Nested Version Control Packages . |
| Unique ID | Specify a configuration name that will readily distinguish it from other configurations. The Unique ID will appear as a selection in a list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in use in the current model. |
| Working Copy Path | The folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every version control configuration you define in EA, should have its own local Working Copy Folder in which to store working copies of the XMI package files - this should not be a shared network folder. Particularly bear this in mind if you are creating an EAP file which is to be shared (eg. A SQL database). |
| Current User | The CVS user name that will be associated with all CVS commands that are issued. This name is used by EA, to determine who has a package "checked-out". |
| CVS EXE Path | The full path of the CVS client's executable file. |

Note: We strongly urge you not to manipulate version controlled package files outside of EA. It is possible to leave the package files in a state that EA cannot recognize.

See Also

- [Version Control with CVS](#)
- [CVS with Local Repositories](#)

8.13.4.5.2 CVS with Local Repositories

Before you can setup EA, you must have a *working directory* that points to a local repository, i.e. one that is installed on your local machine. See your version control software help files for more information.

To set up CVS version control use the following steps:

1. Launch EA and open/create the EA model whose packages you wish to place under version control.
2. From the *Project* | *Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button, enter a suitable name in the *Unique ID* field, then click on the *Type* radio button to select *CVS*.

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configuration Details:

Unique ID:

Type: SCC CVS Subversion TFS

Working Copy path:

Current User:

Workstation Settings:

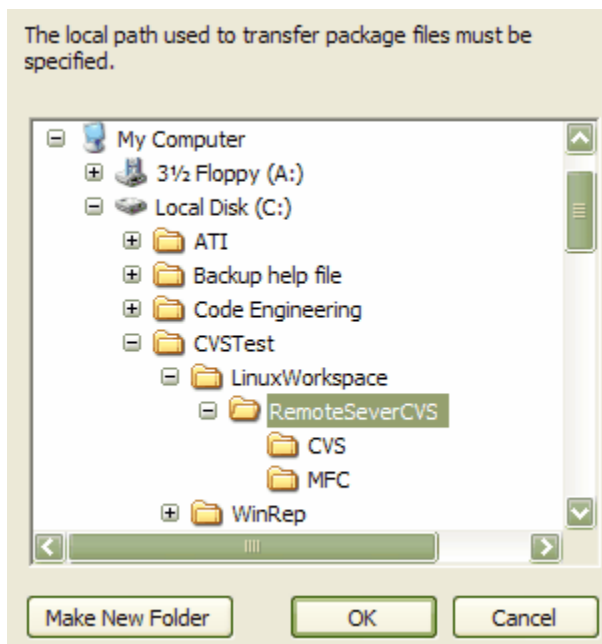
CVS Exe Path:

Defined Configurations:

| Unique ID | Type | Files | Location |
|---------------|------|-------|-----------------|
| TFS_Test2 | TFS | 0 | %TFS_Test2% |
| AccuRev4-test | SCC | 0 | %AccuRev4-test% |
| CVS-tester | CVS | 0 | %CVS-tester% |
| SVN_Test | SVN | 0 | %SVN_Test% |

4. The *Working Copy path* field is specified by clicking on the *Select Path...* button. Select the Local Folder that will be used to keep local working copies of the XML files to be stored in the Version Control repository.

5. If necessary, set the *Client Exe Path*, by clicking *Select Path...* and browsing to the file path for the file 'cvs.exe', the CVS executable.



6. Click the **Save** button to save the configuration you have defined.
7. The new configuration is added to the list of **Defined Configurations**.

Note: A new entry is also created in the [Local Paths](#) list, with the same ID as the new version control configuration. The Local Path entry records the Local Project path, for use in subsequent path substitutions

| Dialog Item | Functionality |
|---|---|
| This model is Private | This setting controls whether the Get Latest and Get All Latest commands are enabled. For an explanation of why this is so, see the Specifying Private or Shared Models topic. |
| Save nested version controlled packages to stubs only | For a full explanation of this option, see Using Nested Version Control Packages . |
| Unique ID | Specify a configuration name that will readily distinguish it from other configurations. The Unique ID will appear as a selection in a list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in use in the current model. |
| Working Copy Plan | The folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every version control configuration you define in EA, should have its own local Working Copy Folder in which to store working copies of the XMI package files - this should not be a shared network folder. Particularly bear this in mind if you are creating an EAP file which is to be shared (eg. A SQL database). |
| Current User | The CVS user name that will be associated with all CVS commands that are issued. This name is used by EA, to determine who has a package "checked-out". |
| CVS EXE Path | The full path name of the CVS client's executable file. |

Note: We strongly urge you not to manipulate version controlled package files outside of EA. It is possible to leave the package files in a state that EA cannot recognize.

See Also

- [Version Control with CVS](#)
- [CVS with Remote Repositories](#)

8.13.4.6 Version Control with Subversion

Subversion is used to manage files and directories and is an open source version control system. To make use of Subversion control you must have version 6.0 or greater of EA.

See Also

- [Setting up Subversion](#)
- [Creating a new Repository Sub-Tree](#)
- [Create a Local Working Copy](#)
- [Configure Version Control with Subversion](#)
- [TortoiseSVN](#)

8.13.4.6.1 Setting up Subversion

Obtaining and Installing Subversion

Note: Enterprise Architect relies on exclusive file locking when applying version control to its packages. File locking was not introduced into Subversion until version 1.2. EA will not work with Subversion, unless you are using Subversion 1.2 or later.

Before EA can be used with Subversion, the appropriate software needs to be installed by a subversion administrator, have your system administrator obtain and install the Subversion server and client applications.

Official Subversion documentation can be found at: <http://svnbook.red-bean.com/en/1.1/index.html>, while executable files for Subversion can be obtained from: http://subversion.tigris.org/project_packages.html#binary-packages.

You will require the Windows executables for your client machines running Enterprise Architect in the windows environment, if you plan to run your Subversion server on a non-windows platform, you will need to download a binary suitable for that platform as well.

Chapter 6 in the Subversion documentation provides guidance on how to configure the server for different methods of access by the client. Secure connection methods are also covered in this section.

Your administrator should setup userIDs and passwords for everybody that will access the repository. Your administrator should then provide all users with the "path to the repository", and ensure that they can all connect.

Before users can make use of Subversion, they need to create local working copies from the repository, by checking-out a repository sub-tree.

Instructions for setting up a repository and creating a local working copy can be found at: <http://svnbook.red-bean.com/en/1.1/ch01s07.html>.

Note: We recommend that each new EA model being added to version control with Subversion, should have a separate repository sub-tree created for it, and users should create a new local working copy from the sub-tree, to be used with that model.

Repository URLs

Subversion repositories can be accessed through many different methods, on local disk, or through various network protocols. A repository location, however, is always a URL. The table below describes how different URL schemas map to the available access methods.

| Schema | Access Method |
|----------|---|
| file:// | Direct repository access (on local disk) |
| http:// | Access via WebDAV protocol to a Subversion-aware Apache server. |
| https:// | Same as http://, but with SSL encryption. |

| | |
|------------|--|
| svn:// | Access via custom protocol to an svnserve server |
| svn+ssh:// | Same as svn://, but through an SSH tunnel. |

For more information on how Subversion parses URLs, see <http://svnbook.red-bean.com/en/1.1/ch07s07.html>.

See Also

- [Version Control with Subversion](#)
- [Creating a new Repository Sub-Tree](#)
- [Create a Local Working Copy](#)
- [Configure Version Control with Subversion](#)
- [TortoiseSVN](#)

8.13.4.6.2 Creating a new Repository Sub-tree

If a repository sub-tree has already been created for your EA model, skip this section and proceed to [Create a Local Working Copy](#). If your EA model has not previously been added to version control, perform the following steps to create a sub-tree for it in your SVN repository.

1. Create a temporary directory structure to import into the SVN repository, which will initialise the repository sub-tree for this EA model. The directory structure should look like this;

```
tempDir
|
+--<EA_Model_Name>
|
+--trunk
|
+--branches
|
+--tags
```

2. Open a command prompt, navigate to tempDir and issue the command

```
svn import . <repositoryURL> --message "A Comment of your choice"
```

3. Note that after the import is finished, the original tree is not converted into a working copy. To start working, you still need to svn checkout a fresh working copy of the tree.
4. You can now delete the directory tempDir and all its contents.
5. For further information see <http://svnbook.red-bean.com/en/1.1/svn-book.html#svn-ch-5-sect-6>

See Also

- [Version Control with Subversion](#)
- [Setting up Subversion](#)
- [Create a Local Working Copy](#)
- [Configure Version Control with Subversion](#)
- [TortoiseSVN](#)

8.13.4.6.3 Create a Local Working Copy

Once you have created a sub-tree in the repository for this model, or if one already exists, you are ready to create the local Working Copy for use with this model.

1. Choose a suitable directory on your system, in which you wish to create your Subversion Working Copy. The directory that contains your model's .EAP file is probably a good choice.

2. Open a command line window, navigate to the directory that will hold your Working Copy directory and check-out the model's sub-tree from the repository, with the following command;
3. `svn checkout <repositoryURL>/<EA_Model_Name>`, where `<EA_Model_Name>` is the directory name that you used in setting up the repository sub-tree above.

After you have created your working copy, we recommend that you verify everything is working correctly before you attempt to use it from within EA. You must be able to commit files to the repository, without being prompted for ID or passwords.

EA interacts with Subversion using its command line client. Firstly, create a file in your working copy folder, then, from a command prompt, add and commit the file to the repository.

Use the following commands:

```
svn add <fileName>
svn lock <fileName>
```

Now, update the file from the repository, lock the file, edit it and commit once more.

Use the following commands:

```
svn update <fileName>
svn lock <fileName>
```

Then edit and save the file using your favourite editor

```
svn commit <fileName> -m"A meaningful comment."
```

See Also

- [Version Control with Subversion](#)
- [Setting up Subversion](#)
- [Create a Local Working Copy](#)
- [Configure Version Control with Subversion](#)
- [TortoiseSVN](#)

8.13.4.6.4 Configure Version Control with Subversion

This section assumes that you have already installed Subversion (both the server and the client parts), and that you have a local working copy, derived from a repository sub-tree, already set up for use with your EA model. If this is not the case, please refer to the [Setting up Subversion](#) topic.

Once you have set up and tested the Local Working Copy, you are ready to define a Version Control configuration for use with the EA model that you wish to place under version control.

To apply version control to your EA model, using the Subversion working copy that you have set up, perform the following steps;

1. Launch EA and open the model for which this Working Copy was created.
2. From the *Project | Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button, enter a suitable name in the *Unique ID* field, then click on the *Type* radio button to select *Subversion*.

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configuration Details:

Unique ID:

Type: SCC CVS Subversion TFS

Working Copy path:

Workstation Settings:

Subversion Exe Path:

Defined Configurations:

| Unique ID | Type | Files | Location |
|---------------|------|-------|-----------------|
| TFS_Test2 | TFS | 0 | %TFS_Test2% |
| AccuRev4-test | SCC | 0 | %AccuRev4-test% |
| CVS-tester | CVS | 0 | %CVS-tester% |
| SVN_Test | SVN | 0 | %SVN_Test% |

4. The *Working Copy path* field is specified by clicking on the *Select Path...* button. Select the Local Folder that will be used to keep local working copies of the XML files to be stored in the Version Control repository.
5. Now, in each of the named fields, enter the appropriate Server Name, Workspace Name, User Name and Password, corresponding to the Local Folder that was specified.
6. Click on *Select Path...* to specify the path for your Subversion client executable
7. Click the *Save* button to save the configuration you have defined.
8. The new configuration is added to the list of *Defined Configurations*.

Note: A new entry is also created in the [Local Paths](#) list, with the same ID as the new version control

configuration. The Local Path entry records the Local Project path, for use in subsequent path substitutions.

9. Press the *Close* button when you have finished defining your version control configurations.

| Dialog Item | Functionality |
|---|--|
| This model is private | This setting controls whether the Get Latest and Get All Latest commands are enabled. For an explanation of why this is so, see the Specifying Private or Shared Models topic. |
| Save nested version controlled packages to stubs only | For a detailed explanation of this option, see Using Nested Version Control Packages . |
| Unique ID | Specify a configuration name that is will readily distinguish it from other configurations. The Unique ID will appear as a selection in list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in the current model. |
| Working Copy path | The folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every PC using TFS version control should have its own TFS Local Folder in which to store working copies of the XMI package files - this should not be a shared network folder. Particularly bear this in mind if you are creating an EAP file which is to be shared (eg. A SQL database). |
| Subversion Exe Path | The full path name of the Subversion client executable file. |

Note: We strongly urge you not to manipulate version controlled package files outside of EA. It is possible to leave the package files in a state that EA cannot recognize.

See Also

- [Version Control with Subversion](#)
- [Setting up Subversion](#)
- [Creating a new Repository Sub-Tree](#)
- [Create a Local Working Copy](#)
- [TortoiseSVN](#)

8.13.4.6.5 TortoiseSVN

TortoiseSVN is a Windows shell extension for Subversion that user's may find worthwhile. The icon overlays that it provides in Windows Explorer are quite useful as a tool for observing the status of your Subversion controlled files. It is also provides the ability to create your repository sub-trees and check out local working copies from within the Windows Explorer using simple menu commands.

Note: We still recommend that users "test" their local working copies, by adding and committing a dummy file from the command prompt window.

Note: Manipulating EA's package files, using tools that are external to EA, may leave those files in a state that EA cannot use.

TortoiseSVN can be downloaded from: <http://tortoisesvn.tigris.org/>.

See Also

- [Version Control with Subversion](#)
- [Setting up Subversion](#)
- [Creating a new Repository Sub-Tree](#)
- [Create a Local Working Copy](#)
- [Configure Version Control with Subversion](#)

8.13.4.7 Version Control with TFS

In order to use Team Foundation Server for version control with EA, all users must have a TFS client installed on their local machine and each intended user must have an account that provides read/write access to a workspace on the server.

Each user will need to set up a local working folder on their own machine, that is mapped, through the workspace, to a Source Control folder on the server.

These preliminary steps should be performed on each PC and for each user, before making any attempt to define a Version Control Configuration within EA, that will use TFS.

Connect an EA Model to Version Control using TFS.

1. Open/Create the EA model you wish to place under version control.
2. From the *Project | Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button, enter a suitable name in the *Unique ID* field, then click on the *Type* radio button to select *TFS*.

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configuration Details:

Unique ID: TFS-config

Type: SCC CVS Subversion TFS

Working Copy path: C:\WC_WorkSpaces\TFS\WorkFolder1 Select Path...

Server Name:

Workspace Name:

User Name: SPARXSYSTEMS\userOne

Password: ●●●●●●●●

Workstation Settings:

TFS Exe Path: Files\Microsoft Visual Studio 8\Common7\IDE\tf.exe Select Path...

New Save Delete

Defined Configurations:

| Unique ID | Type | Files | Location |
|-----------|------|-------|----------|
| | | | |

Close Help

4. The *Working Copy path* field is specified by clicking on the *Select Path...* button. Select the Local Folder that will be used to keep local working copies of the XML files to be stored in the Version Control repository.

Note: EA will query TFS to retrieve the Server and Workspace names associated with this folder, when attempting to save the configuration data.

5. In the fields *User Name* and *Password*, enter values that will allow access to the TFS workspace associated with the *Working Copy path* specified above.

6. The *TFS Exe Path* should be set to the default installation path. Click on *Select Path...* if you need to modify this field.

7. Click the *Save* button to save the configuration you have defined.

8. The new configuration is added to the list of *Defined Configurations*.

Note: A new entry is also created in the [Local Paths](#) list, with the same ID as the new version control configuration. The Local Path entry records the Local Project path, for use in subsequent path substitutions.

9. Press the *Close* button when you have finished defining your version control configurations.

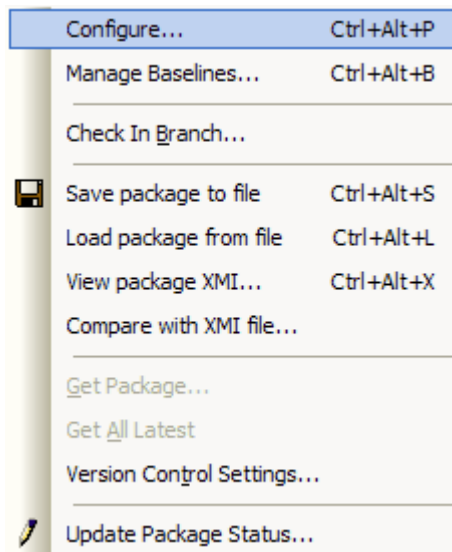
| Dialog Item | Functionality |
|---|---|
| This model is private | This setting controls whether the Get Latest and Get All Latest commands are enabled. For an explanation of why this is so, see Specifying Private or Shared Models . |
| Save nested version controlled packages to stubs only | For a full explanation of this option, see Using Nested Version Control Packages . |
| Unique ID | Specify a configuration name that is will readily distinguish it from other configurations. The Unique ID will appear as a selection in list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in the current model. |
| Working Copy Path | The folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every PC using TFS version control should have its own TFS Local Folder in which to store working copies of the XMI package files - this should not be a shared network folder. Particularly bear this in mind if you are creating an EAP file which is to be shared (eg. A SQL database). |
| Server Name | The name of the Team Foundation Server that you wish to connect to. |
| Workspace Name | The name of a pre-defined TFS workspace that you will be using |
| User Name | The user name that you use to connect to the Team Foundation Server. The user name that you specify should give you read/write permissions in the specified workspace. |
| Password | The password associated with the user name you specify. EA stores this password, in encrypted form, as part of the VC configuration data. |
| TFS Exe Path | The full path name of the TFS client's executable file. |

Note: We strongly urge you not to manipulate version controlled package files outside of EA. It is possible to leave the package files in a state that EA cannot recognize.

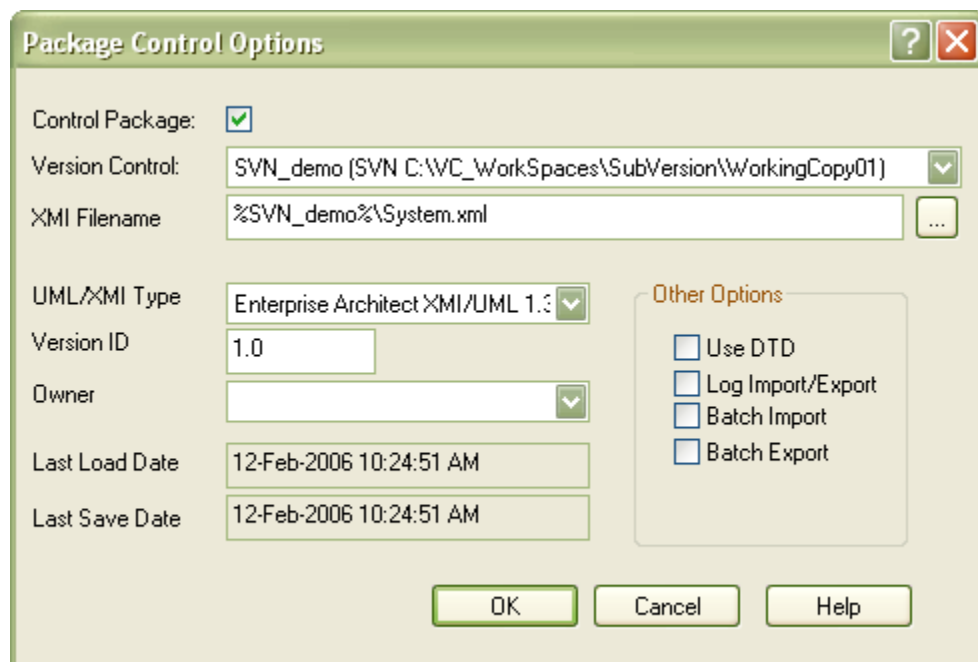
8.13.4.8 Configure Package for Version Control

To configure a version controlled package, follow the steps detailed below:

1. In the Project Browser, right click on the package you wish to control or configure.
2. From the *Package Control* submenu, select *Configure*.



3. The *Package Control Options* dialog will be displayed.



4. Ensure that the *Control Package* checkbox is selected.
5. Select a version control configuration from the *Version Control* drop down menu.
6. Specify an XMI filename or press the ... button to browse to an xml file (EA creates a default filename using the package name, when the ... button is selected).
7. Fill in any of the remaining options on the dialog - you can configure the following:
 - Set the *UML/XMI Type* to determine the type of XMI generated, this includes Enterprise Architect XMI/UML 1.3, Rational Rose/Unisys UML 1.3 and Generic XMI 1.0/UML 1.3. Currently only Enterprise Architect UML1.3 is supported for complete import/export round tripping of packages

- Set the *Version ID* (manual process)
- Set the package *Owner*
- Set whether to *Use a DTD*; and
- Set whether to *Log Import/Export* activity to a log file
- Whether the package is marked as a *Batch Import* package
- Whether the package is marked as a *Batch Export* package

8.13.4.9 Checking In and Checking out Packages.

To work on a version controlled package the user must have the package checked out. When a package is checked out to a specific user other users may not make changes to the package until it has been checked in.

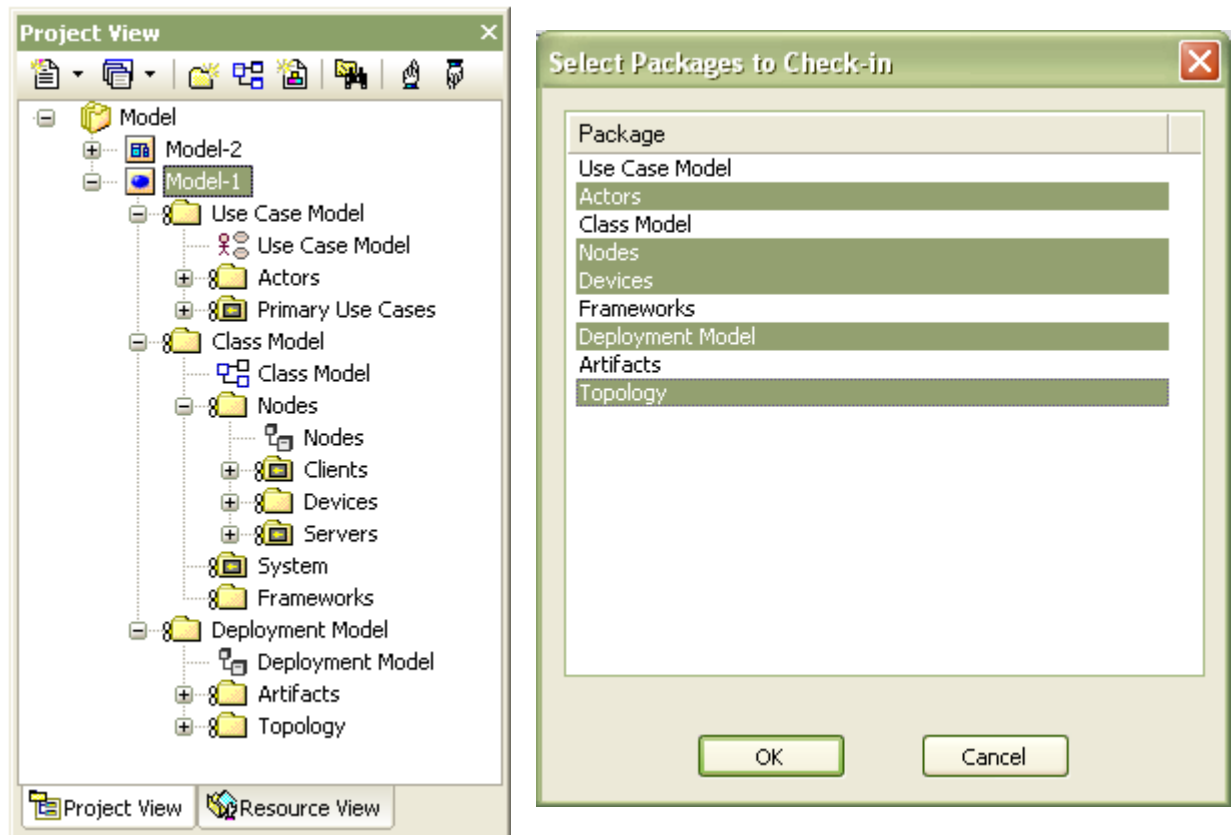
Check In/Check Out

1. In the Project Browser, right click on the package icon.
2. From the *Package Control* submenu, select *Check In*, *Check Out* or *Undo Checkout* as appropriate.
3. Optionally enter a comment if prompted to do so.
4. The Project Browser icon should change the user, when a package is checked in the package will be represented in the *Project View* as a package with a figure 8 to the left of the package and the package icon displayed normally, if the package has been checked in the icon changes to a small green horizontal cross. The example below shows the upper package as being a checked out package, whilst the lower package is checked in.



Check In Branch

1. In the Project Browser, right click on the package icon at the root of the model branch that is to be checked in.
2. A dialog will display, listing all version controlled packages within that branch that are checked out to the current user.
3. Select packages in the list, to be checked in. (Use Ctrl-Click to add or remove a selection, use Shift-Click to select between items.)
4. Click OK to check-in the selected packages.
5. Optionally enter a comment when prompted to do so. (This comment will be used for all packages that are checked in.)
6. The package icon will change to indicate that the packages have been checked-in.



8.13.4.10 Offline Version Control

Normally, when loading a model that uses version control, EA will initialize a connection to the version control system for each Version Control Configuration that is defined in the model. If EA is unable to connect a Version Control Configuration for any reason, it will display warning messages alerting the user to this fact and it will provide "offline" version control functionality for all packages associated with the failed connection.

Users can even prevent EA from attempting to make any version control connections by choosing "Project | Version Control | Work Offline" before loading a model. This is useful if you know that EA will not be able to connect to your version control system.

For example, if you are working at a laptop computer that is disconnected from your network and you have an EA model that uses a large number of Version Control Configurations, by choosing to work offline before you load the model, you will avoid all the error messages that EA would normally display as each version control connection attempt fails.

It is also possible to switch between working offline and working online at any time, either before or after a model is loaded. EA will disconnect or reconnect version control (depending on connection availability) according to your selection.

Using Version Control Whilst Disconnected From Your Version Control Server

From version 6.0 onwards, EA will "remember" the status of a model's version controlled packages.

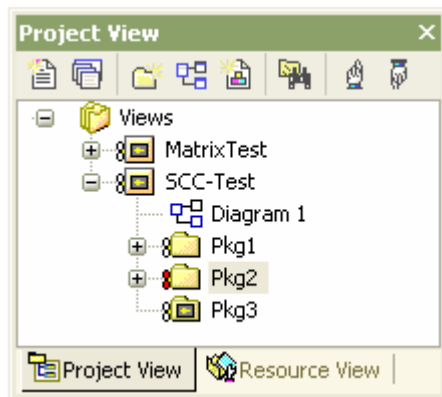
Packages that were checked out to you prior to disconnection from the server will still be shown as checked out to you, even though you are no longer connected to the server. You will still be able to edit these

packages as you normally would.

Packages that were not checked out to you prior to disconnecting from the server will be shown as version controlled and locked. These packages of course cannot be edited until they are checked out.

Offline Check Out

From version 6.0 onwards, it is possible to “check-out” a version controlled package, when your machine is disconnected from the version control server. In this way, it is possible to “check-out” and edit a package, even after you have disconnected from the version control server. In the example below the icon for Pkg2, indicates that it has been checked-out whilst offline.



Important Note: You should be aware that the version control system and therefore other users, will have no way of knowing that you have “checked-out” a package whilst offline. It is **not** possible to merge changes to an XMI file that result from two users editing the same package at the same time. As such, one set of changes will be lost, if an offline checkout leads to two people editing the same package at the same time.

Checking in a Package That Was Checked Out Offline

Once you reconnect your machine to the version control server, if the package you checked out offline is not currently checked out by another user, then you will be able to check in that package.

However, before EA checks in such a package, it compares the local working copy of the package file, against the latest revision in the repository. (These package files remain unchanged in your work area, until EA exports the package again before checking in.)

If the repository version remains unchanged from when you last updated your local copy, EA will export and check in your package without further prompting.

If on the other hand, the repository now contains a file that has changed since you last updated your local copy, checking in your package will overwrite whatever those changes may be. You will be warned of the pending data loss and given the opportunity to abort the check in.

It is not possible to merge the changes to a package’s XMI file that result from two users editing the same package at the same time. One set of changes will be lost!

At this point, you need to decide whether to discard your own changes, using the “Undo Check Out” command, or continue with your check in, thereby overwriting any changes that have been committed to the repository since you last updated your local copy from the repository.

You can use the *File Properties* command to determine who checked in the last changes to this package. This may help you to discover what changes have been uploaded and decide whose changes will prevail.

Update Before You Disconnect

Whenever you are connected to the version control server, if you modify a package, you will always be working with the latest version of that package. This is because you cannot modify a package until you check it out from version control, and checking it out, loads the latest revision from the repository into your model.

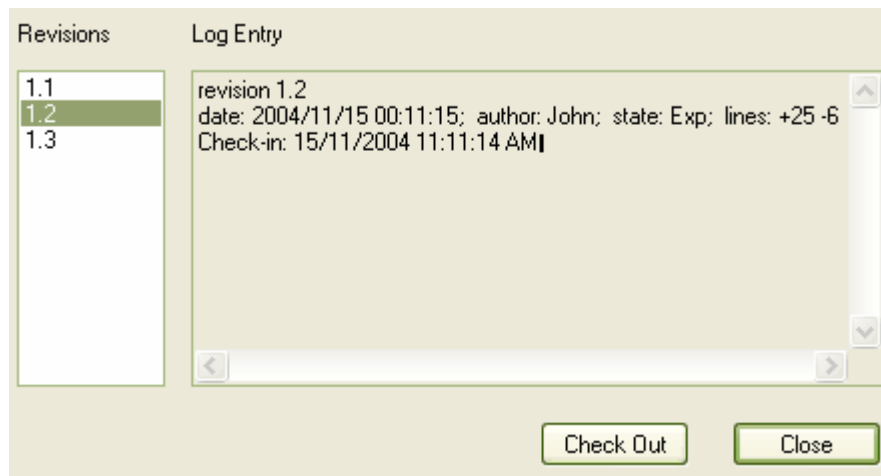
These rules do not apply when you are disconnected from the version control server. You will be working on whatever versions you have on your machine, dating back to the last time you updated your local copies of the version controlled packages. So, if you are planning to work on a model whilst disconnected from version control, it is a very good idea to make sure that you have the latest versions of all packages before you disconnect. The *Get All Latest* command makes this a simple task.

8.13.4.11 Review Package History

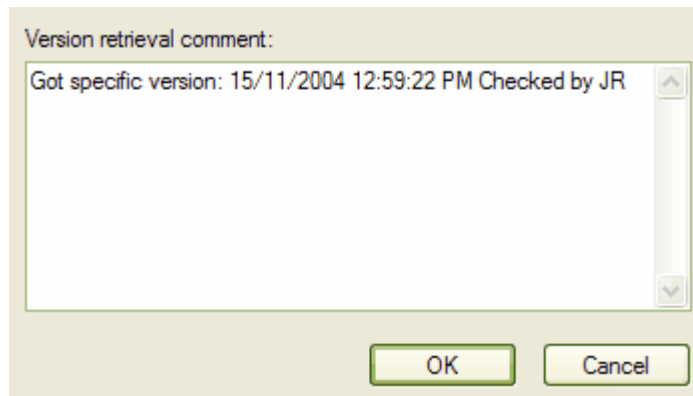
Reviewing package history allows users to view the history of checked in package revisions. This will open up the package in read-only mode allowing users to view the package contents, but not make any changes to the package. To review the package history use the following steps:

Review Package History

1. In the *Project Browser*, right click on the package configured for version control.
2. From the package context menu mouse over the Package Control submenu and select the File History option.
3. From the *Package Control* submenu, select *File History*. If the package has been configured for CVS the following instructions apply. If the package has been configured through SCC, the history will be accessed through the mechanism offered by the third party SCC provider.
4. This will open the *CVS File History* dialog, from here the user may view the log entries by clicking on the revision numbers in the *Revisions* field. To view the package history select a revision and then press the *Check Out* button.



5. This will bring up a dialog warning that the model will be opened in read-only mode. Press the *Yes* button to continue or *No* to cancel the action.
6. The *Add Comments* dialog allows the user to add comments regarding the retrieval of the package history. Enter the comments in the text field then press the OK button to proceed.



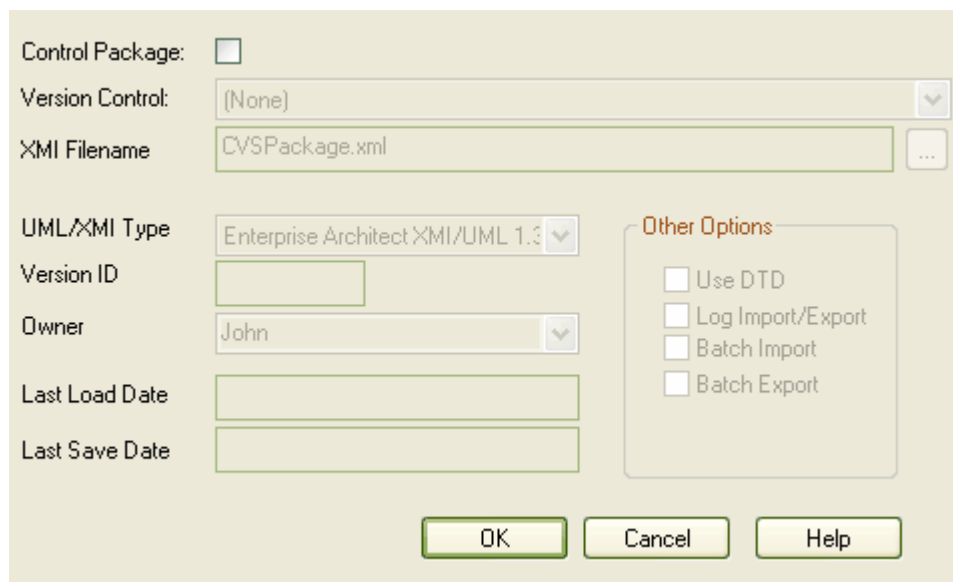
7. The package will then be retrieved in read only mode to allow the user to view the history of the package at the specified version. To go back to the latest version check in the package by using the check in command from the Project View by right clicking on the package and from the context menu mouse over the *Package Control* item and from the submenu select the *Check In..* option.

8.13.4.12 Remove Package from Version Control

Packages may have their connections to version control removed by using the following steps:

Remove a Package from Version Control

1. In the *Project Browser*, right click on the package configured for version control.
2. From the *Package Control* submenu, select *Configure* or press the *Ctrl + Alt + P* hotkey combination.
3. Uncheck the *Control Package* checkbox.



4. Press *OK* to remove from version control.

5. The package will now be removed from version control.

Note: before removing a package from version control the package must be checked in.

8.13.4.13 Add Previously defined Version Control Configurations

Once a version control configuration has been defined in one model it is possible to add the configuration to other models. To use this feature follow the instructions detailed below:

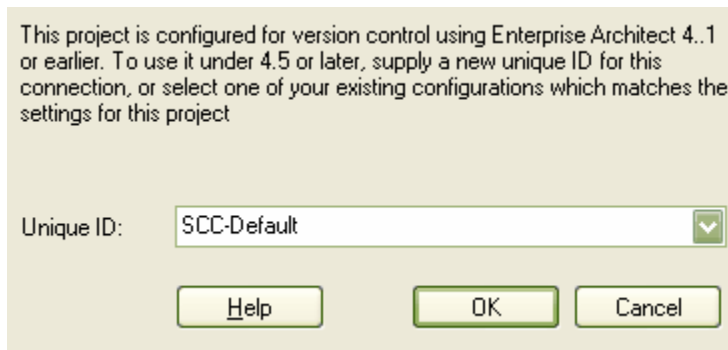
1. Open the model that is to have the predefined version control configuration added to it.
2. Right click on any package in the Project View and select *Package Control | Version Control Options*. The *Version Control Configurations* dialog will appear.
3. Click on the *New* button.
4. From the *Unique ID* drop down menu, select one of the previously defined version control configurations.
5. Press the *Save* button to confirm the version control configuration.

8.13.4.14 SCC Version Control Upgrade for 4.5

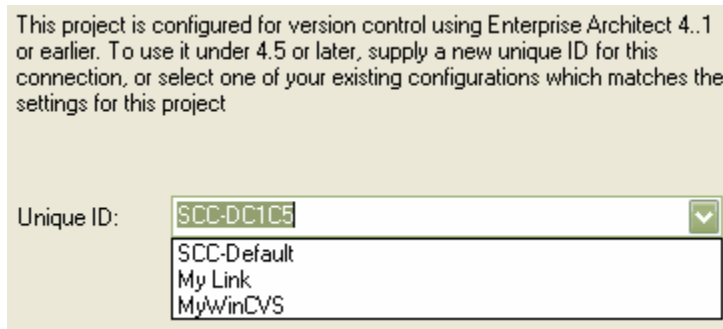
When a version controlled project created under a version of EA earlier than 4.5 is opened in version 4.5 (or later) the SCC connection must be identified with a new unique ID. The user may assign a name to the existing SCC configuration or associate the project with a configuration that has previously been assigned a unique ID.

By having a unique ID for version control configurations, the user may assign a version control configuration quickly and efficiently. This is achieved by using configurations that have been created previously for other version controlled repositories. This allows the many packages to be configured to use an existing version control repository, this may apply to more than packages created for more than just one model allowing for a great deal of flexibility. To upgrade an existing SCC version control project created before version 4.5, in version 4.5(or later) of EA use the following steps.

1. Open a project with a SCC version control configuration created in EA earlier than version 4.5 (or later).
2. The *Select or Create Unique ID for Version Control* dialog will prompt the user to create an ID for an existing configuration or choose a previously created one from the *Unique ID* drop down list.
3. The existing SCC configuration will be the initial value which will be represented by the value SCC-XXXXXX, this number is not especially meaningful, therefore it is recommended that the configuration be given a meaningful name.



4. The user may associate the version controlled package with previously defined configuration by selecting an existing configuration from the *Unique ID* drop down list (if one exists).



5. After the selections have been made press the *OK* button to proceed loading the model.

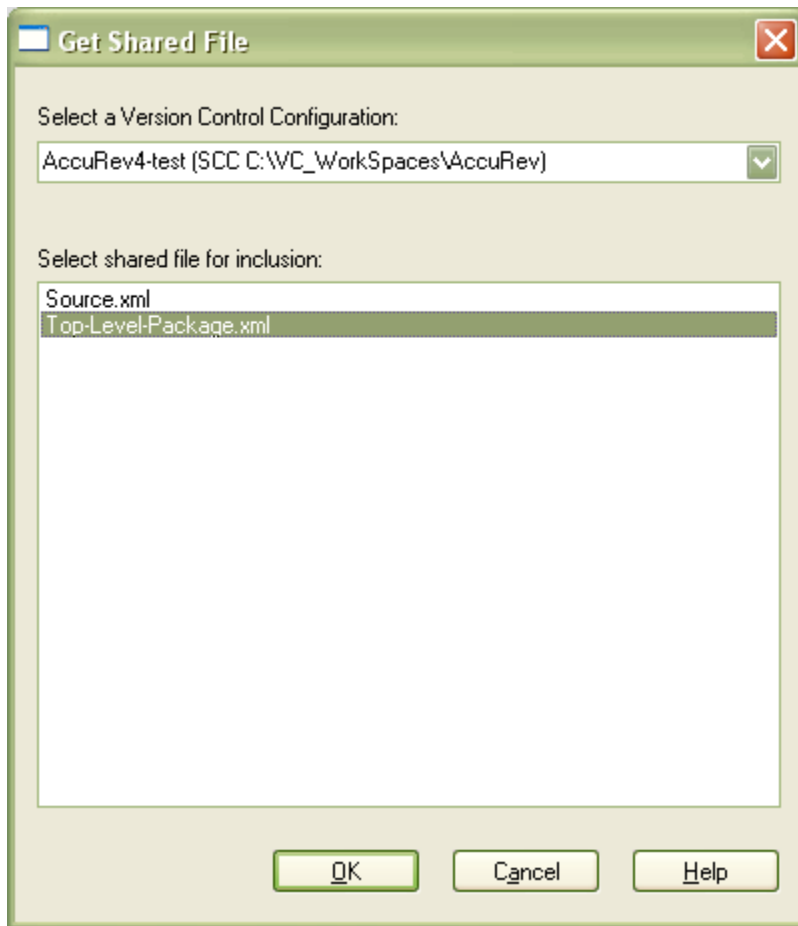
8.13.4.15 Including Other Users Packages

Packages that have been created by other users, or by yourself in another model, can be retrieved from version control and imported into your current model, by using the *Get Package* command.

Other users may be creating packages that you wish to use in your model. If you are not sharing a SQL database or EAP file, those packages will not automatically become part of your model. If the packages have been placed into version control, they may be retrieved and imported into your model as children of an existing package, using the *Get Package* command.

You must have access to the package files through the version control system and you must define a version control configuration through which to access those files. The version control configuration must use the same *Unique Id* that was originally used to add the package to version control.

1. In the Project Browser, right-click on a package that you wish to use as the parent of the incoming package.
2. From the *Package Control* submenu, select *Get Package*.
3. The following dialog will open.



4. From the drop-down list, select the version control configuration associated with the package you wish to retrieve. The file list will then be populated with the names of files available through that configuration, for retrieval and import into your model.
5. Select the package file you wish to import into your model and click **OK**.

8.13.4.16 Specifying Private or Shared Models

In the *Version Control Configurations* dialog, the *Private Model* option affects the availability of the *GetLatest / GetAllLatest* functionality. These *Get* functions **cannot be used with shared models**.

The reason for this is to prevent the possibility of overwriting any new modifications to a package, with out-of-date versions of the package that are retrieved from version control.

Consider the following scenario, with two users, both working on a **shared model** at the same time.

If user1 checks-out and modifies a package within EA, user2, who also has this shared model open in EA, can see the changes to the model immediately.

If, at this point, user2 was to perform a *GetLatest* on the package (or a *GetAllLatest*), he would retrieve from Version Control, a package that is now out of date (user1 has it checked-out and has made changes that are yet to be checked-in) and the out-of-date package would be imported into the model

loaded on user2's machine, immediately updating the model database (either the shared .EAP file or the DB repository) overwriting the modifications that user1 has just made.

Similarly, if a user checked-out and modified a package and then performed a *GetLatest* on that package, the modifications would be overwritten by the version of package retrieved from Version Control.

The *Get Latest* function is consequently disabled, for packages that are checked-out to the current user.

8.13.4.17 Using Nested Version Control Packages

From version 5.0 of Enterprise Architect it is possible to use nested packages much more efficiently and effectively than in the past.

When saving a package to the version control system, only stub information is exported for any nested packages. This ensures that information in a nested package is not inadvertently over-written by a top level package.

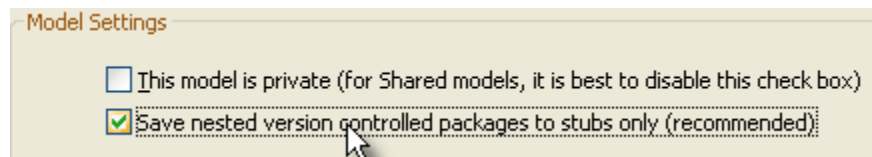
When checking out a package, EA does not modify or delete nested packages, only the top level package is modified.

As a consequence of this new behaviour, if you check out or get a version controlled package with nested packages not already in your model, you will see stubs in the model for the nested packages only. If you do a "GetAllLatest" call from the version control menu, EA will populate these new stubs from the version control system.

Using the above technique it is now possible to populate a large and complex model from only the root packages, using GetAllLatest to recursively iterate through the attached and nested packages.

This is now a much more powerful and efficient means of managing your project and makes handling very large models, even in a distributed environment, much simpler.

It is recommended you do not mix EA version 5.0 and previous versions when sharing a version controlled model. If this is necessary it is best to go to the version control options screen and uncheck the "Use nested package stubs" check box, defaulting EA back to the version 4.5 behaviour (for the current model only).



8.14 Model Sharing and Team Deployment

Introducing Team Development

EA offers a diverse set of functionality designed specifically for sharing projects in team-based and distributed development environments. Project sharing can be achieved through network deployment of model repositories, replication, XMI Import/Export, Version Control, Package Control and User Security.

Network deployment may be undertaken using two different schemas for deployment, either using:

- .EAP based repositories or
- DBMS server based repositories.

Replication requires the use .EAP based repositories, and cannot be performed on repositories stored on a DBMS server. DBMS server based repositories offer better response times than .EAP files on networks due to the inherent structure of the DBMS. DBMS also offers a better solution when networking problems are encountered, as they have the ability to backtrack transactions caused by external breakdowns.

Replication

Replication is a simple process which allows for data interchange between .EAP based repositories (not DBMS) and is suitable for use in situations where many different users work independently. Modelers merge their changes into a Design Master on an "as required basis". With replication it is recommended that a backup is carried out prior to replication.

XMI Import Export

The XMI Import/Export can be used to model discrete packages that may be exported and be shared between developers. XMI allows the export of packages into XML files which may then be imported into any model.

Package control can be used to set up packages for version control and allow for the batch export of packages using XMI. Version Control allows for a repository to be maintained by a third -party source code control application which is used to control access and record revisions.

Security

User security is used to limit the update access to model elements. It provides control over who in a project can make changes to model elements.

For more information regarding the use of EA with shared models and team deployment please see the Deployment of Enterprise Architect white paper available from www.sparxsystems.com.au/downloads/whitepapers/EA_Deployment.pdf.

Note: DBMS Repository support and User Security are only available with the Corporate Edition of Enterprise Architect.

See Also

- [Distributed Development](#)
- [Project Sharing](#)
- [XMI Import/Export](#)
- [Replication](#)
- [Package Control](#)
- [Version Control](#)
- [User Security](#)

8.14.1 Sharing an Enterprise Architect Project

Note: Project Sharing and Replication are only enabled in the Professional and Corporate versions of Enterprise Architect

Sharing a project among a team of designers, developers and analysts is the most efficient way of using EA to manage a team development. Many people can work on the model at the same time and contribute their particular skill. Team members can always see what the latest changes are, keeping the team informed and up to date with the project status.

Sharing an EA Project

You may share an Enterprise Architect project in three ways:

1. Using a [shared network directory](#). In this scenario you place the project file on a shared network drive. Individual developers and analysts can then open and work on the project concurrently. Note that some project views (especially the Project Browser) will require occasional refreshing to see changes made by other users.
2. Using replication. [Replication](#) is a powerful means of sharing projects between isolated or mobile users. In the replication scenario a project is converted to a design master, then replicas made of the master. Users take the replicas away, modify the project, then bring their replicas back to be synchronized with the master file.

3. Using a shared DBMS based repository (Corporate Edition only).

8.14.2 Sharing a Project

The easiest way to share a project amongst a work group of developers and analysts is to place the project file on a shared network drive and have people connect concurrently from their Workstation.

Note: *Enterprise Architect will accept a number of concurrent connections without issue, although there may be occasional 'lock-outs' when one user wishes to access or update something another user is in the process of modifying.*

Network Issues

The main issues with shared network access are:

- Changes to the Project Browser are not automatically updated. To compensate for this, users must occasionally reload their project to view any project changes at this level.
- If two or more people work on the same diagram concurrently, unexpected results may occur. It is best to allow only one analyst to work on a diagram at a time.
- If a User's machine crashes, the network suffers an outage or a machine is turned off unexpectedly, the project file may require repair, to compensate for the sudden inconsistency. A [repair](#) facility is provided (select *Repair .EAP File* from the *Tools | Manage .EAP File* submenu) to carry out this task. This only applies to the file based version of EA - the DBMS based version does not suffer this problem.

8.14.3 Distributed Development

Enterprise Architect supports distributed development using two different techniques as described below.

Replication

Use the Replication features to allow geographically separated analysts to update and modify parts of the model in replicas, then merge these back together at a central location.

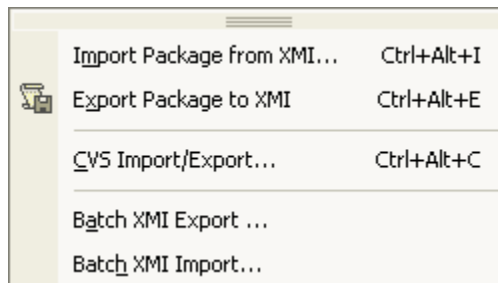
For further information see the [Replication](#) section.

XMI Import/Export

Use the XMI based Import/Export facility to model discrete packages, export to XML and share among the development team. This approach has several benefits over replication:

1. You can assemble a model from only the parts you need to get your job done.
2. You can assemble a full model if required.
3. You can assemble a model from different package versions for different purposes (eg. customer visible, internal release only etc.).
4. You can roll-back parts of a model on an as required basis.
5. There is less chance of 'collisions' between developers if each works on a discrete package.
6. The process is controllable using a [version control](#) system.

Use the [Import and Export menu options](#) (below) to access this feature - they are available through the *Project | Import/Export* submenu. Also see the [XMI section](#) for further information about XMI based import and export.



The [Controlled Package](#) feature can also be used to assist in the process.

Note: XMI based import/export is UML1.3 / XMI1.1 compliant. Users can also write XML based tools to manipulate and extract information from XML files to enhance the development process.

8.14.4 User Security

What is User Security in EA?

User security in EA can be used to limit the access to update functions within the model. Elements may be locked on a per-user or per-group basis and where user security is enabled a password is required to log in to the model. Security in EA is not designed to prevent unauthorized access: rather it is intended to provide a means of improving the collaborative design and development by preventing concurrent editing and limiting the possibility of inadvertent model changes by users not designated as model authors.

User Security basics

User security can be switched on in the Corporate Edition of Enterprise Architect. The user security in EA offers two security policies, the standard security model and the rigorous security model. In the standard security model all elements are considered unlocked, and as the need arises, a user may lock any element or set of elements at the user or group levels depending on permission rights that the user has to the model. The rigorous security model assumes that everything in the model is locked until explicitly checked out with a user lock. In this mode, an EA model is read-only until a user applies an editing lock on one or more elements. For more detailed information regarding the security policies view the [security policy](#) topic.

User Security Tasks

There are various security tasks which may only be performed by users with Administrative rights to the model, these tasks include:

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#) (only available for Oracle 9i and 10g and SQL Server Repositories).
-

Other Security tasks can be performed by the users who do not have Administrative rights, these tasks include:

- [Locking Model Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

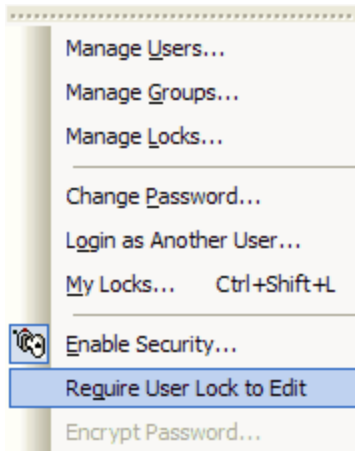
Note: User security is not enabled by default in EA you must [enable it](#) first.

8.14.4.1 Security Policy

There are two possible security policies in Enterprise Architect:

1. In the first mode, all elements and diagrams are considered unlocked, and as the need arises, a user may lock any element or set of elements at the user or group level. This mode is good for cooperative work groups where there is a solid understanding of who is working on which part of the model, and locking is used mainly to prevent further changes, or to limit who has access to a part of the model.
2. The second mode is more rigorous. It assumes everything is locked until explicitly checked out with a user lock. In this mode, an EA model is read-only until a user applies an editing lock to one or more elements. A single 'check out' function is available on a diagram to check out the diagram and all contained elements in one go. There are also functions on the context (right-click) menus of packages, diagrams and elements in the model browser to apply a user lock when this form of mode is in use. You would use this mode when there is a strict need to ensure only one person can edit a resource at one time. This is suitable for much larger projects where there may be less communication between users.

Toggle between these modes using the *Require User Lock to Edit* menu option in the *Project | Security* submenu:



Note: When you add new elements in Mode 1 (elements editable by default), no user lock is created automatically for the newly created element.

Note: When you add new elements in Mode 2 (elements locked by default), a user lock is created on the new element to allow instant editing.

See Also

- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)

- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

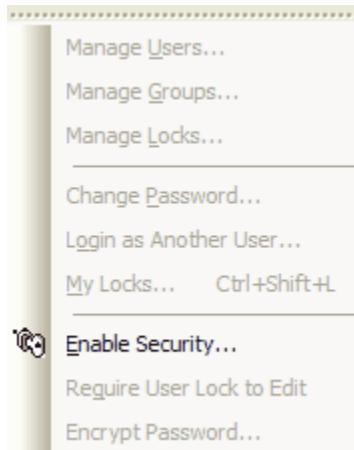
8.14.4.2 Enabling Security

User security is not enabled by default in EA. To enable security you can follow the instructions below.

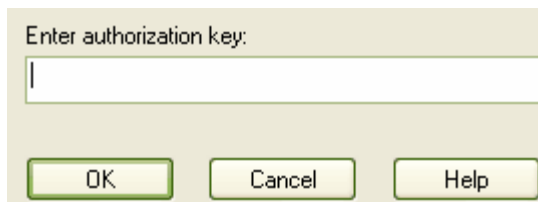
Enable Security

To enable security in EA, follow these steps:

1. Open the *Project* | *Security* submenu.
2. Select the *Enable Security* option.



3. In the Authorization dialog that appears, enter the key provided when you purchased EA.



4. If you enter the correct code and press *OK*, security is enabled. An Admin user is created with full permissions and a password of 'password'.
5. Close EA then reopen it, logging in as Admin.
6. You may now set up users and permissions as required.

**See Also**

- [Security Policy](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

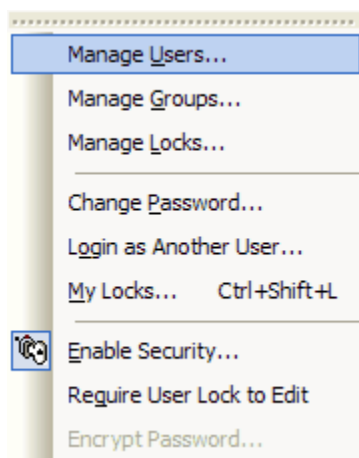
8.14.4.3 Maintaining Users

If you enable security, you will have access to the *System Users* dialog which you can use to set up more users for your model.

Set Up a User

To set up a user for your model, follow the steps below:

1. From the *Project* | *Security* submenu, select the *Maintain Users* option.



2. The *System Users* dialog will appear.

User Details

Login:

Firstname:

Surname:

Department:

Set Permissions

Accept Windows Authentication

Users:

| Surname | Firstname | Login |
|---------------|-----------|-------|
| Administrator | The | admin |

- You can use the *System Users* dialog to set up new users by providing their name and other details. You can also assign them to groups, set up their *Single Permissions* or *View All* permissions for the currently selected user

Note: If you tick the *Accept Windows Authentication* option, on opening the model EA will check the users database for your Windows ID and if it matches will automatically log you in without prompting for a password. The EA model administrator will need to ensure that all Windows ID are entered in the user database with the appropriate permissions and have a password set-up to prevent rogue access to the model. The User ID should be in the following format : <DOMAIN>\<username>.

See Also

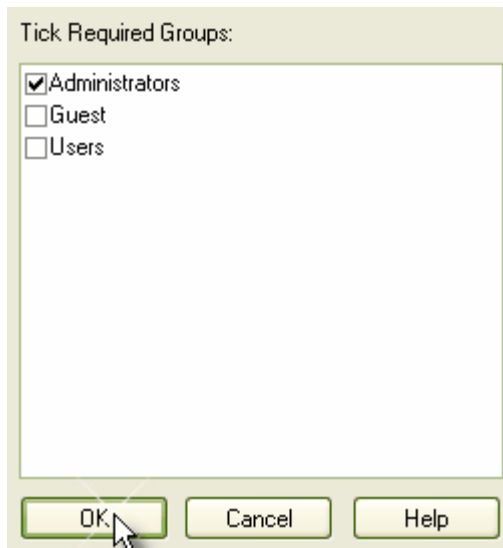
- [Security Policy](#)
- [Enabling Security](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)

- [Managing Your Own Locks](#)

8.14.4.4 Set Up User Groups

To set up user groups follow these steps:

1. From the *Project | Security* submenu, select the *Maintain Users* option.
2. Click on *Group Membership* on the *System Users* dialog.
3. In the *User Groups* dialog, tick all of the groups this user will belong to.



4. Press *OK* to assign the user to the group/s.

Note: To create new user groups, see the [Maintaining Groups](#) topic.

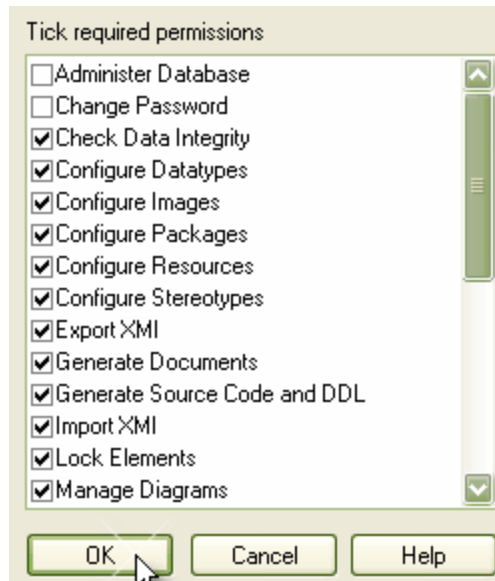
See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.5 Set Up Single Permissions

You can set specific user permissions from the *User Permissions* dialog. Specific user permissions are added to permissions from group membership to provide an overall permission set. To set up single permissions for a user follow the steps below:

1. From the *Project | Security* submenu, select the *Maintain Users* option.
2. Click on *Single Permissions* on the *System Users* dialog.
3. In the *User Permissions* dialog, select the specific permissions that you want to apply to this user.



4. Press *OK* to assign the selected permissions to the user.

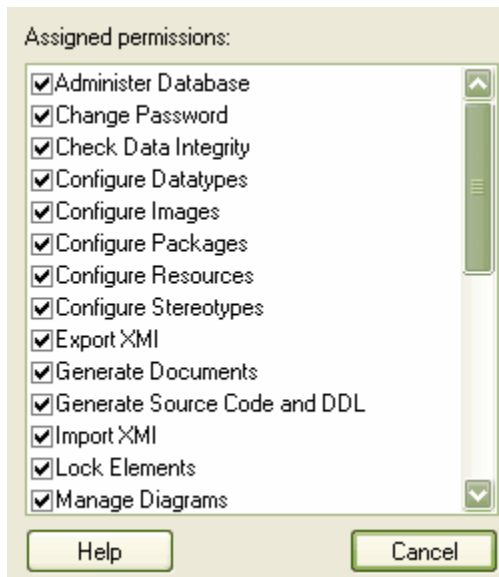
Note: A user's total permissions are those granted by Group Membership + those granted by Specific permission assignment.

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.6 View All User Permissions

The *All user permissions* dialog shows a list of all permissions a user has - derived from their individual profile and from their membership of security groups. It can be found by selecting *Maintain Users* from the *Project | Security* submenu, then selecting the required user and clicking on *View All*.



See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.7 Maintaining Groups

Security groups make it easy to configure sets of permissions and have them apply to a number of users in one action.

Set Up a Security Group

To set up a security group, follow the steps below:

1. Select *Maintain Groups* from the *Project | Security* submenu.
2. The *Security Groups* dialog will appear.

Group Name: BusinessDept

Description: Business department users

Set Group Permissions

Groups: New Save Delete

| Group Name | Description |
|----------------|-----------------------|
| Administrators | System Administrators |
| Guest | Guest |
| Users | Users |

Help Close

3. Specify a *Group Name* and a *Description*.
4. Press *Save*.

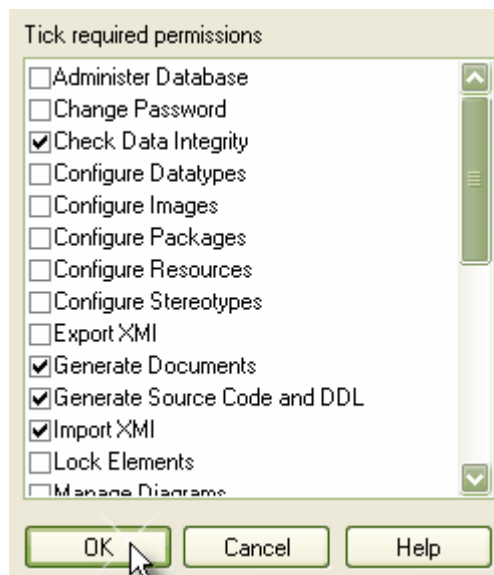
See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.8 Set Group Permissions

To set up permissions to apply to a security group, follow the steps below:

1. Select *Maintain Groups* from the *Project | Security* submenu.
2. Press *Set Group Permissions* on the *Security Groups* dialog.
3. In the Group Permissions dialog tick the required permissions.



4. Press **OK** to assign the permissions. All of the users assigned to this group will share in this set of permissions.

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.9 List of Available Permissions

The following table lists the available permissions in the Corporate Edition of EA and their meaning:

| Permission | Meaning |
|--------------------------------|--|
| Security - Enable/Disable | Enable or disable user security in Enterprise Architect. |
| Security - Manage Users | Maintain users, groups and assigned permissions. |
| Security - Manage Locks | View and delete element locks. |
| Manage Reference Data - Update | Update and delete reference items. |
| Update Diagrams | Update diagram properties and layout, including the Page Setup dialog. |
| Administer Database | Compact and repair database. |
| Manage Replicas | Create and synchronize replicas. |
| Lock Objects | Lock an element. |
| Manage Project Information | Update and manage resources, metrics, risks etc. |

| | |
|---|---|
| Configure Resources | Import and manage resources tab items: patterns, profiles, favorites etc. |
| Update Elements | Save changes (including delete) for elements, diagrams, packages, links, etc. |
| Export XMI | Export model to XMI. |
| Import XMI | Import model from XMI. |
| Manage Tests | Update and delete Test records. |
| Manage Issues | Update and delete Issues. |
| Change Password | Change password of current user. |
| Transfer Data | Transfer model between different repositories. |
| Check Data Integrity | Check and repair model integrity. |
| Configure Datatypes | Add/Modify and Delete Datatypes. |
| Configure Stereotypes | Add/Modify and Delete Stereotypes. |
| Configure Images | Configure alternate element images. |
| Generate Source Code and DDL | Generate source code and DDL from model element. Synchronize if already exists. |
| Reverse Engineer from DDL and Source Code | Reverse engineer from source code or ODBC. Synchronize with model elements. |
| Generate Documents | Generate RTF and HTML documents from model packages. |
| Configure Packages | Configure controlled packages and package properties. |
| Manage Diagrams | Create new diagrams, copy existing and delete diagrams. Also save diagram as UML Pattern. |
| Spell Check | Spell check package and set spell check language. |
| Configure Version Control | Set up Version Control options for the current model. |
| Use Version Control | Check files in and out using Version Control. |

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.10 View and Manage Locks

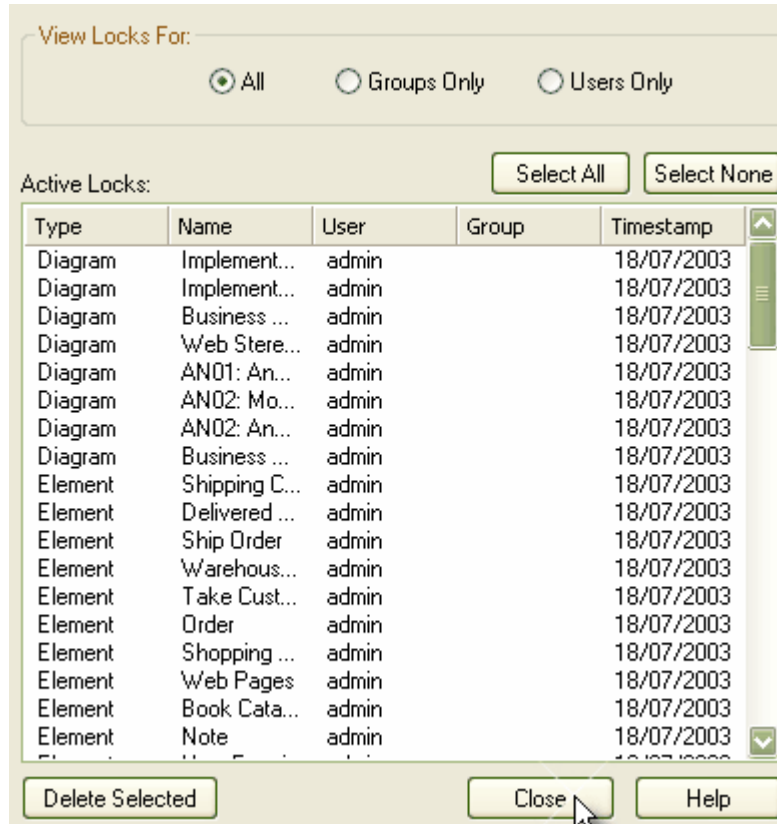
From time to time it may be necessary to examine or delete locks placed on elements by users. To do this, a function is provided to view and manage active locks

Delete a Lock

To view locks, and delete if necessary, follow the steps below:

1. Select **View and Manage Locks** from the **Project | Security** submenu.
2. From the **Active Locks** dialog (shown below), select the type of lock to view - All locks, Group locks only or User locks only.

3. To delete a lock, highlight it and press *Delete Selected*.
4. *Close* the dialog when finished.



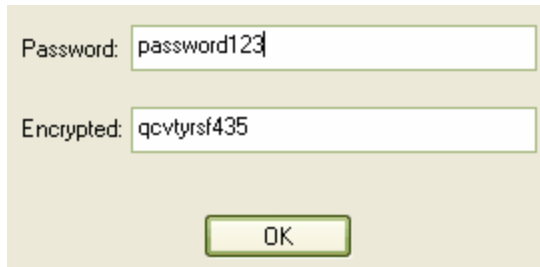
See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.11 Password Encryption

Users of SQL Server or Oracle 9i and 10g repositories have the option of encrypting the password used to set up the connection of EA to the repository. The EA user will not have the real password thereby preventing the user accessing the repository using other tools such as Query Analyzer or SQLPlus.

Once security is enabled, the administrator needs to log on to access the dialog to create encrypted passwords. To encrypt a password open the *Project* menu, then mouse over the *Security* menu option and choose *Encrypt Password* from the submenu. In the example below the Password "password123" is used as the password to access the repository.

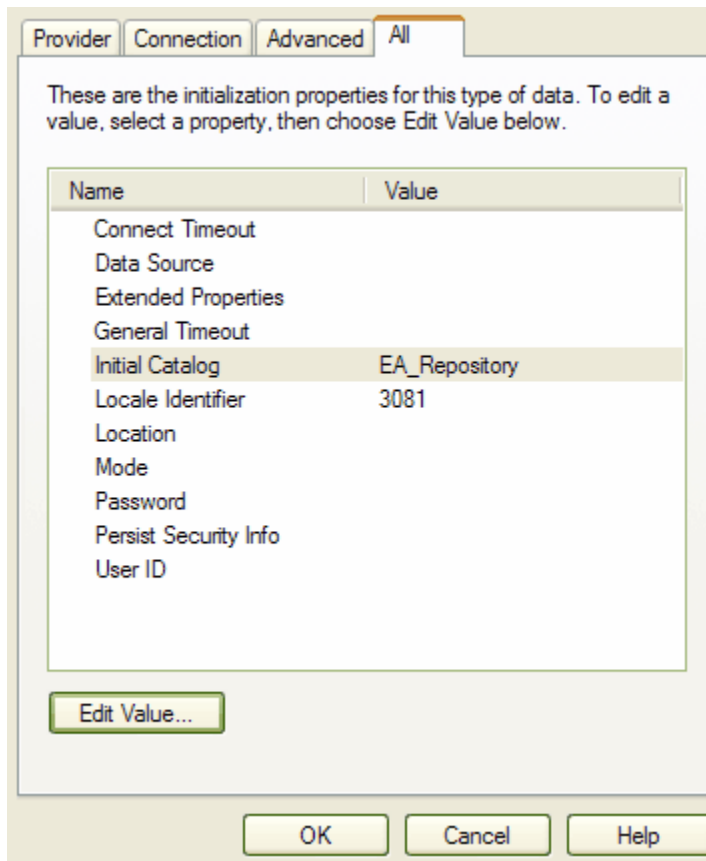


The image shows a dialog box with a light beige background. It contains two text input fields. The first field is labeled 'Password:' and contains the text 'password123'. The second field is labeled 'Encrypted:' and contains the text 'qcvtysrf435'. Below these fields is a single button labeled 'OK'.

To connect EA to the repository, the user enters the encrypted password prefixed with \$\$, so the encrypted password will become '\$\$qcvtysrf435'. Do not use the *'Test Connection'* button as it will cause an error with encrypted passwords.

For more information relating to connecting to Oracle 9i and 10g and SQL Server look at the to [Oracle 9i and 10g Data Repository](#) and Connect to [SQL Server Data Repository](#) topics respectively.

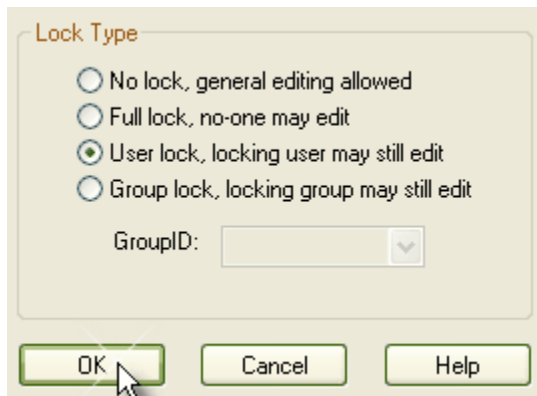
Note: For SQL Server repositories, the user will need to enter the Initial Catalog details from the *All* tab of *Data Link Properties*.

**See Also**

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.12 Locking Model Elements

With security enabled, locking a model element is slightly different to when security is not enabled. If you have the *Project | Security | Require User Lock to Edit* option turned off, when you elect to lock a diagram or element, the *Element Lock* dialog will appear:



There are 4 lock options available:

- No Lock. Do not lock this element - clear any existing lock.
- Full lock. Lock this element so no-one can edit it.
- User lock. Lock this element so only the locking user may make further edits.
- Group lock. Lock this element so any member of the group specified may update the element - but others are excluded.

Select the appropriate lock and press **OK**.

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

8.14.4.13 Adding Connectors Between Locked Elements

When working with locked elements, the ability to add connectors depends on the locked status of the source and target elements. The rules are:

- **Source unlocked, target unlocked:** any kind of connector may be added.
- **Source locked, target unlocked:** prohibited, except for composition connectors.
- **Source unlocked, target locked:** allowed, except for composition connectors.
- **Source locked, target locked:** prohibited for all connectors.

In summary, a connector can be added if its source is unlocked, regardless of the locking state of the destination (think of it as modifying what the source can see). The exception is composition connectors, where

the target (i.e. parent) must be unlocked (think of it as modifying the parent by adding children).

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

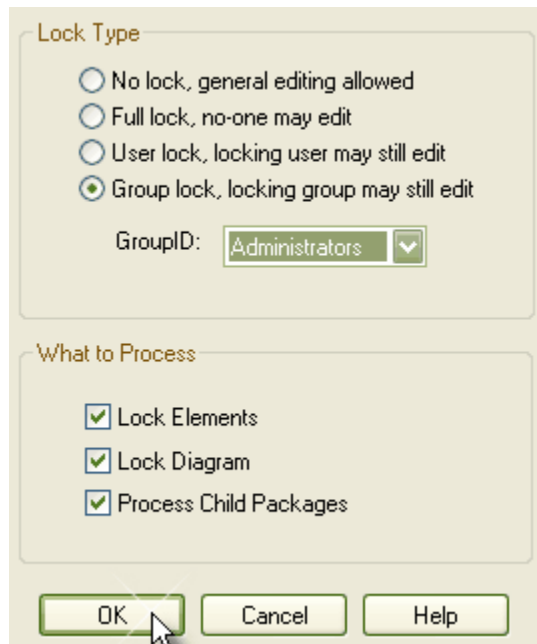
8.14.4.14 Locking Packages

You can lock all the contents of a package (and optionally all contents in child packages) in one step, using the *Lock Package* function. The locks are automatically applied to elements and to diagrams - as if they had been individually set or cleared. Lock types and details are the same as for [locking a single element](#).

Lock a Package

To lock a package, if you have the *Project | Security | Require User Lock to Edit* option turned off, follow these steps:

1. In the Project Browser, right click on the package you wish to lock.
2. Select the *Lock Package Contents* menu option.
3. In the *Lock/Unlock Package(s)* dialog, select the *Lock Type* to apply, whether to lock element and/or diagrams, and whether to process package children (ie. lock the whole branch).



4. Press **OK** to apply the lock.

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

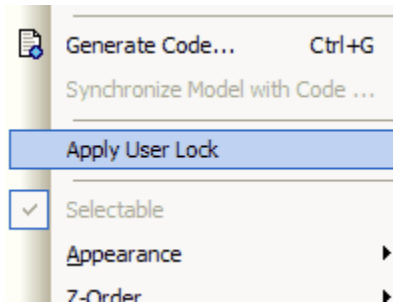
8.14.4.15 Apply a User Lock

In Mode 2 Security, where a User Lock is required before any edit can occur, you can use the following menu functions.

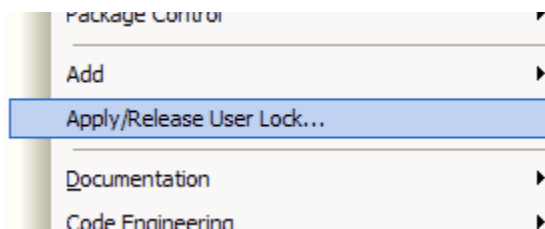
Right click on a diagram or an element to open its context menu. Select the **Apply** or **Release User Lock** option.

Note: The menu option will be **Apply** if there is no user lock currently on, it will be **Release** if there is currently a user lock applied.

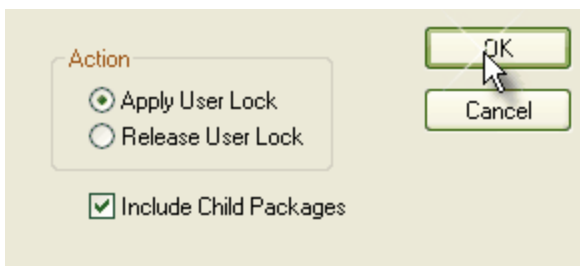
EA will apply or release the lock for the diagram and for any elements contained in the diagram.



In the Project Browser you can right click on Packages, Diagrams and Elements and apply a User Lock to the selected item.



When selecting items from the Project Browser, you will get the following dialog. Tick whether you wish to apply or release a user lock on the selected item.



Note that for a package, you can elect to also lock all child packages at the same time. If any elements in the package tree are locked by other users, you will receive a list of elements that couldn't be locked at the end of the process.

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)

- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Discovering Who Has Locked An Object](#)
- [Managing Your Own Locks](#)

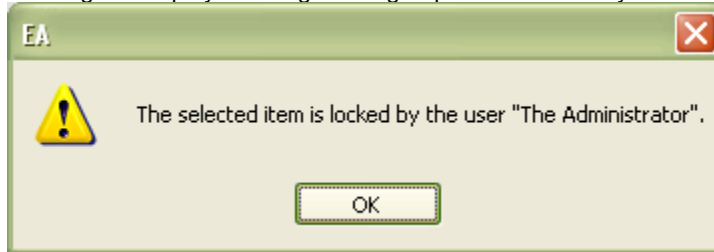
8.14.4.16 Discovering Who Has Locked An Object

If you right-click in the Project View on a Diagram, Package or Element that is locked by another user or user group, then choose "Lock..." from the context menu, a dialog will display, showing which group or user currently holds the lock on that item.

Users can discover who has applied a lock to a particular locked package by following these instructions.

- Right-click on the locked Object
- Select Lock... from the context menu

A dialog will display showing which group or user currently holds the lock on that item.



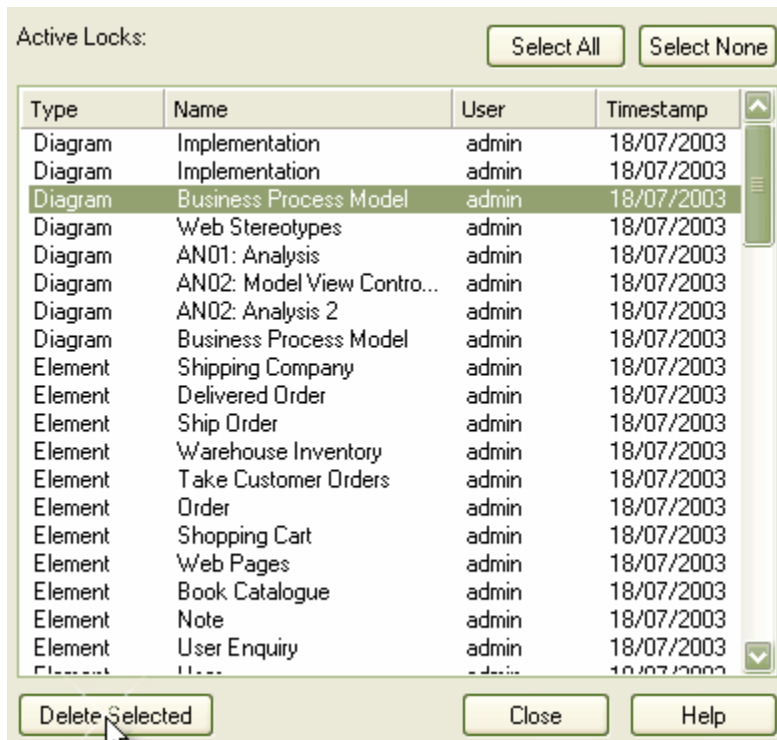
See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Managing Your Own Locks](#)

8.14.4.17 Managing Your Own Locks

It is possible to view and delete your own user level locks in EA. This is especially useful when working in Mode 2 security (user locks required to edit).

To manage your locks use the *Project* | *Security* | *Manage My Locks* menu item. This will open the *Active Locks* dialog.



In the *Active Locks* dialog you can select one or more locks and delete it by pressing *Unlock Selected*.

See Also

- [Security Policy](#)
- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [Set up Single Permissions](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [Set Group Permissions](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#)
- [Locking Model Elements](#)
- [Adding Connectors Between Locked Elements](#)
- [Locking Packages](#)
- [Apply a User Lock](#)
- [Discovering Who Has Locked An Object](#)

8.14.5 Replication

In addition to sharing Enterprise Architect projects in real time over a network, projects may also be shared using *Replication* - options for which are available through the *Tools | Manage .EAP File* submenu.

Replication allows different users to work independently of one another, and to merge their changes at a later time. To avoid difficulties in this inevitably hazardous process, please read all topics in this section carefully.

Enterprise Architect Merge Rules

Enterprise Architect uses follows these rules in merging:

- Additions are cumulative - i.e. Two replicas creating 3 new classes each will result in 6 new classes after merging.
- Deletions prevail over modifications, if one replica changes a class name and other deletes the class, performing a merge will result in both files losing the class.
- Conflicting modifications appear in the "Resolve Replication Conflicts" dialog under *Tools | Manage EAP File*.

See [Resolving Conflicts](#) for details on how to deal with conflicting modifications.

Using Replication

To use replication, follow these steps:

1. Convert the base project to a [Design Master](#) using the *Make Design Master* option in the *Tools | Manage .EAP File* submenu).
2. Create replicas from the design master using the *Create New Replica* option in the *Tools | Manage .EAP File* submenu.
3. Take the replica away and work on it as required, then bring it back for synchronization with the design master.
4. [Synchronize the replicas](#). During synchronization, all changes to both the master and the replica are propagated in both directions, so at the end they both contain the same information.

Avoid Change Collisions

If two or more people make changes to the same element, eg. a class, Enterprise Architect will arbitrarily overwrite one person's change with other other's. To avoid this, different users should work on different packages.

However, since Enterprise Architect does not enforce this rule, it is possible for users' work to conflict. To minimize the difficulties this causes, please note the following guidelines:

- If users are likely to have worked in the same area of the model - they should both witness the synchronization and confirm that they are happy with the net result.
- If small pieces of information have been lost, they should be typed into one of the merged models after synchronization.
- If a large piece of information has been lost - for example, a large class note that was overwritten by another user who had made a minor change to the same class use the [Resolve Replication Conflicts](#) dialog.

Disable or Remove Replication Features

If you have converted a project to a [Design Master](#) but now wish to disable the replication features, use the *Remove Replication* option in the *Tools | Manage .EAP File* submenu. Make sure you back up all your files first!

See Also

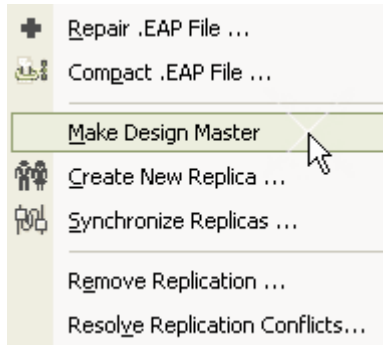
- [Creating Replicas](#)
- [Design Masters](#)
- [Synchronizing Replicas](#)
- [Remove Replication](#)

- [Upgrading Replicas](#)
- [Resolving Conflicts](#)

8.14.5.1 Creating Replicas

To create a replica, follow these steps:

1. First create a [Design Master](#), then select the *Create New Replica* option in the *Tools | Manage .EAP File* submenu and follow the instructions.



2. This process will create a replica of the current project which may then be modified independently, and afterwards re-merged with the main project.

See Also

- [Design Masters](#)
- [Synchronizing Replicas](#)
- [Remove Replication](#)
- [Upgrading Replicas](#)
- [Resolving Conflicts](#)

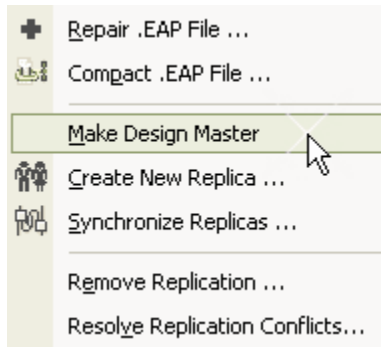
8.14.5.2 Design Masters

A *design master* is the first converted Enterprise Architect project that supports replication. From the design master you may create replicas, which may be modified independently of the master project, and re-merged at a later date.

Create a Design Master

To create a design master, follow these steps:

1. Load an Enterprise Architect project (always back up the original first!).
2. Select *Make Design Master* on the *Tools | Manage .EAP File* submenu and follow the instructions.



Tip: Replication is a powerful means of using Enterprise Architect in a work group or multi-user scenario.

See Also

- [Creating Replicas](#)
- [Synchronizing Replicas](#)
- [Remove Replication](#)
- [Upgrading Replicas](#)
- [Resolving Conflicts](#)

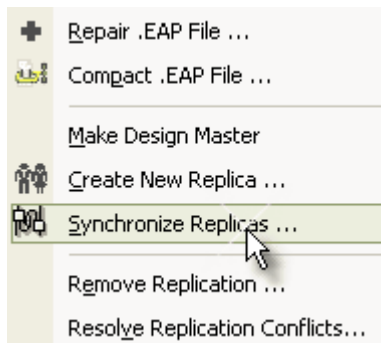
8.14.5.3 Synchronizing Replicas

To copy changes from one member of the replica set to another, you use the Synchronize function. Note that information will be copied both ways - including deletes, updates and inserts.

Synchronize a Replica

To synchronize a replica and a design master, follow these steps:

1. Open the Master project file.
2. Select **Synchronize Replicas** from the **Tools | Manage .EAP File** submenu to merge the open project and the selected replica back together.



Note: When you synchronize, both projects will end up containing identical information.

Change Collisions

Note that if two or more people work on the same element (package, diagram, etc.) then the replication engine has problems in resolving which change is the master. To avoid this, always work on separate areas in the model when you are using replicas. You may also use the **Resolve Replication Conflicts** option in the **Tools | Manage .EAP File** submenu.

See Also

- [Creating Replicas](#)
- [Design Masters](#)
- [Remove Replication](#)
- [Upgrading Replicas](#)
- [Resolving Conflicts](#)

8.14.5.4 Remove Replication

Replication makes many changes to the database structure of your model. As a consequence the model file becomes considerably larger with additional information. If you no longer require a model to be replicable, you can elect to remove all replication features.

Remove Replication

To remove replication, follow these steps:

1. If a Repository is not opened - it does not allow the menu option for removal of the replication (a temporary repository - not the one having replication removed) needs to open at the time. Ensure you have a repository opened at the time of creation.
2. From the *Tools | Manage .EAP File* submenu, select *Remove Replication*, to open the *Remove Replication Wizard*.
3. Enter the full path and file name of the project to have replication removed. Press *Next*.
4. Enter the full path and file name of the base EA model (with no replication) to act as template. Press *Next*.
5. Enter the full path and desired file name for the output file. Press *Next*.
6. Select whether you wish to have a log file created, and enter a file name for the log file.
7. Press *Run* to begin removing replication.
8. Enterprise Architect will create a new project containing all the model information.
9. Your model has now had replication removed, and should be considerably smaller.

See Also

- [Creating Replicas](#)
- [Design Masters](#)
- [Synchronizing Replicas](#)
- [Upgrading Replicas](#)
- [Resolving Conflicts](#)

8.14.5.5 Upgrading Replicas

With new releases of Enterprise Architect there may be changes to the underlying project structure - eg. more tables, changed queries, etc. If you are using replicas to share and work with EA projects, it is very important that you open the Design Master before opening any of the replicas with an updated version of EA.

Warning: Upgrading Replicas takes special care!

Changes to the database design in a replica set may ONLY be done to the Design Master. Next time the

replicas are synchronized with the Master, the design changes are propagated through to the replicas. Trying to update a replica first will at best do nothing, and at worst cause the update of the Master to fail.

One other strategy is to remove replication from a copy of the replica set, upgrade that project and convert it into a new Design Master from which new replicas are created.

See Also

- [Creating Replicas](#)
- [Design Masters](#)
- [Synchronizing Replicas](#)
- [Remove Replication](#)
- [Resolving Conflicts](#)

8.14.5.6 Resolving Conflicts

When two or more people have changed the same element between synchronization points, EA will have trouble resolving which change to accept and which to discard. A choice is made based on some rules within the JET replication manager, but the discarded changes are also stored so you may manually override that choice.

After synchronizing replicas, open the *Resolve Conflicts* dialog and determine if there were any conflicts. Select whether to accept each change or use one of the discarded changes instead.

| Table with Conflicts | Description |
|----------------------|-----------------|
| t_object | Object Details |
| t_diagramobjects | Diagram Objects |
| t_diagram | Diagrams |

| Row ID |
|---|
| {guid {908F86CE-4337-48E8-93D5-CA1AB54... |

| Field | Current Value | Conflicting Value |
|-------|---------------|-------------------|
| | | |

Recommendations for Resolving Conflicts

Enterprise Architect stores model information in database records. When two records have been modified in different ways by different users, they appear in this dialog. Normally it is not necessary or desirable to examine conflicts - since they represent relatively inconsequential pieces of information that may very easily be modified through the normal Enterprise Architect interface eg by moving a diagram element.

The only case in which this dialog box should be used is where there is substantial piece of information has been overridden by another user - and you want to retrieve

1. Click on an entry in the *Table with Conflicts* list which is likely to contain the lost information.
2. Click on each entry in *Conflicting Records* list.
3. When the lost information appears in the *Conflict Details* list, click on *Overwrite with Conflict* button.

See Also

- [Creating Replicas](#)
- [Design Masters](#)
- [Synchronizing Replicas](#)
- [Remove Replication](#)
- [Upgrading Replicas](#)

8.15 Baselines and Differences

Baselines

Enterprise Architect (Corporate Edition) provides a facility to "Baseline" (snapshot) a model branch at a particular point in time. Baselines are in XMI format and are stored within the model in compressed format. More than one baseline can be stored against a single EA package. Using Baselines, a snapshot can be taken of a model branch at a particular point in development for later comparison to the current package state.

Compare (Diff)

The Compare (diff) Utility (Professional and Corporate Editions) lets you explore what has changed within a model over time and how previous versions of a model branch differ to what is currently in the model. This utility allows you to compare a model branch in EA with a Baseline created using the Baseline functionality, a file on disk, created previously using the EA XMI export facility, or the current version controlled XMI file on disk as created when using Version Control in EA

For more information, see:

- [Baselines](#)
- [The Compare Utility](#)

8.15.1 Baselines

Baselines

Enterprise Architect (Corporate Edition) provides a facility to "Baseline" (snapshot) a model branch at a particular point in time. Baselines are in XMI format and are stored within the model in compressed format. More than one baseline can be stored against a single EA package.

Using Baselines, a snapshot can be taken of a model branch at a particular point in development for later comparison to the current package state. This is most useful for determining changes made to the model during development compared to some Baseline saved at a crucial point, for example the completion of a phase or version iteration.

Baselines are particularly useful during requirements management to check for changes, additions and deletions which have occurred since the start of the current work phase. Knowing how a model has changed is an important part of managing change and the overall development process.

Baselines are generally used in conjunction with the Compare utility (diff) which is also built into the Corporate and Professional versions of EA.

A typical scenario for using baselines would be to:

1. Create the base model branch to a sufficient point to create a Baseline (checkpoint). Create and store the Baseline as Version 1.
2. As work continues on development, managers and developers can check the current model branch against the baseline for important modifications, additions and deletions. The Compare (diff) tool can be invoked from the Baseline dialog to check the current model branch against the stored version.
3. As required, minor baselines can be created to check recent progress. these "temporary baselines" are useful for managing change when a lot of work is being done and it is important to only see what has changed in the last 24 hours for example.
4. At sign off or the move to a new version/phase, a major baseline can be created to capture the new state of the model. Minor baselines created earlier may be deleted if required to save space.

Limitations:

Currently diagrams are not differenced, only the elements and links between them - including all associated information, such as tests, resources, problems etc.

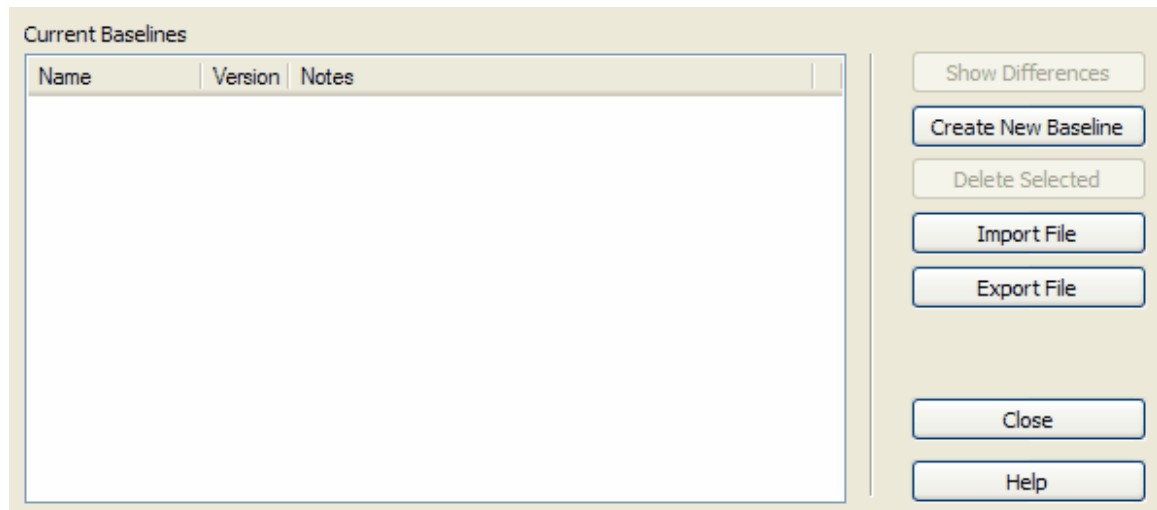
Large packages may take considerable time to fully difference.

See Also

- [Managing Baselines](#)
- [Creating Baselines](#)

8.15.1.1 Managing Baselines

To open the Manage Baselines dialog, right-click on the package at the head of the appropriate model branch and select the *Package Control | Manage Baselines* command.



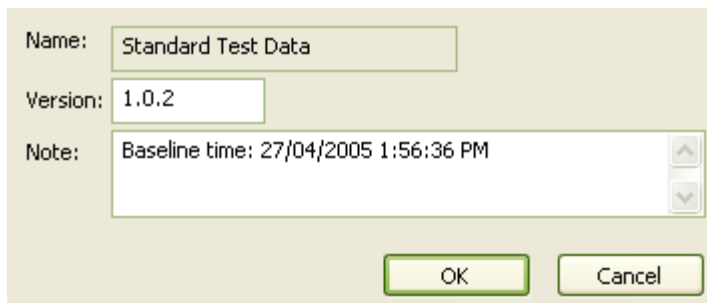
| Item | Functionality |
|---------------------|--|
| Current Baselines | The list of baselines for the current model branch. |
| Show Differences | Runs the compare utility on the selected baseline and the current model branch. |
| Create New Baseline | Creates a new baseline |
| Delete Selected | Deletes the selected baseline. |
| Import File | Import an XMI file from the file system as a baseline for this current model branch. |

| | |
|-------------|---|
| Export File | Use the export button to save the selected baseline to an XMI file. |
| Help | Opens the help file on this page. |
| Close | Closes the Manage Baselines dialog. |

8.15.1.2 Creating Baselines

The New Baseline dialog is opened by pressing *Create New Baseline* from the *Manage Baselines* dialog (see [Managing Baselines](#)).

Pressing *OK* will create a new baseline and return you to the Manage Baselines dialog.



| Item | Functionality |
|---------|--|
| Name | Fixed as the Package Name for the currently selected model branch. |
| Version | You are free to enter whatever you like in this field, but it must be unique for the current branch. |
| Note | By default shows current time and date, but you are free to enter anything in this field. |

8.15.2 The Compare Utility (Diff)

The Compare Utility (Diff)

Enterprise Architect (Corporate and Professional Editions) has a comprehensive and powerful differencing utility built in. This utility allows you to compare a model branch in EA with:

1. A Baseline created using the Baseline functionality
2. A File on disk, created previously using the EA XMI export facility (must be in EA format and have the same package as the root node)
3. The current version controlled XMI file on disk as created when using Version Control in EA

Compare (diff) lets you explore what has changed within a model over time and how previous versions of a model branch differ to what is currently in the model. It is even possible to do a full model compare by exporting all of Model A to XMI, then using Compare to File from within the current model (Model B).

Comparing and checking model development at various points in the process is an important aspect of managing change and development, keeping track of what is being modified and ensuring the development and design process is on track.

Access to the Compare Utility is available from:

1. The [Baseline Dialog](#) (from the Project Browser context menu, select Package Control | Manage Baselines).

2. From the Project Browser context menu, select Package Control | Compare to File.
3. From the Project Browser context menu, select Package Control | Compare with Version on Disk.

See [Example Comparison](#) for an example of the results of a model comparison.

Limitations

The following known limitations apply:

- Diagram compare is currently not supported.
- Differencing very large package structures can be slow.
- Currently no "Print" facility to export Compare results to file.

8.15.3 Example Comparison

The diagram below shows the result of a package comparison. You will notice the following:

- A new tab DiffMerge has been added next to the open diagram tabs.
- A hierarchy of model elements is displayed in the left-hand pane. It is clearly visible, from the Status column and from the triangular icon, which elements in the hierarchy have changed since the baseline.
- In the right-hand pane, a table of properties is displayed for the element currently selected in the left-hand pane. The values of the properties in the current model and in the baseline are displayed. If any properties have changed between the model and the baseline, the row is highlighted. This example shows that for the class named "AbstractFactory", the date modified, the code generation language and the version number have all changed since the baselined version.

| Model Elements | Status | Property | Model | Baseline |
|------------------|---------|----------------|---|---|
| Logical View | | Abstract | true | true |
| Data Model | | Alias | | |
| Logical Model | | Author | | |
| AbstractFactory | Changed | Date Created | 6/09/2003 4:02:31 PM | 6/09/2003 4:02:31 PM |
| CreateProductA() | Changed | Date Modified | 27/04/2005 4:27:17 PM | 27/04/2005 4:25:40 PM |
| CreateProductB() | Changed | Difficulty | 1 | 1 |
| Links | | Filename | | |
| AbstractProductA | | Language | C++ | <none> |
| AbstractProductB | | IsLeaf | false | false |
| Client | | IsRoot | false | false |
| ConcreteFactory1 | Changed | Keywords | | |
| ConcreteFactory2 | Changed | Multiplicity | | |
| ProductA1 | | Name | AbstractFactory | AbstractFactory |
| ProductA2 | | Notes | This class declares an interface for operations that create abstract product objects. | This class declares an interface for operations that create abstract product objects. |
| ProductB1 | | Parent Package | Logical Model | Logical Model |
| ProductB2 | | Persistence | | |
| Data Model | | Phase | 1.0 | 1.0 |
| Logical Model | | Scope | Public | Public |
| | | Status | Proposed | Proposed |
| | | Stereotype | | |
| | | Type | Class | Class |
| | | Version | 1.1 | 1.0 |

8.15.4 Merging Baselines

Topic Under Construction

8.16 Spell Checking

EA provides a powerful spell checking facility. This operates at the project level and allows you to quickly spell check an entire project.

See Also

- [Using the Spell Checker](#)
- [Correcting Words](#)
- [User Dictionaries](#)
- [Select Language](#)

8.16.1 Using the Spell Checker

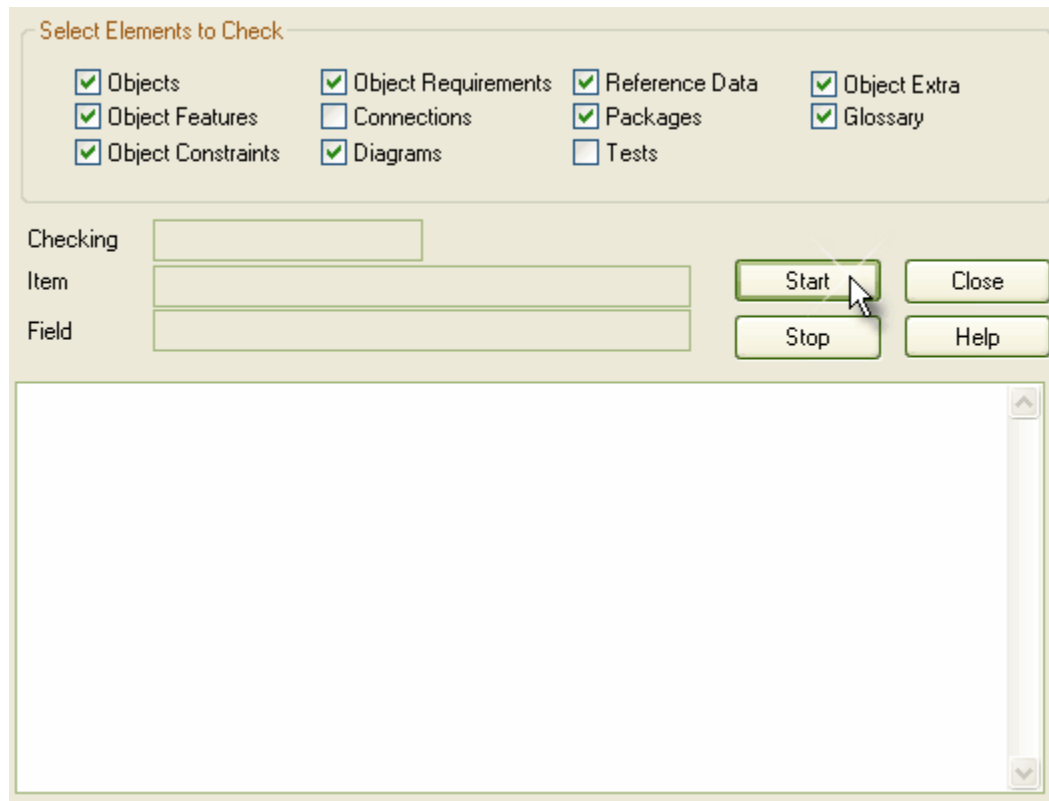
Enterprise Architect comes with an inbuilt spell checker.

Note: EA currently supports checking an entire model, and spell checking by single package. A future release will support more detailed spell checking at the element and diagram level.

To perform a spell check, follow these steps:

1. From the **Tools** menu, select **Spell Check Project** or **Spell Check Current Package**, depending on which level of spell check you require
2. The Spell Check dialog will open. Select all the items that you wish to spell check within your model.

Note: The **Spell Check Project** option allows you to check spelling for the entire model, whereas **Spell Check Current Package** only checks the package currently open, and does not allow you to check the options shown below.

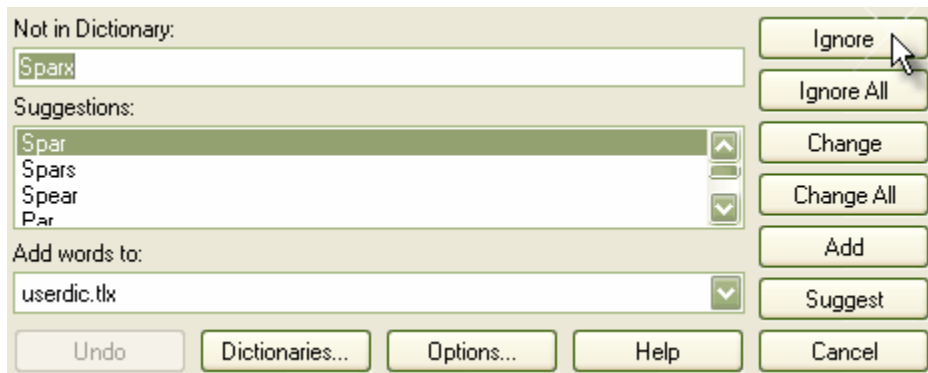


3. Press **Start** to begin the spell check.
4. As the spell check proceeds, you will see the text being checked in the visible edit area. If an error is detected, the word correction dialog will appear and offer you several [options](#) to correct the error.



8.16.2 Correcting Words

As the spell checking progresses, any errors or unknown words will be highlighted in the *Check Spelling* dialog. This allows you to correct the spelling of a word, ignore the error, add the word to a user dictionary, suggest alternatives - and otherwise assist in the spelling correction process.



To correct the current word you can:

- Modify the spelling by hand and press *Change* or *Change All*
- Accept the suggested alternative and press *Change* or *Change All*
- Press *Ignore* or *Ignore All* to ignore the word
- Press *Add* to add to the current user dictionary
- Press *Suggest* for alternatives
- Press *Cancel* to abort the spell check entirely

8.16.3 User Dictionaries

The inbuilt spell check stores user defined words in the *User Dictionary* (userdict.tlx) stored in the EA installation directory. During the spell check process, if you elect to add a word, it will be written into this file for later reference.

8.16.4 Select Language

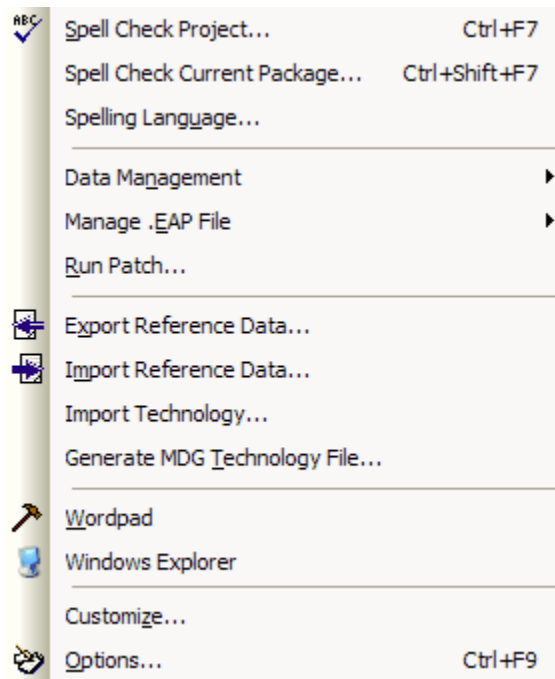
Enterprise Architect comes supplied with two dictionaries - US and British English. Additional dictionaries are available for download from the registered area of Sparx Systems. Once these have been downloaded and installed, you may select another language to perform the spell check in.

NOTE: Before selecting a language as described below, ensure you have **downloaded the additional language pack** and installed it in the EA install directory. Language packs are available from this page: http://www.sparxsystems.com.au/registered/reg_ea_down.html

Select a Different Language

To select another language:

1. From the *Tools* menu, select *Spelling Language*.

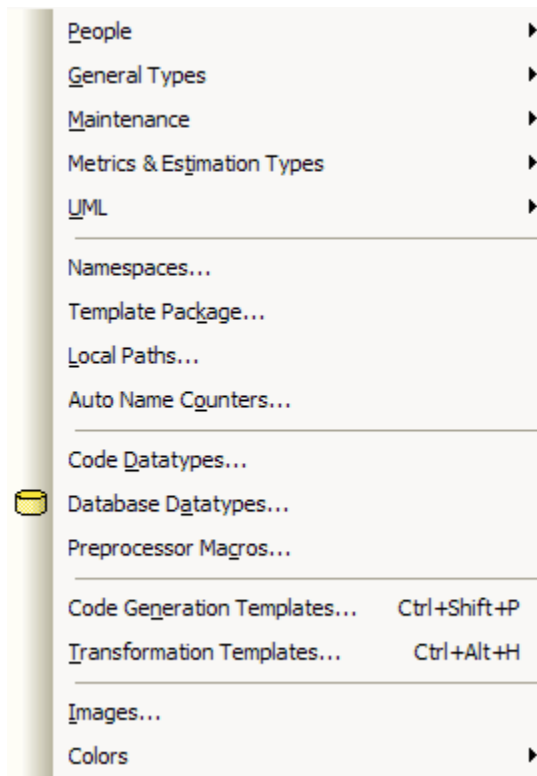


2. In the *Spell Check Language* dialog, select the required dictionary by checking the appropriate field.
3. Press *OK*. The selected language will remain the current language until changed.



8.17 Reference Data

Reference data is used in many places to provide content for drop down list boxes. Setting up a project often involves setting up the base set of reference types to use. Reference data options can be set up from the *Settings* menu.

**See Also**

- [People](#)
- [General Types](#)
- [Maintenance](#)
- [Metrics and Estimation](#)
- [UML](#)
- [Data Types](#)
- [Import and Export Reference Data](#)

8.17.1 People

You can control the following for your project from the *Settings* | *People* submenu:

- [Model Authors](#)
- [Clients](#)
- [Resources](#)
- [Roles](#)

8.17.1.1 Model Authors

The *Project Author(s)* dialog allows you to enter a list of project authors. This information is used to denote the author of specific elements.

To access this dialog, select *Model Authors* from the *Settings* | *People* submenu.

Set up one or more project authors

Name:

Role:

Notes:

Defined Authors

| Name | Description |
|-----------------|-----------------------------|
| Geoffrey Sparks | Analyst, Designer Architect |
| Paul Mathers | C++ Programmer |

The options that are available for this dialog includes:

| Element | Description |
|-----------------|--|
| Name | The name of the person registered as a Project Author. |
| Role | The role(s) they play in the project (eg. Designer, Analyst, Architect) This role may be defined by using the Roles function or created in this dialog |
| Notes | Additional notes. |
| Defined Authors | List of authors already defined. |

8.17.1.2 Clients

Clients are the eventual owners of the software system. Capture client details associated with the current model using the *Project Clients* dialog.

To access this dialog, select *Clients* from the *Settings* | *People* submenu.

Information on clients

Name: Organisation:

Role(s):

Phone 1: Phone 2:

Mobile: Fax:

Email:

Notes:

Defined Clients

| Name | Role(s) |
|--------------|------------------|
| Arthur Stoa | Project Manager |
| Joanna Stoa | Business Analyst |
| Margery Stoa | Assistant PM |

The options that are available for this dialog includes:

| Element | Description |
|-------------------------------|--|
| Name | The name of the person listed as a resource. The resource Name is available for use in the Maintenance Workspace . |
| Organisation | The name of the organisation that the resource is employed by |
| Role(s) | The role(s) they play in the project (eg. Designer, Analyst, Architect). |
| Phone 1, Phone 2, Mobile, Fax | Contact Phone Numbers for the resource |
| Email | Email Address for the resource. |
| Notes | Additional notes. |
| Defined Clients | Clients that have already been defined. |

8.17.1.3 Resources

Resources are model authors, analysts, programmers, architects etc. That is, anyone who may work on the system over time - either adding to the model -OR- programming and designing elements of the system outside EA. A variety of information about resources may be recorded using the **Resources** dialog.

To access this dialog, select **Resources** from the **Settings | People** submenu.

Resources

Name: Organisation:

Role(s):

Phone 1: Phone 2:

Mobile: Fax:

Email:

Notes:

Available Resources

| Name | Role(s) |
|-----------------|--|
| Elosie Norman | Developer |
| Geoffrey Sparks | Solution Architect, Lead Designer, Project Manager |
| Paul Mathers | Lead Developer, Technical Architect |

The options that are available for this dialog includes:

| Element | Description |
|-------------------------------|--|
| Name | The name of the person listed as a resource. The resource Name is available for use in Resource Management . |
| Organisation | The name of the organisation that the resource is employed by |
| Role | The role(s) they play in the project (eg. Designer, Analyst, Architect). |
| Phone 1, Phone 2, Mobile, Fax | Contact Phone Numbers for the resource |
| Email | Email Address for the resource. |
| Notes | Additional notes. |
| Available Resources | Resources that have already been defined. |

8.17.1.4 Roles

Resources (People) associated with a project may play one or more **Roles** in the analysis, design or implementation. Use the **Project Roles** dialog to set the role types that are captured within EA. Examples of project roles include: application analyst, architect, developer and project manager.

To access this dialog, select **Roles** from the **Settings | People** submenu.

The options that are available for this dialog includes:

| Element | Description |
|---------------|---|
| Role | The name of the role. The role is available for use with the Model Author . |
| Description | A short description of the role |
| Notes | Additional information related to the role. |
| Defined Roles | Roles that have been previously defined including predefined roles. |

8.17.2 General Types

You can control the following for your project from the *Settings | General Types* submenu:

- [Requirement types](#)
- [Status types](#)
- [Constraint types](#)
- [Scenario types](#)

8.17.2.1 Requirement Types

The *Requirement Types* dialog allows you to specify the generic set of requirement types that can be entered into the requirements sections of dialogs. This helps to maintain a single set of typed requirements.

To access this dialog, select *Requirements* from the *Settings | General Types* submenu.

| Requirement: | Description: | Weight: |
|--|---|---------|
| Display | System will display in a specified format | 1.1 |
| Information will be displayed in a particular way | | |
| Defined Requirement Types | | |
| <input type="button" value="New"/> <input type="button" value="Save"/> <input type="button" value="Delete"/> | | |
| Name | Description | Wei... |
| Display | System will display in a specified fo... | 1.1 |
| Functional | Functional Requirement | 1.2 |
| Performance | Performance based requirement | 1.0 |
| Printing | System printing requirement | 1.0 |
| Report | The system will reduce a report | 1.0 |
| Testing | Testing requirement | 1.6 |
| Validate | Validate a particular rule | 1.0 |
| <input type="button" value="Cancel"/> <input type="button" value="Help"/> | | |

8.17.2.2 Status Types

You can configure the basic list of *Status Types* used in EA. Note that not all dialogs use this list ... but most do.

To access this dialog, select *Status Types* from the *Settings* | *General Types* submenu.

The screenshot shows a dialog box for editing a status type. At the top, the 'Status' is set to 'Validated' and the 'Description' is 'Approved and Checked'. Below this, there is a 'Status Type Color' section with a color picker showing a light green color and a 'Restore Default' button. To the right is a 'Preview' window showing a rectangular box with a green vertical bar on the left side. At the bottom, there are buttons for 'Applies to ...', 'New', 'Save', 'Delete', 'Close', and 'Help'. Below the buttons is a table listing various status types and their descriptions.

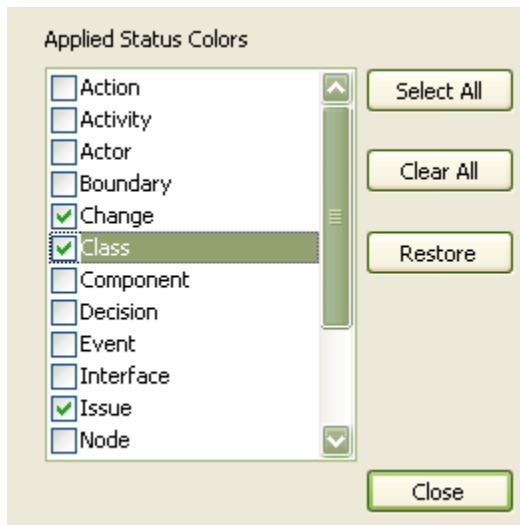
| Type | Description |
|-------------|--|
| Approved | Item is approved |
| Implemented | Finished |
| Invalidated | There is an issue that needs to be ad... |
| Mandatory | Required |
| New | Recently been identified |
| Proposed | Item has been proposed |
| Solved | Issue has been solved |
| Validated | Approved and Checked |

Status Type Color

It is possible to assign a color to each Status Type which will give a visual indication of the status of each diagram object. Select a named Status Type from the list then click on the color picker control to set the color for that status. Click **Save** to keep your changes.

Applying Colors to UML Elements

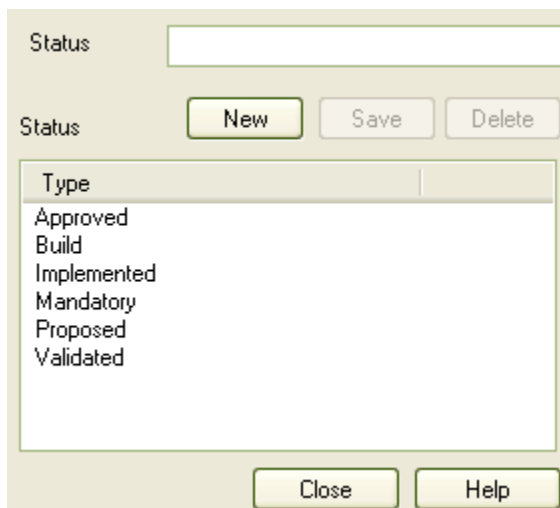
By default, status colors only apply to [Requirement, Issue and Change](#) elements. You may decide that you also want to apply these colors to other UML elements, such as Use Cases or Classes. To do this, simply click **Applies to...** and select the element types from the list.



8.17.2.3 Constraint Status Types

You can configure the basic list of *Constraint Status Types* used in EA. Set the status type.

To access this dialog, select *Constraint Status Types* from the *Settings | General Types* submenu.



8.17.2.4 Constraint Types

The *Constraint Types* dialog can be used for defining constraints. This will be picked up in a variety of places where constraints may fall into more categories than the basic Pre- Post- and Invariant conditions.

To access this dialog, select *Constraint Types* from the *Settings | General Types* submenu.

Set up general constraint types

Constraint:

Description:

Note:

Defined Constraint Types

| Name | Description |
|----------------|--------------------------------------|
| Assumption | An assumed condition |
| Invariant | A state the object must always be in |
| Post-condition | An ending state that must be met |
| Pre-condition | A starting state that must be met |
| Process | A process that must occur |

An example of an additional constraint type might be an *Assumption*.

8.17.2.5 Scenario Types

A drop down list of scenario types is available in the [scenario dialog](#). The standard types are Basic Path and Alternate Flow. Set additional scenario types using the *Scenario Types* dialog.

To access this dialog, select *Scenarios* from the *Settings | General Types* submenu.

| Scenario Type | Description | Weight |
|---------------|----------------------|--------|
| Alternate | Alternate pathway | 1.0 |
| Basic Path | Basic execution path | 1.0 |
| Simple | Standard scenario | 1.0 |

8.17.3 Maintenance

You can control the following for your project from the *Settings* | *Maintenance* submenu:

- [Problem types](#)
- [Testing types](#)

8.17.3.1 Problem Types

For the maintenance and change control screens, you can set the base *Problem Types* that are handled. Examples are hardware related issues, performance problems, software bugs and network problems.

To access this dialog, select *Problems* from the *Settings* | *Maintenance* submenu.

Problem Type: Description: Weight:

Defined Types

| Type | Description | Wei... |
|---------|---------------------|--------|
| HW | Hardware related | 1.00 |
| Network | Network problems | 1.00 |
| Perform | Performance | 1.50 |
| SW | Software | 2.00 |
| User | User caused problem | 1.01 |

Buttons: New, Save, Delete, Cancel, Help

8.17.3.2 Testing Types

Use the *Test Types* dialog to add testing types to the basic set that comes with EA. Typical test types are load tests, performance tests, function tests, etc.

To access this dialog, select *Testing* from the *Settings | Maintenance* submenu.

Test Type: Description: Weight:

Regression Regression Testing 1

Regression

Defined Types

| Name | Description | Wei... |
|------------|------------------------|--------|
| Load | Performance under load | 1.0 |
| Regression | Regression Testing | 1.0 |
| Standard | Simple Test procedure | 1.0 |

Buttons: New, Save, Delete, Close, Help

8.17.4 Metrics and Estimation

Risks, metrics, effort, TCF values, EFC values and Default Hour Rate for the project are controlled from the [Settings](#) | [Metrics and Estimation](#) submenu. For further information on these, see the [Project Management](#) section.

8.17.5 UML

You can control the following for your project from the [Settings](#) | [UML](#) submenu:

- [Stereotypes](#) - also see a list of [Standard Element Stereotypes](#)
- [Tagged Values](#)
- Cardinality

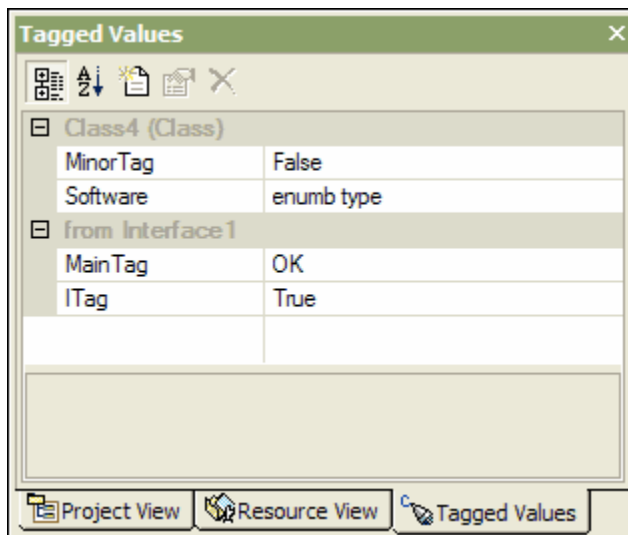
8.17.5.1 Standard Element Stereotypes

| Stereotype | Base Class | Type |
|------------------|-------------------|------------|
| «access» | Permission | Stereotype |
| «appliedProfile» | Package | Stereotype |
| association | Association | Constraint |
| «association» | AssociationEnd | Stereotype |
| «auxiliary» | Class | Stereotype |
| «become» | Flow | Stereotype |
| «call» | Usage | Stereotype |
| complete | Generalization | Constraint |
| «copy» | Flow | Stereotype |
| «create» | BehavioralFeature | Stereotype |
| «create» | CallEvent | Stereotype |
| «create» | Usage | Stereotype |
| «derive» | Abstraction | Stereotype |
| derived | ModelElement | Tag |
| «destroy» | BehavioralFeature | Stereotype |
| «destroy» | CallEvent | Stereotype |
| destroyed | Association | Constraint |
| destroyed | Association | Constraint |
| disjoint | Generalization | Constraint |
| «document» | Abstraction | Stereotype |
| documentation | Element | Tag |
| «executable» | Abstraction | Stereotype |
| «facade» | Package | Stereotype |
| «file» | Abstraction | Stereotype |
| «focus» | Class | Stereotype |
| «framework» | Package | Stereotype |
| «friend» | Permission | Stereotype |
| global | Association | Constraint |
| «global» | AssociationEnd | Stereotype |

8.17.5.2 Tagged Values

Tagged Values are used in a variety of places within EA to specify additional information about an element. Add default Tagged Value names using the *Tagged Values* dialog. These will appear in the drop list of Tagged Value names in the Tagged Value dialogs for elements, operations and attributes etc. For more information regarding the use of Tagged Values see the [Tagged Values Window](#) section.

To access this dialog, select *Tagged Values* from the *Settings | UML* submenu.



8.17.6 Data Types

Different programming languages support different inbuilt data types. The *Programming Languages Datatypes* dialog lets you extend and manage the set of inbuilt data types associated with a language as well as create new programming languages for use within EA.

To access this dialog, select *Code Datatypes* from the *Settings* menu.

Product Name:

Datatype:

Common Type:

Size

None Default: Max:

Length

Precision & Scale

Defined Datatypes for Programming Languages

| Product | Datatype | Size Unit | Default | Max |
|---------|----------|-----------|---------|-----|
| Java | short | | | |
| Java | long | | | |
| Java | int | | | |
| Java | float | | | |
| Java | double | | | |
| Java | char | | | |
| Java | byte | | | |
| Java | boolean | | | |

The options that are available for the Programming Languages Datatypes dialog includes:

| Element | Description |
|--------------|--|
| Product Name | The name of the programming language. |
| Add Product | The Add Product button allows the user to add a new programming language to the dropdown fields for class elements within the EA model and allows the new language to be made available to the Code template editor once at least one datatype has been added to the language. |
| Datatype | The name for the datatype, this will be the language specific name of the datatype. |
| Common Type | The common type is the generic name of the datatype , for example the Java boolean datatype has a common datatype Boolean. |
| New | Create a new data type. |
| Save | Saves the newly created datatype. |
| Delete | Deletes custom datatypes, the predefined datatypes may not be deleted. |

See Also

- [Code Template](#)
- [Class Attributes](#)
- [Class Operations](#)

8.17.7 Import and Export Reference Data

Reference data (including Glossary and Issue information) may be imported and exported to XML files for convenient updating of models.

Examples of where this may useful are:

- Copying glossaries from one model to another

- Adding additional stereotype profiles by merging new stereotypes into the model
- Updating reference data from files supplied by Sparx Systems as a maintenance release
- Copying resources, clients, etc. from one model to another

See Also

- [Export Reference Data](#)
- [Import Reference Data](#)

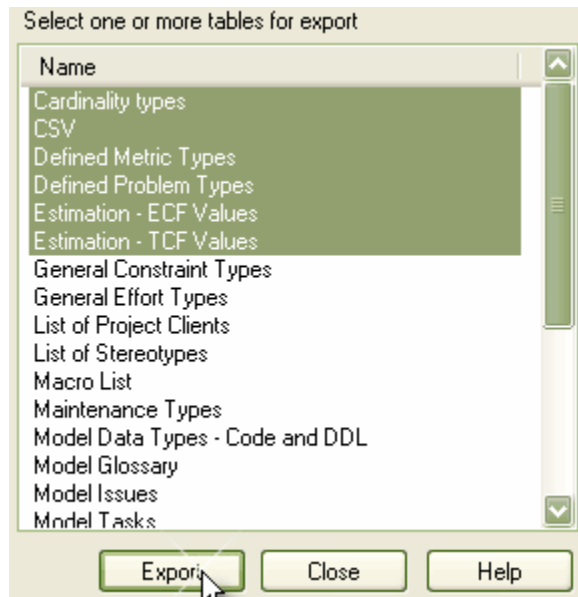
8.17.7.1 Export Reference Data

When reference and project data is exported, Enterprise Architect writes it out to a custom XML file. This includes table information, filter information, rows and columns.

Export Data

To export data, follow these steps:

1. From the *Tools* menu, select *Export Reference Data*.
2. In the *Data Exporter* dialog, select the table(s) you wish to export - you can select one or more tables for a single file.



3. Press *Export*.
4. Enter a valid file name with a .XML extension when prompted to do so.
5. This will export the data to the file. You can use any text or XML viewer to examine this file.

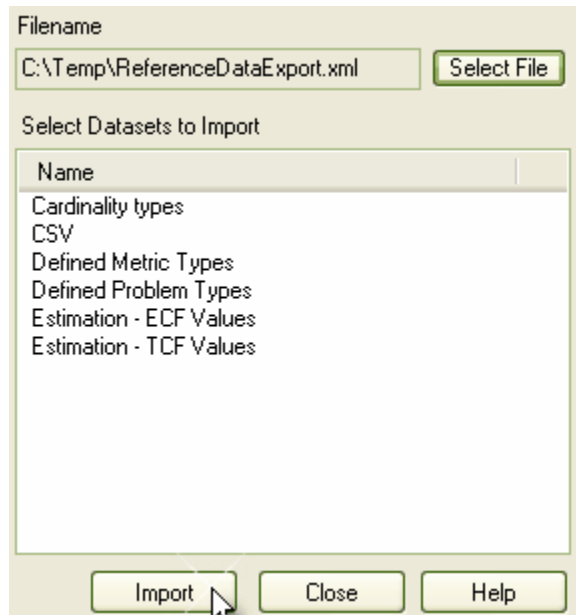
8.17.7.2 Import Reference Data

When you import data into Enterprise Architect, the system will merge the incoming data with the existing. If a record already exists it is updated to the new values - if not, a new record is added. Enterprise Architect never deletes records.

Import Data

To import data, follow these steps:

1. From the *Tools* menu, select *Import Reference Data*.
2. The *Import Reference* data dialog will appear.



3. Press *Select File* and select the filename to import data from - this must be an XML file produced by the EA [Data Exporter](#).
4. If you have entered the name of a valid file, a list of available tables to import will be displayed.
5. Select one or more of the tables to import.
6. Press *Import* to start the process - a message will pop up when the import is complete. Generally the process is quite fast.

Part

9

9 Project Management

Enterprise Architect provides some support for managing your project. Project managers may use Enterprise Architect to assign resources to elements, measure risk and effort and to estimate project size. Change control and maintenance are also supported (see [Maintenance Support](#)).

See Also

- [Estimation](#)
- [Resources](#)
- [Testing](#)
- [Life Cycle](#)
- [Changes and Defects](#)
- [Model Tasks List](#)
- [Project and Model Issues](#)
- [Model Glossary](#)
- [Update Package Status](#)
- [Manage Bookmarks](#)

9.1 Estimation

Metrics and Estimation

Project estimation is the task of working out how much time and effort is required to build and deploy a solution.

The use case metrics facility in EA provides a starting point for estimating project effort. Using this facility you can get a rough measure of the complexity of a system and some indication of the effort required to implement the model. Like all estimation techniques, this one requires some experience with previous projects to 'calibrate' the process.

There is additional information available on use case metrics at www.sparxsystems.com.au/UCMetrics.htm.

Calibrating

The following values will need careful calibration in order to gain the best possible estimates:

- [Technical Complexity Factors](#) are values that attempt to quantify the difficulty and complexity of the work in hand.
- [Environment Complexity Factors](#) are values that attempt to quantify non-technical complexities such as team experience and knowledge.
- [Default Hour Rate](#) sets the number of hours per **use case point**.

Estimating

Once all of the calibration values have been entered, the project timescale can be estimated. This is done through the [Use Case Metrics dialog](#).

9.1.1 TCF Dialog

Technical Complexity Factors are used in determining the Use Case Metrics estimation technique. You may add or modify these using the TCF Values dialog.

Open this dialog by selecting *TCF Values* from the *Settings | Metrics & Estimation Types* submenu.

Set up Technical factors for estimation

Factor Number: Description: Weight: Assigned Value:

TCF04 Complex internal processing 1.00 4.00

Defined Technical Types

| Type | Description | Wei... | Value | Ex Value |
|-------|--|--------|-------|----------|
| TCF01 | Distributed System | 2.00 | 5.00 | 10.00 |
| TCF02 | Response or throughput performa... | 1.00 | 4.00 | 4.00 |
| TCF03 | End user efficiency (online) | 1.00 | 2.00 | 2.00 |
| TCF04 | Complex internal processing | 1.00 | 4.00 | 4.00 |
| TCF05 | Code must be re-usable | 1.00 | 2.00 | 2.00 |
| TCF06 | Easy to install | 0.50 | 5.00 | 2.50 |
| TCF07 | Easy to use | 0.50 | 3.00 | 1.50 |
| TCF08 | Portable | 2.00 | 3.00 | 6.00 |
| TCF09 | Easy to change | 1.00 | 3.00 | 3.00 |
| TCF10 | Concurrent | 1.00 | 2.00 | 2.00 |
| TCF11 | Includ special security features | 1.00 | 2.00 | 2.00 |
| TCF12 | Provide direct access for third parti... | 1.00 | 5.00 | 5.00 |
| TCF13 | Special user training facilities are re... | 1.00 | 3.00 | 3.00 |

Unadjusted TCF: 47.00

Defined Technical Types

The editable list should contain all factors that could affect the technical complexity of the project environment.

These configured factors, whose summed Ex Values yield the Unadjusted TCF, works together with the ECF, to skew the overall complexity up or down depending on the level of technical complexity and the corresponding level of environmental support.

Weight

The TCF weight factor indicates how much technical complexity you assign to a factor - eg. "the system will be developed in ADA" may warrant a higher TWF than "the system will be a shell script". A weight evaluates its respective factor, but is irrelative to a project; the value field assesses each factor's role within a project. The supplied factors and their associated weights are defined by the Use Case Points Method, although they may be adjusted to suit a project's specific needs.

Value

For most purposes, the only table column needing adjustment will be 'Value', which indicates the degree of influence a particular factor holds over the project. As a suggested gage, a value of '0' indicates no influence, a '3' indicates average influence, and a '5' indicates strong influence.

9.1.2 ECF Dialog

Environment Complexity Factors are used in determining the Use Case Metrics estimation technique. You may add or modify these using the ECF Values dialog.

Open this dialog by selecting *ECF Values* from the *Settings | Metrics & Estimation Types* submenu.

Set up Environment factors for estimation

Factor Number: Description: Weight: Value:

ECF02 Application experience 0.50 3.00

Defined Environment Types

| Type | Description | Wei... | Value | Ex V... |
|-------|--|--------|-------|---------|
| ECF01 | Familiar with Rational Unified Process | 1.50 | 4.00 | 6.00 |
| ECF02 | Application experience | 0.50 | 3.00 | 1.50 |
| ECF03 | Object-oriented experience | 1.00 | 4.00 | 4.00 |
| ECF04 | Lead analyst capability | 0.50 | 4.00 | 2.00 |
| ECF05 | Motivation | 1.00 | 3.00 | 3.00 |
| ECF06 | Stable requirements | 2.00 | 4.00 | 8.00 |
| ECF07 | Part-time workers | -1.00 | 0.00 | 0.00 |
| ECF08 | Difficult programming language | -1.00 | 3.00 | -3.00 |

Unadjusted ECF: 21.50

Defined Environment Types

The editable list should contain all factors affecting the general design and development environment, including team experience and knowledge, team size, expertise and other non-functional environmental factors.

These configured factors, whose summed Ex Values yield the Unadjusted ECF, works together with the TCF, to skew the overall complexity up or down depending on the level of technical complexity and the corresponding level of environmental support.

Weight

A weight evaluates its respective factor's complexity in comparison to other factors, but is irrelative to a project; the value field assesses each factor's role within a project. The supplied factors and their associated weights are defined by the Use Case Points Method, although they may be adjusted to suit a project's specific needs.

Value

For most purposes, the only table column needing adjustment will be 'Value', which indicates the degree of influence a particular factor holds over the project. As a suggested gage, a value of '0' indicates no influence, a '3' indicates average influence, and a '5' indicates strong influence.

9.1.3 Estimating Project Size

Enterprise Architect uses a simple estimation technique based on the number of Use Cases to be built, the difficulty level of those Use Cases, some project environment factors and some build parameters. Once the parameters are set up and the Use Cases defined, open the *Use Case Metrics* dialog by:

- Navigating to the package of interest and selecting *Use Case Metrics* from the *Project* menu
- OR-
- Right clicking on the package of interest in the Project Browser and selecting *Package Metrics* from the *Documentation* submenu

Use Cases

Root Package: Use Case Model

Bookmarked: Phase like Keyword like

Reload All * Include Actors

| Package | Name | Type | Com... | Phz |
|------------------------|---------------------------|---------|----------|-----|
| UC01-1: User Manage... | Register with Book Shop | UseCase | 5.000... | 1.0 |
| UC01-1: User Manage... | Login | UseCase | 5.000... | 1.0 |
| UC01-2: Select Books | Request UnListed Book | UseCase | 5.000... | 1.0 |
| UC01-2: Select Books | Locate Book by Title o... | UseCase | 5.000... | 1.0 |
| UC01-2: Select Books | Browse Book Catalogue | UseCase | 10.00... | 1.0 |
| UC01-3: Manage Order | Pay for Order | UseCase | 5.000... | 1.0 |
| UC01-3: Manage Order | View Cart | UseCase | 5.000... | 1.0 |
| UC01-3: Manage Order | Remove Item from Cart | UseCase | 5.000... | 1.0 |
| UC01-3: Manage Order | Add title to Cart | UseCase | 5.000... | 1.0 |

Unadjusted Use Case Points (UUCP) = Sum of Complexity 60

Use Cases 11 Ave Hours per Use Case Easy: 42 Med: 85 Diff: 127

Technical Complexity Factor

Unadjusted TCF Value (UTV): 47.1
 TCF Weight Factor (TWF): 0.01
 TCF Constant (TC): 0.6
 TCF = TC + (TWF x UTV): 1.071

Environment Complexity Factor

Unadjusted ECF Value (UEV): 19.97
 ECF Weight Factor (EWF): -0.03
 ECF Constant (EC): 1.4
 ECF = EC + (EWF x UEV): 0.8009

Re-Calculate Help Report Close

Package Estimated

Use Case Points (UCP) = UUCP * TCF * ECF = 60 * 1.071 * 0.8009 = 51 UCP

Default Hours Estimated work effort (hours) = 10 * 51 = 510 hours

Note: This technique is of value only once you have a had couple of known projects to use as a baseline. Please, **DO NOT** use the provided "guesstimates" as a real world measure until you have some real world base lines to measure against.

| Element | Description |
|--------------|---|
| Root Package | The root package in the hierarchy. All use cases under here may potentially be included in the report. |
| Phase like | Include use cases with phase that matches the wildcard value in the field below (use * to match any characters, for example 1.* for 1.1 and 1.2 etc.) |
| Keyword like | Include use cases with keyword that matches the wildcard value in the field below (use * to match any characters). |

| | |
|---------------------------------|--|
| Technical Complexity Factor | This area lists parameters that describe the degree of technical complexity the project has. While the unadjusted TCF value comes from the TCF dialog, the other values compose the Use Case Points Method formula. These fields should be modified with caution. The final project estimate is directly proportional to the TCF. |
| Environmental Complexity Factor | This area lists parameters that calculate the degree of environmental complexity the project has, from factors such as programmer motivation or experience. The listed parameters compose the formula calculating the ECF, defined by the Use Case Points Method; the only parameter affected by the project is the unadjusted ECF value, derived from the ECF dialog. The final project estimate is directly proportional to the ECF. |
| UUCP | Unadjusted use case points = sum of use case complexity numbers. |
| Number of Use Cases | Total use case count in estimate. |
| Ave hours per use case | Averages the number of hours assigned to easy, medium and hard use cases - for information purposes only. |
| Package Estimate | Detailed breakdown of final figure - note that you will need to tailor the hours per use case point figure to the level that matches your type of project and capability based on known previous project outcomes. |
| Default Hours | Set the default hours that is fed into the final calculation. |
| Re-calculate | Re-run the estimate - usually after you change the hours/use case point number. |
| Reload | Re-run the search - usually after you change the filter criteria. |
| Report | Produce a rich text formatted report from the current estimate. |

9.1.4 Default Hours

Set the default hour rate per adjusted use case point by using the *Estimation Hour Rate* dialog. Open this dialog by:

- Pressing *Default Rate* on the *Use Case Metrics* dialog.

-OR-

- Selecting *Default Hour Rate* from the *Settings | Metrics & Estimation Types* submenu.

Set the project defaults for duration and hourly rate for the Use Case effort estimation

Duration:

Hourly Rate:

OK Cancel Help

Enter the default hour rate and duration and press **OK** to change current value.

Note: The value you enter will be stored as a local setting on your computer only.

Setting an hour rate is the most difficult factor in an accurate estimation. Typical ranges can vary from 10-30 hours per use case point. Studying the Use Case Points Method, from which this variable is defined, can help to understand its role in the estimation and facilitate selection of a suitable initial value. The best way to estimate this value is through the analysis of previous, completed projects. By calculating the project estimation on a completed project whose use cases and environment are configured within EA, the hour rate can be adjusted to render an appropriate value for your unique work environment.

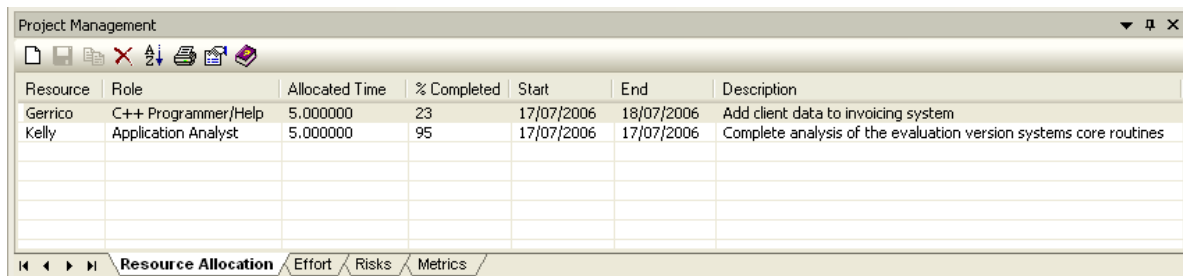
9.2 Resource Allocation

What is a Resource?

Resources are the people who work on a project. They can be assigned roles and allocated tasks, which allows for tracking of effort and estimation of time to complete.

Project Management Window

Resources are added, modified and deleted from the Project Management window. To access the **Project Management Window**, select **View | Project Management** from the main menu. [**Ctrl+Shift+7**]



| Resource | Role | Allocated Time | % Completed | Start | End | Description |
|----------|---------------------|----------------|-------------|------------|------------|---|
| Gerrico | C++ Programmer/Help | 5.000000 | 23 | 17/07/2006 | 18/07/2006 | Add client data to invoicing system |
| Kelly | Application Analyst | 5.000000 | 95 | 17/07/2006 | 17/07/2006 | Complete analysis of the evaluation version systems core routines |
| | | | | | | |
| | | | | | | |
| | | | | | | |

What to Do?

To find out more information about Project Management tasks, especially those involving :

- To allocate a resource to an element, see [Resource Management](#).
- To record additional project management information for an element, see:
 - [Effort Management](#) to record effort expended on the element.
 - [Risk Management](#) to record risk associated with the element.
 - [Metrics](#) to record metrics measured for an element.
- To obtain a report of resource allocation details, see [Resource Report](#).
- To configure Project Management data and populate the drop-down lists used on the Project Management dialogs, see:

- [Roles](#).
 - [System Clients](#).
 - [Effort Types](#).
 - [Metric Types](#).
 - [Risk Types](#).
- To find out about the functions of the Resource Allocation toolbar, see [The Project Management Window](#)

9.2.1 Resource Management

Enterprise Architect provides the ability to link a named resource in a named role to a given model element. This enables the project manager to track how far development of required components and classes has progressed (provided the programmers and others keep their figures up to date).

The screenshot shows the 'Project Management' dialog box with the 'Resource Allocation' tab selected. It features a table with columns for 'Resource' and 'Role'. The 'Resource' column lists 'Gerrico' and 'Kelly', with 'Gerrico' selected. The 'Role' column lists 'C++ Programmer/Help' and 'Application Analyst'. To the right of the table, there are several input fields: 'Resource:' (Gerrico), 'Role:' (C++ Programmer/Help), 'Completed (%)' (23), 'Allocated Time' (5.00), 'Expected Time' (5), and 'Time Expended' (2). Below these are 'Start Date' (17/07/2006) and 'End Date' (18/07/2006). A 'Description' field contains the text 'Add client data to invoicing system'. At the bottom, there are tabs for 'Resource Allocation', 'Effort', 'Risks', and 'Metrics'.

To enter Resource Allocation details for an element select the element and then choose the *Add new* option from the toolbar. This will present the following dialog which allows the user to enter the following data:

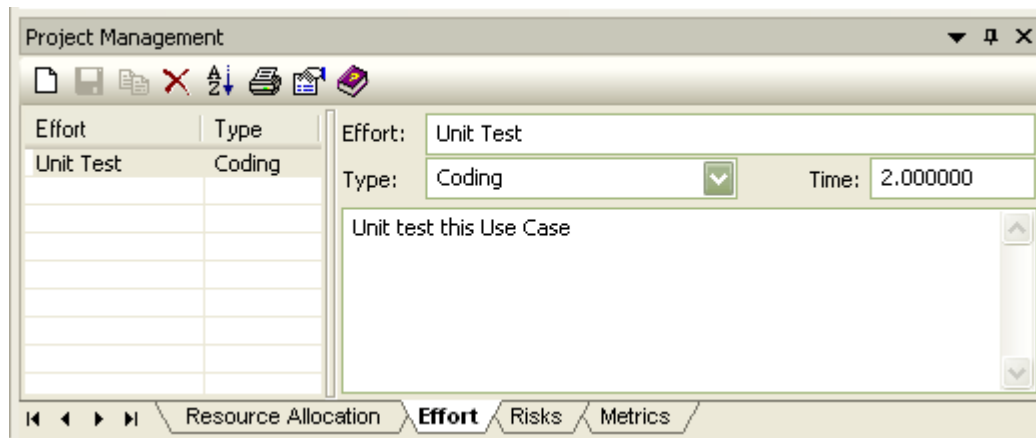
- The name of the *Resource*
- The *Role* of the resource
- The *Start* and *End* date for the resource
- The *Time* allocated to the resource
- The *% Done* the resource has completed
- The *Expected* time allocated to the resource
- The *Actual* time expended on the resource
- The *Description*
- The *History* the resource

To edit existing items, click the Show/Hide Properties icon on the toolbar.

9.2.2 Effort Management

To enter Effort details for an element select the element and then choose the *Add new* option from the toolbar. This will allow you to enter the following data:

- A name for the *Effort* (short description)
- The *Type* of effort
- The *Time* the effort will expend
- Some *Notes* on the effort

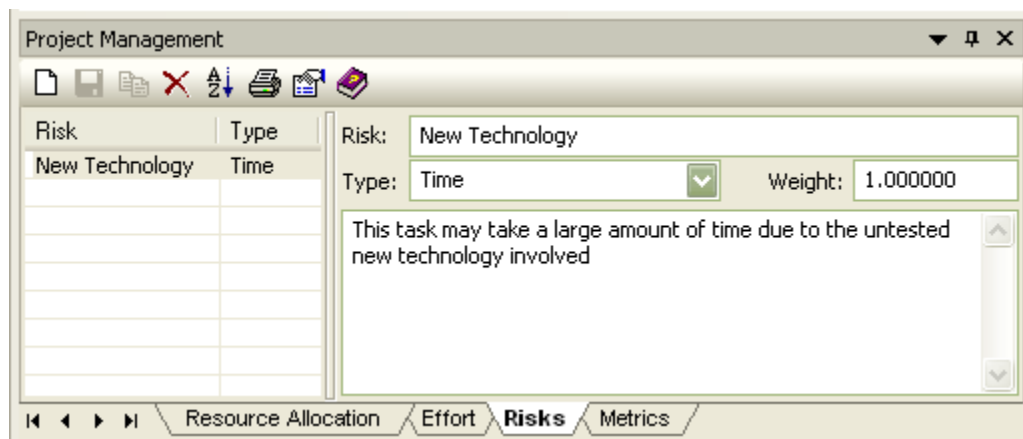


To edit an existing item, click the Show/Hide Properties icon on the toolbar.

9.2.3 Risk Management

To enter Risk details for an element select the element and then choose the *Add new* option from the toolbar. This will allow you to enter the following data:

- A name for the *Risk* (short description)
- The *Type* of risk
- A *Weight* for the risk
- Some *Notes* on the risk

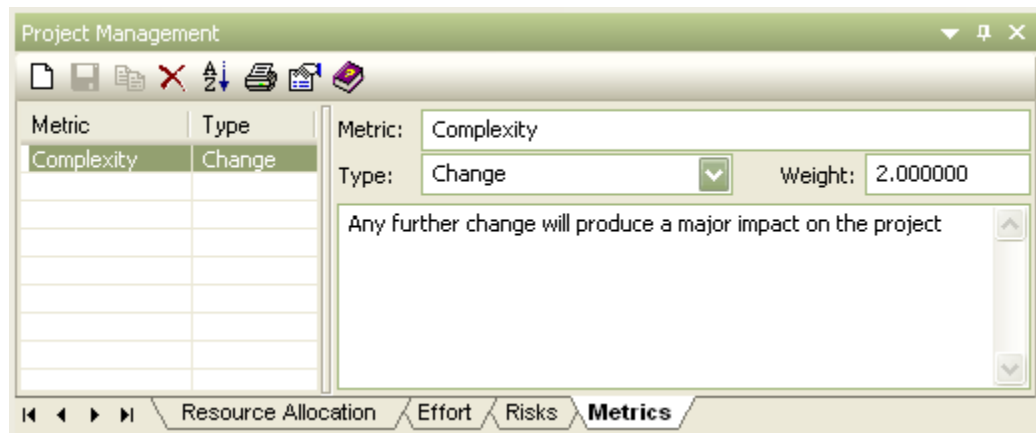


To edit an existing item, click the Show/Hide Properties icon on the toolbar.

9.2.4 Metrics

To enter Metrics for an element select the element and then choose the *Add new* option from the toolbar. This will allow you to enter the following data:

- A name for the *Metric* (short description)
- The *Type* of metric
- A *Weight* for the metric
- Some *Notes* on the metric



To edit an existing item, click the Show/Hide Properties icon on the toolbar.

9.2.5 Resource Report

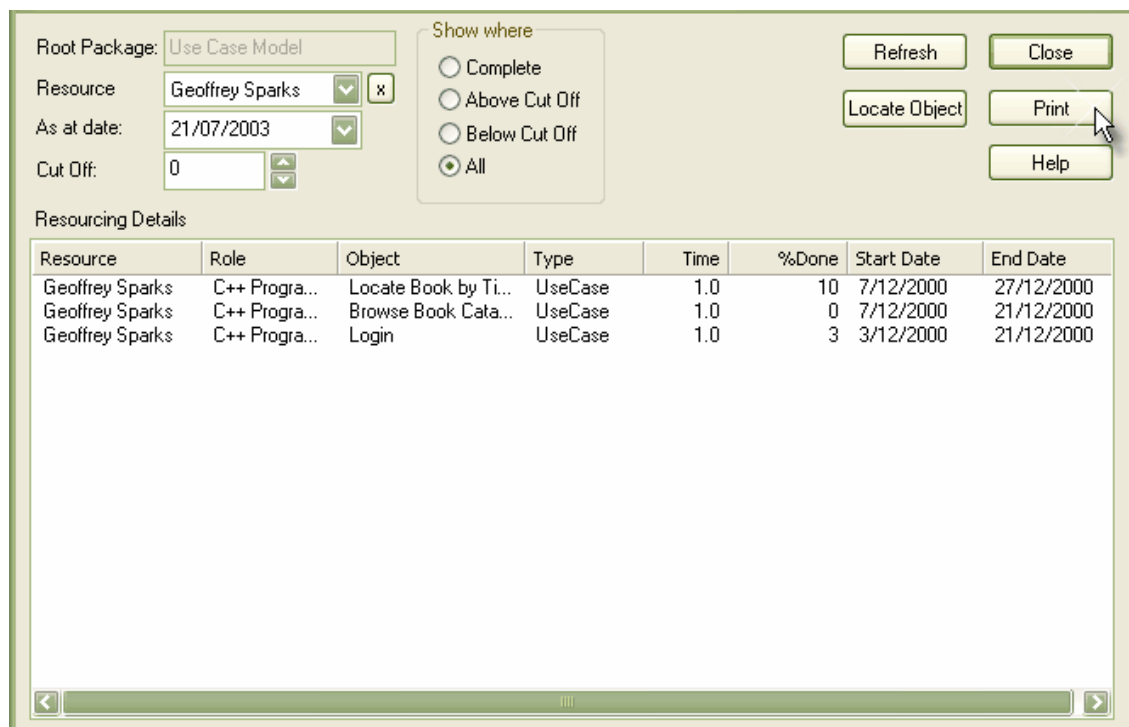
For a resource report on a package, do one of the following:

- In the Project Browser, right click on the package you require a report for. In the context menu, from the *Documentation* submenu, select *Resource Allocation*.

-OR-

- If the diagram currently active belongs to the package you require a report for, select *Resource and Tasking Details* from the *Project | Documentation* submenu.

The Resource Report dialog displays a list of all elements that have resources allocated to them. The result list includes the resource allocated, start and end dates, % complete and other relevant information. You may print out the results if required.



| Control | Description |
|------------------|--|
| Root Package | The name of the root package for which resourcing is being determined. |
| Resource | The (optional) name of a specific resource assigned to the project. |
| As At Date | Date to run the resource report for. |
| Show Where | Show resourcing where % complete is - Complete, Above the cut-off, below the cut-off, all. |
| Refresh | Refresh the form. |
| Locate Object | Find the selected element in the results list in the Project Browser (tree). |
| Print | Print a report. |
| %Complete | Set the limit to include or exclude resource details for - see "Show Where". |
| Resource Details | List of resources that meet search criteria. |
| Cancel | Close the New Project dialog without creating a new project. |

9.2.6 Roles

Project roles define the activities that resources may undertake with may be used with the Resource Management dialog in the *Project Management* window. Creating a role using this dialog creates a global list of roles which may be added to any element in the model.

To add a new role, follow the steps below:

1. From the *Configuration* | *People* submenu, select *Roles*, to open the *Project Roles* dialog.

Role: C++ Programmer Description: Programming in Visual C++

Defined Roles

| Type | Description |
|---------------------|---------------------------------------|
| Application Analyst | Define and model the application s... |
| Business Analyst | Model business processes |
| C++ Programmer | Programming in Visual C++ |
| Developer | Application development |
| Java Programmer | Java programming |
| Project Manager | Manage schedule |
| Solution Architect | Lead Technical and Project Archit... |
| Use Case Modeller | Use Case modelling |

Close Help

2. Enter the *Role* title, *Description* and optional *Note*.
3. Press *Save*.

9.2.7 System Clients

Enter names and additional information for system clients using the Project Clients dialog. This information is used in such places as the maintenance screens to indicate who reported a fault.

To add a new client, follow these steps:

1. From the *Configuration* | *People* submenu, select *Clients*. This will open the *Project Clients* dialog.

Information on clients

Name: Organisation:

Role(s):

Phone 1: Phone 2:

Mobile: Fax:

Email:

Notes:

New Save Delete

Defined Clients

| Name | Role(s) |
|----------------|------------------|
| Arthur Stoaat | Project Manager |
| Joanna Stoaat | Business Analyst |
| Margery Stoaat | Assistant PM |

Close Help

2. Fill in the appropriate client details.
3. Press *Save*.

9.2.8 Effort Types

You can specify the *Effort Types* used when assigning effort to an element in EA which may be used with the Effort dialog accessed from the *Project Management* window. Creating a Effort using this dialog creates a global list of Effort types which may be added to any element in the model. From the *Settings* | *Metrics &*

Estimation Types submenu, select *Effort*, to open the *Effort Types* dialog.

| Name | Description | Wei... |
|--------------|--------------------------------------|--------|
| Analysis | Analyzing System | 1.0 |
| Coding | Developing code | 1.0 |
| Construction | Design and build system compone... | 1.0 |
| Design | Designing specifications | 1.0 |
| Elaboration | Refine specification. Set up project | 1.0 |
| Transition | Implementation, acceptance testing | 1.0 |

The list of types will appear in the drop list for Effort Type. The weighting is a numeric value you may assign to this type for your own metrics.

Note: Although EA does not currently provide detailed reports on effort within a model, you can use the Automation Interface or similar tools to create your own custom reports based on effort information you enter.

9.2.9 Metric Types

You can specify the *Metric Types* used when assigning Metrics to an element in EA which may be used with the Metric dialog accessed from the *Project Management* window. Creating a Metric using this dialog creates a global list of Metric which may be added to any element in the model. From the *Settings | Metrics & Estimation Types* submenu, select *Metrics* to open the *Metric Types* dialog.

Metric Type: Description: Weight:

Change Change control, stability 1

Change requests,

Defined Metrics

New Save Delete

| Name | Description | Wei... |
|----------|-------------------------------------|--------|
| Breakage | Convergence, rework, software sc... | 1.0 |
| Change | Change control, stability | 1.0 |
| Cost | Budget, cost, expenditure | 1.0 |
| Progress | Iteration, planning, actuals | 1.0 |
| Team | Staffing, team dynamics | 1.0 |

Cancel Help

The list of types will appear in the drop list for Metric Type. The weighting is a numeric value you may assign to this type for your own metrics.

Note: Although EA does not currently provide detailed reports on Metrics within a model, you can use Automation Interface or similar tools to create your own custom reports based on Metric information you enter.

9.2.10 Risk Types

You can specify the *Risk Types* used when assigning Risk to an element in EA which may be used with the Risk dialog accessed from the *Project Management* window. Creating a risk type using this dialog creates a global list of risk types which may be added to any element in the model. From the *Settings | Metrics & Estimation Types* submenu, select *Risks*, to open the *Risk Types* dialog.

| | | |
|--|-------------------|---------|
| Risk Type: | Description: | Weight: |
| TIME | Schedule is tight | 1 |
| Comments | | |
| Defined Risks | | |
| <input type="button" value="New"/> <input type="button" value="Save"/> <input type="button" value="Delete"/> | | |
| Name | Description | Wei.. |
| TIME | Schedule is tight | 1.0 |
| <input type="button" value="Close"/> <input type="button" value="Help"/> | | |

The list of types will appear in the drop list for Risk Type. The weighting is a numeric value you may assign to this type for your own Risk.

Note: Although EA does not currently provide detailed reports on Risks within a model, you can use Automation Interface or similar tools to create your own custom reports based on Risk information you enter.

9.3 Testing

Introduction to Testing

Enterprise Architect allows you to create test scripts for model elements. Typically you would create unit tests for things that are being built, such as classes and components; integration tests to test how components work together; system tests to ensure the system meets business needs; acceptance tests to test user satisfaction; and scenario tests to test the end-to-end suitability and functionality of the application.

Basic Tasks

Simple tasks that you will want to do include:

- [Open the Testing Workspace](#)
- [Use the Test Details Dialog](#)

Categories

Tests come in the following categories:

- [Unit tests](#)
- [Integration tests](#)
- [System tests](#)
- [Acceptance tests](#)
- [Scenario tests](#)

Using Tests

Other things that you might want to do when working with tests include:

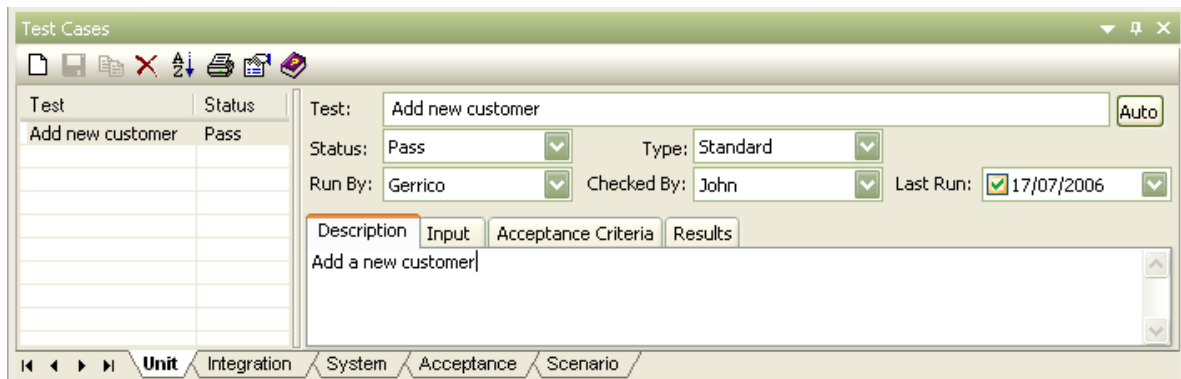
- [Import Scenario as Test](#)

- [Import Test from other elements](#)
- [Test Details Report](#)
- [Show Test Scripts in Compartments](#)
- [Create Test Documentation](#)

9.3.1 The Testing Workspace

The *Test Cases* window provides a quick and convenient method of working with element tests. When you select an element in a diagram or in the Project Browser, if the Testing Workspace is visible, the lists of tests for that element are loaded ready for modification or addition.

To open the Testing Workspace, select *Testing* from the *View* menu. Alternatively, you can use the keyboard shortcut *Alt+3*. This window may be docked to the application workspace.



Right click on the Test Cases window to access the context menu, to add or delete tests. To edit existing items, click the Show/Hide Properties icon on the toolbar.

There are five tabs along the base of the window - one for each of the following types of testing:

- [Unit testing](#)
- [Integration testing](#)
- [System testing](#)
- [Acceptance testing](#)
- [Scenario testing](#)

See Also:

- [Test Details dialog](#)

9.3.2 The Test Details Dialog

The *Test Details* dialog opens from the Test Cases window when you opt to add or modify test cases.

Tip: Add multiple test cases in one batch by using the *Apply*, *Save* and *New* options.

Test Details and Execution Status

Test: Type:

Description:

Input:

Acceptance Criteria:

Execution

Status: Last Run Date:

Run By: Checked By:

Results:

9.3.3 Unit Testing

Use Unit Testing to test classes, components and other elements as programmers build them.

The Unit Testing tab is displayed in the Test Cases window by default. Open the Test Cases window by selecting **Testing** from the **View** menu. Then open a diagram and select an element - all of the test scripts for that element will appear in the **Test Cases** window.

Test Cases

| Test | Status |
|------------------|--------|
| Add new customer | Pass |
| | |
| | |
| | |
| | |

Test:

Status: Type:

Run By: Checked By: Last Run:

Description Input Acceptance Criteria Results

Add a new customer

Unit Integration System Acceptance Scenario

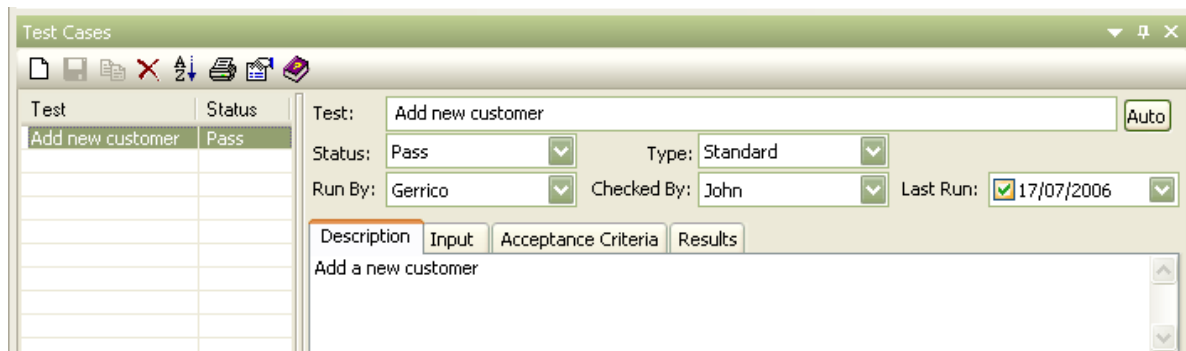
| Control | Description |
|-----------|--------------|
| Unit Test | Name of test |

| | |
|---------------------|--|
| Type | The type of test |
| Status | The current status of test (passed, failed...) |
| Last Run | The date test last run |
| Description | A description of the test |
| Run By | Test run by (person) |
| Checked By | Test run checked by (person) |
| Input | Input data |
| Acceptance Criteria | Acceptance conditions |
| Results | Results of last test |
| Defined Tests | List of defined tests associated with this element |

9.3.4 Integration Testing

Use Integration Testing to test how the constructed components work together.

To display the Integration Testing screen, open the Test Cases window by selecting *Testing* from the *View* menu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Integration* tab.



| Control | Description |
|---------------------|---|
| Integration Test | Name of test. |
| Type | The type of test. |
| Status | The current status of test (passed, failed...). |
| Last Run | The date test last run. |
| Description | A description of the test. |
| Run By | Test run by (person). |
| Checked By | Test run checked by (person). |
| Input | Input data. |
| Acceptance Criteria | Acceptance conditions. |
| Results | Results of last test. |
| Defined Tests | List of defined tests associated with this element. |

9.3.5 System Testing

Use System Testing to test the system performs the right business functions correctly.

To display the System Testing screen, open the Test Cases window by selecting *Testing* from the *View* menu.

Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *System* tab.

The screenshot shows the 'Test Cases' window. On the left is a table with two columns: 'Test' and 'Status'. The first row contains 'Add new customer' and 'Pass'. To the right of the table are several fields: 'Test:' with a text box containing 'Add new customer' and an 'Auto' button; 'Status:' with a dropdown menu set to 'Pass'; 'Type:' with a dropdown menu set to 'Standard'; 'Run By:' with a dropdown menu set to 'Gerrico'; 'Checked By:' with a dropdown menu set to 'John'; and 'Last Run:' with a date field set to '17/07/2006' and a checkmark. Below these fields are four tabs: 'Description', 'Input', 'Acceptance Criteria', and 'Results'. The 'Description' tab is active, showing the text 'Add a new customer'.

| Control | Description |
|---------------------|--|
| System Test | Name of test |
| Type | The type of test |
| Status | The current status of test (passed, failed...) |
| Last Run | The date test last run |
| Description | A description of the test |
| Run By | Test run by (person) |
| Checked By | Test run checked by (person) |
| Input | Input data |
| Acceptance Criteria | Acceptance conditions |
| Results | Results of last test |
| Defined Tests | List of defined tests associated with this element |

9.3.6 Acceptance Testing

Use Acceptance Testing to ensure users are satisfied with the system.

To display the Acceptance Testing screen, open the Test Cases window by selecting *Testing* from the *View* menu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Acceptance* tab.

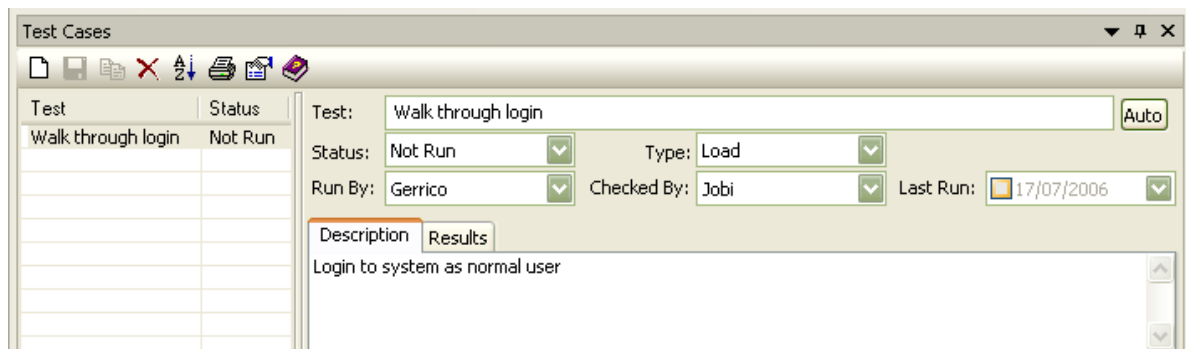
This screenshot is identical to the previous one, showing the 'Test Cases' window with the same table and fields. However, the 'Acceptance Criteria' tab is now selected and active, while the 'Description' tab is no longer active. The text 'Add a new customer' is still visible in the description field.

| Control | Description |
|---------------------|--|
| Unit Test | Name of test |
| Type | The type of test |
| Status | The current status of test (passed, failed...) |
| Last Run | The date test last run |
| Description | A description of the test |
| Run By | Test run by (person) |
| Checked By | Test run checked by (person) |
| Input | Input data |
| Acceptance Criteria | Acceptance conditions |
| Results | Results of last test |
| Defined Tests | List of defined tests associated with this element |

9.3.7 Scenario Testing

Use Scenario Testing to test the application with real-world situations and scenarios. An end-to-end test of all functions.

To display the Scenario Testing screen, open the Test Cases window by selecting *Testing* from the *View* menu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Scenario* tab.



| Control | Description |
|---------------|---|
| Unit Test | Name of test. |
| Type | The type of test. |
| Status | The current status of test (passed, failed...). |
| Last Run | The date test last run. |
| Description | A description of the test. |
| Run By | Test run by (person). |
| Checked By | Test run checked by (person). |
| Results | Results of last test. |
| Defined Tests | List of defined tests associated with this element. |

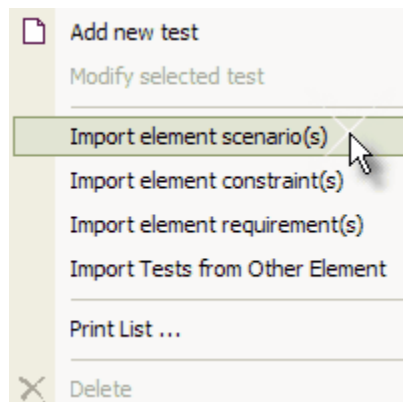
9.3.8 Import Scenario as Test

It is possible to import a Scenario from a Use Case or other element into the Test Scenarios list. This avoids having to duplicate the scenario information manually.

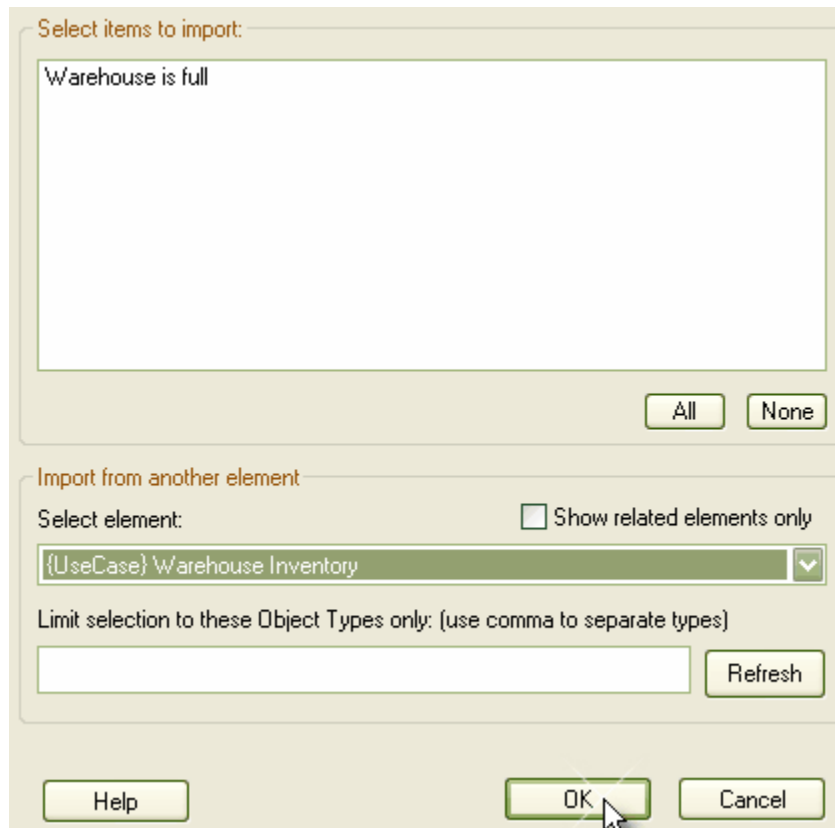
Import a Scenario

To import a scenario, follow the steps below:

1. Open the Test Scenario list - first open the *Test Cases* window by selecting *Testing* from the *View* menu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Scenario* tab.
2. Right click and select *Import Element Scenario(s)*. This will open the *Import Scenarios* dialog.



3. Select the scenarios to import in the list provided, from this dialog it is possible to import scenarios from any element in the model by using the Select Element drop down menu.



4. Press **OK** to import the selected scenario(s).

The Import Scenario Dialog has the following options:

| Control | Description |
|---------------------------|---|
| Select items to import | Selects the Scenario/s to import. |
| Select Element | Select any element from the model. |
| Limit Selection | Filter out specific element types. |
| Refresh | Refresh available options. |
| Show related objects only | Filters selection to apply only to related objects. |

9.3.9 Import Test from other elements

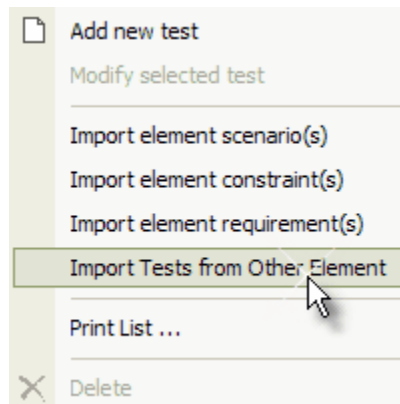
It is possible to import any test from a Use Case or other element into the Test Case. This avoids having to duplicate the test information manually.

Import a Test

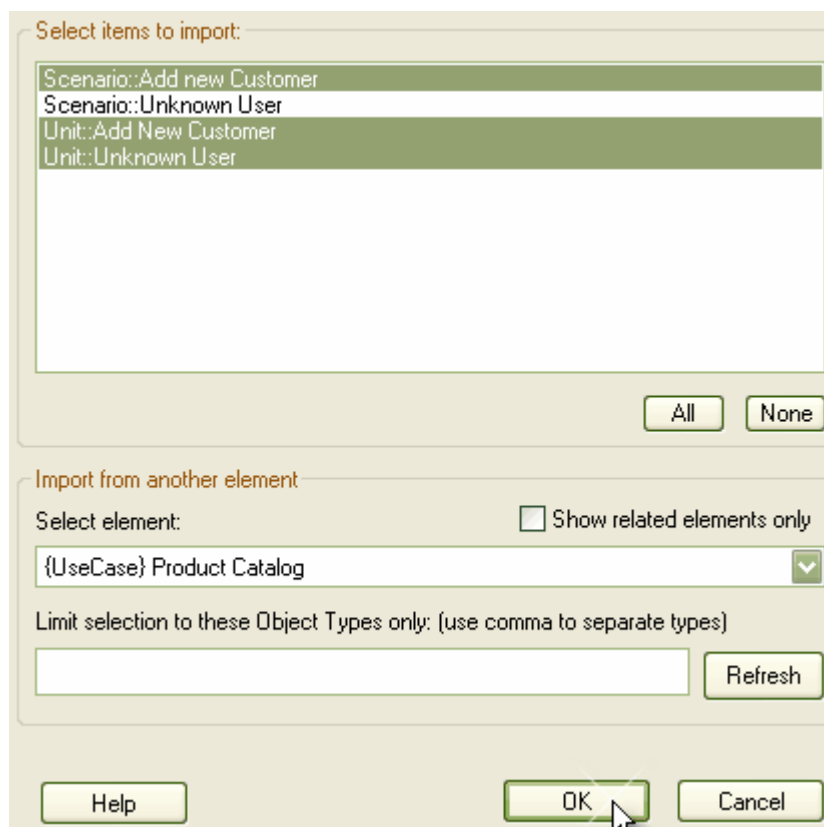
To import a test, follow the steps below:

1. Open the Test list - first open the **Test Cases** window by selecting **Testing** from the **View** menu. Then open a diagram and select an element - all of the test scripts for that element will appear in the **Test Cases** window.
2. Right click and select **Import Tests from Other Element**. This will open the **Import Element Tests**

dialog.



3. Select the scenarios to import in the list provided, from this dialog it is possible to import scenarios from any element in the model by using the Select Element drop down menu.



4. Press **OK** to import the selected test(s).

The Import Element Tests Dialog has the following options:

| Control | Description |
|---------|-------------|
|---------|-------------|

| | |
|---------------------------|---|
| Select items to import | Selects the Scenario/s to import. |
| Select Element | Select any element from the model. |
| Limit Selection | Filter out specific element types. |
| Refresh | Refresh available options. |
| Show related objects only | Filters selection to apply only to related objects. |

9.3.10 Test Details Report

You can view the *Test Details* report window for a package, which allows you to run filtered reports on all elements in the package hierarchy under your selection. You can also print the report details.

To access the *Test Details* window, select a package in the Project Browser, right click and select *Testing Details* from the *Documentation* submenu.

The screenshot shows the 'Test Details' report window. At the top, there are several controls: 'Root Package' set to 'UC01-1: User Management', 'Run By' and 'Checked By' dropdown menus, and 'Test Type' radio buttons for 'Unit' (selected), 'Integration', 'System', 'Acceptance', and 'All'. Below these are 'Static' radio buttons for 'Passed', 'Failed', 'Not Run', and 'All' (selected). There are also 'Locate Object', 'Refresh', 'Close', 'Print', and 'Help' buttons. The main area contains a table with the following data:

| Test | Type | Status | Run by | Checked by | Date Run |
|------------------|------|---------|---------------|-----------------|------------|
| Add new customer | Unit | Not Run | | | 23/07/2003 |
| Unknown User | Unit | Fail | Elosie Norman | Geoffrey Sparks | 23/07/2003 |

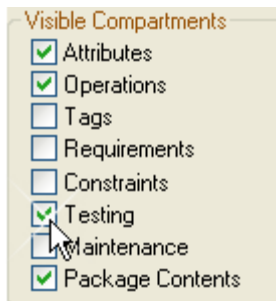
The test report includes the following fields:

| Control | Description |
|---------------|--|
| Run By | Filter for tests run by this person. |
| Checked By | Filter for tests checked by this person. |
| Test type | Select the test type desired. |
| Test result | Select the result to filter for. |
| Locate object | Locate an element in the result list in the Project Browser. |
| Print | Print a summary of the test results. |
| Refresh | Re-run the report query. |

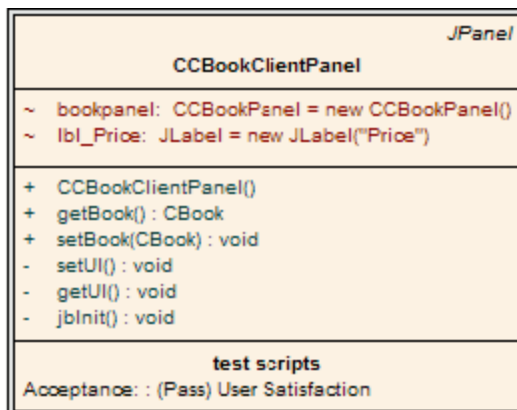
9.3.11 Show Test Scripts in Compartments

Any element that is capable of displaying a compartment can be used to show test scripts in a diagram. To make use of feature the element must have an attached test, to use this feature use the following instructions:

1. Open up a diagram with the element with the attached test item/s.
2. Double click the diagram background to bring up the Diagram Properties Dialog.



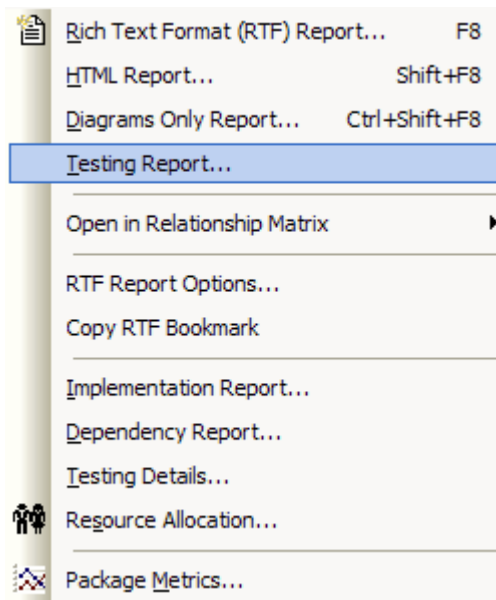
3. Check the *Testing* checkbox in the *Visible Compartments* panel.
4. The test/s will now appear as an item in the test scripts compartment of the diagram element.



9.3.12 Test Documentation

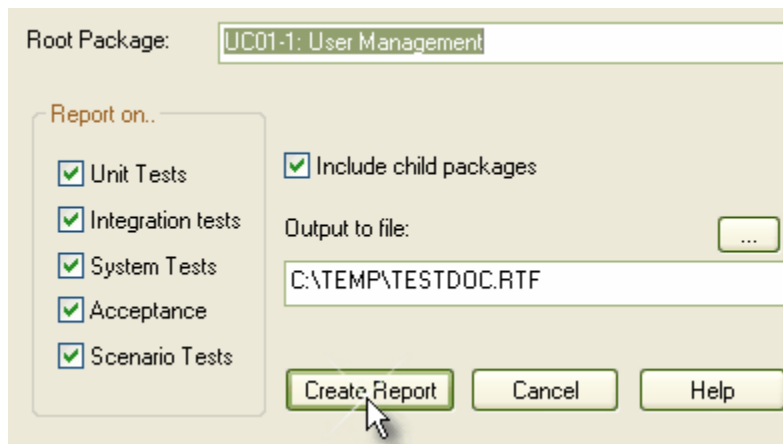
Enterprise Architect allows you to output in Rich Text format, the test scripts and results you have entered against elements in the model. For more information on entering test scripts and details see the rest of the [Testing](#) topic.

To create the documentation, access the *Create Test Documentation* dialog by right clicking on a package in the Project Browser window and selecting *Testing Documentation* from the *Documentation* submenu.



Note: You can also access the *Create Test Documentation* dialog by selecting *Testing Report* from the *Project / Documentation* submenu.

The *Create Test Documentation* dialog allows you to set up your report.



You can configure which tests to include or exclude in the report, whether to include child packages and what file to output to.

9.4 Maintenance

Maintenance Elements

The maintenance elements are defects, changes, issues and tasks. They all apply to individual model elements and may be used to record and capture problems, changes, issues and tasks as they arise and document the solution and associated details. They are defined as follows:

- A **defect** can be considered as a failure to meet a requirement for the current model element.
- A **change** can be considered as a change in requirement for the current model element.
- An **issue** is a record of a risk or other factor that might affect the project being recorded for the current model element.
- A **task** is a means of recording work in progress and work outstanding for the current model element.

Note that each of these maintenance elements applies at the model element level. For changes and issues that apply to the whole system, see the section [Changes and Defects](#); for tasks that apply to the whole system, see the section [Model Tasks](#).

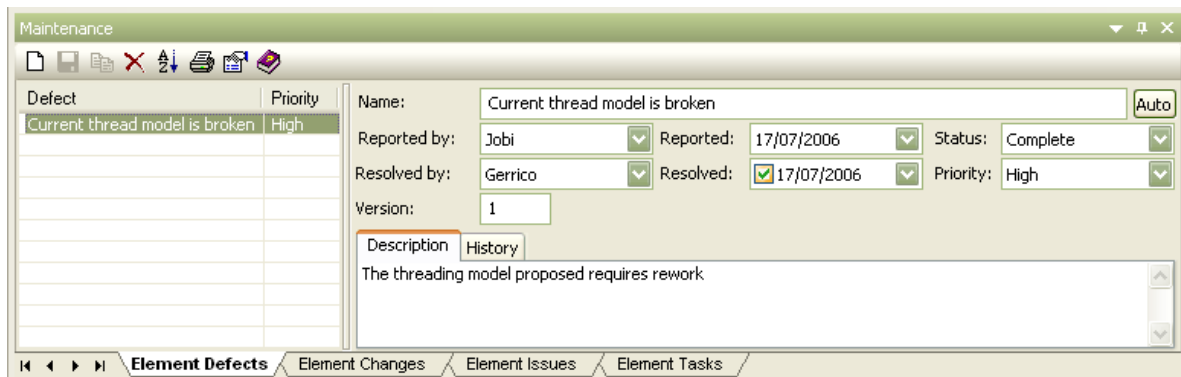
See Also

- [The Maintenance Workspace](#) shows how to create, modify, print and delete maintenance elements.
- [Show Maintenance Scripts in Compartments](#) shows how to display maintenance elements on diagrams.
- [Maintenance Element Properties](#) shows how to fill in the Properties dialog for the various maintenance elements.

9.4.1 The Maintenance Workspace

EA makes it easy to record and capture problems and issues as they arise and document the solution and associated details. The *Maintenance* window provides a quick method of viewing and modifying the list of defects, changes, issues and to do items associated with a particular model element. Access this window by selecting *Maintenance* from the *View* menu, or by pressing *Alt+4*.

Four tabs provide access to *Element Defects*, *Element Changes*, *Element Issues* and *Element Tasks* - click on the tab of interest then select model elements in diagrams or in the Project Browser to see the associated maintenance items. You can include defects, changes, issues and tasks in the main RTF documentation and HTML produced by EA. The RTF setup dialog has check boxes to show or hide element defects, changes, issues and tasks.

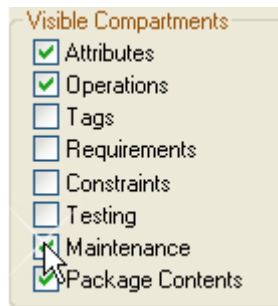


Using the toolbar, you are able to add or delete items and show/hide the properties window to allow editing of each item in the list.

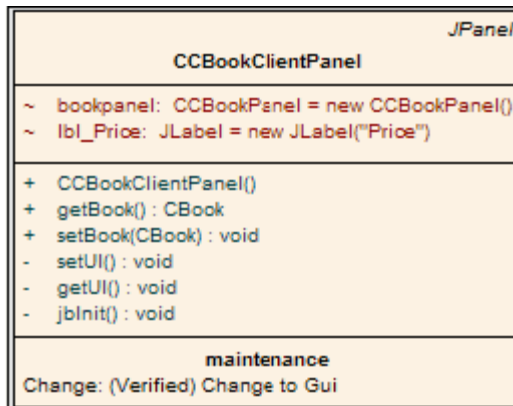
9.4.2 Show Maintenance Scripts in Compartments

Any element that is capable of displaying a compartment can be used to show maintenance scripts in a diagram. To make use of feature the element must have an attached maintenance item, to use this feature use the following instructions:

1. Open up a diagram with the element with the attached maintenance item/s.
2. Double click the diagram background to bring up the Diagram Properties Dialog.



3. Check the Show Maintenance checkbox in the Appearance Options panel.
4. The Maintenance Items will now appear as an item in the maintenance scripts compartment of the diagram element.



9.5 Changes and Defects

Change and Defect Elements

Changes and Defects are **structured comments** that can be used in managing change in a project. A **Defect** element (also known as an **Issue** element) corresponds to a failure to match the requirements for the current system. A **Change** element corresponds to a change in requirements for the current system.

Using Structured Comments

You can track changes and defects (issues) in an Enterprise Architect model. Change and Issue elements may be created in UML diagrams and linked using Realization, Dependency, Aggregation and other relationships to show what model element each affects and how each is resolved.

More Information

- Creating [Defects \(Issues\)](#)
- Creating [Changes](#)
- Change and Issue [Element Properties](#)
- [Assigning People to Defects or Changes](#)

9.5.1 Defects (Issues)

An Issue element is a structured comment which contains information about defects and issues that relate to the system/model. This corresponds in some sense to a failure to meet defined requirements for the current system. EA allows you to generate and handle Issues in much the same way as you can handle requirements (see the section on requirement tracking for more information).

You can link Issues using Realization connectors to model elements that are responsible for the Defect. You can even structure a hierarchy of Issues using aggregation.

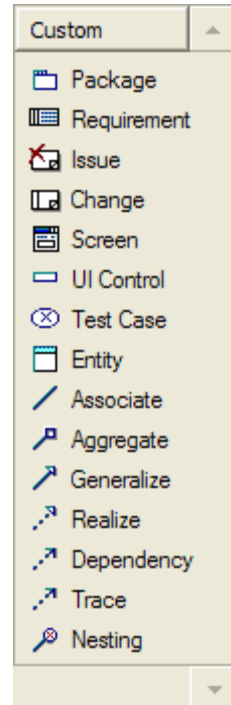
Add an Issue Using the UML Toolbox

To add an Issue to the model using the UML Toolbox (pictured right) you can either:

1. Open a Custom diagram.
2. Go to the *Custom* tab on the UML Toolbox.
3. Click on the *Issue* icon.
4. Create the issue by clicking on the diagram.
5. Enter the details as required.

-OR-

1. Drag the *Issue* icon to a target spot on the diagram.
2. Enter details as required.



Add an Issue Using the Insert New Element Dialog

To add an Issue to the model using the *Insert New Element* dialog (pictured below) follow the steps below:

1. Right click on a Package in the Project Browser.
2. Select *New Element* from the *Insert* submenu.
3. Select the 'Issue' from the *Type* drop down list.
4. Enter a *Name* for the element.
5. Press *OK*.

 A screenshot of the 'Insert New Element' dialog box. It has a 'Type' dropdown menu set to 'UseCase', a 'Name' text field with an 'Auto' button, and a 'Stereotype' dropdown menu. Below these are two checked checkboxes: 'Open Properties Dialog on Creation' and 'Close dialog on OK'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

9.5.2 Changes

A change element is a structured comment which contains information about changes that have been requested to the system/model. This corresponds in some sense to a change in requirements for the current system. EA allows you to generate and handle Changes in much the same way as you can handle requirements (see the section on requirement tracking for more information).

You can link Changes using Realization connectors to model elements that will implement the Change, and you can structure a hierarchy of changes using aggregation.

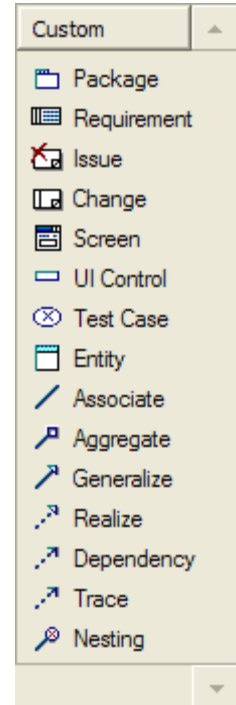
Add a Change Using the UML Toolbox

To add an Change to the model using the UML Toolbox (pictured right) you can either:

1. Open a Custom diagram.
2. Go to the *Custom* tab on the UML Toolbox.
3. Click on the *Change* icon.
4. Create the change by clicking on the diagram.
5. Enter the details as required.

-OR-

1. Drag the *Change* icon to a target spot on the diagram.
2. Enter details as required.



Add an Issue Using the Insert New Element Dialog

To add a Change to the model using the *Insert New Element* dialog (pictured below) follow the steps below:

1. Right click on a Package in the Project Browser.
2. Select *New Element* from the *Insert* submenu.
3. Select 'Change' from the *Type* drop down list.
4. Enter a *Name* for the element.
5. Press *OK*.

 A screenshot of the 'Insert New Element' dialog box. It has a light beige background. At the top, there are three fields: 'Type:' with a dropdown menu showing 'UseCase', 'Name:' with an empty text box and an 'Auto' button to its right, and 'Stereotype:' with a dropdown menu. Below these fields are two checked checkboxes: 'Open Properties Dialog on Creation' and 'Close dialog on OK'. At the bottom, there are three buttons: 'OK', 'Cancel', and 'Help'.

9.5.3 Element Properties

The property dialog for Changes and Issues is similar to that used by *Requirements*. It has a *Properties* tab containing the name of the Issue and relevant management details (owner, dates, etc). You can also [associate files](#) with the issue and [add tagged values](#).

The screenshot shows a 'Properties' dialog box with the following fields and values:

| | | |
|--------------------|----------------------------------|------------------------|
| Short Description: | Problem with password encryption | |
| Status: | Proposed | Type: Functional |
| Difficulty: | Medium | Phase: 1.0 |
| Priority: | Medium | Last Update: 1/11/2004 |
| Author: | John Redfern | Created: 1/11/2004 |
| Version: | 1.0 | |

Details:
When the user enters a password...]

9.5.4 Assign People to Defects or Changes

As an example of how you might use the relationship matrix to monitor issues or changes - the screen below illustrates staff (actors) being linked through Realization connectors to Issues. Each highlighted square indicates a responsibility that a staff member has to work on or correct a named issue.

This same approach can be used for any mix of model elements.

Source: \Work ... Type: <All> Link Type: Realisation
Target: UC01-1: User Management ... Type: <All> Direction: Source -> Target

| | | | | | | Collaborations | | | | | | | |
|-----------------|--|--|--|--|--|-------------------------|--|--|--|--|---|--|--|
| | | | | | | Customer | | | | | | | |
| | | | | | | Login Screen | | | | | | | |
| | | | | | | Login | | | | | | | |
| | | | | | | Register Screen | | | | | | | |
| | | | | | | Register with Book Shop | | | | | | | |
| | | | | | | secMan | | | | | | | |
| | | | | | | Security Manager | | | | | | | |
| | | | | | | SecurityManager | | | | | | | |
| Geoffrey Sparks | | | | | | | | | | | X | | |
| Paul Mathers | | | | | | | | | | | X | | |

9.6 Model Tasks List

The *Model Tasks* list is a convenient 'To Do' list of all major project work items that are not caught elsewhere. You may also track things like requests, meetings, etc.

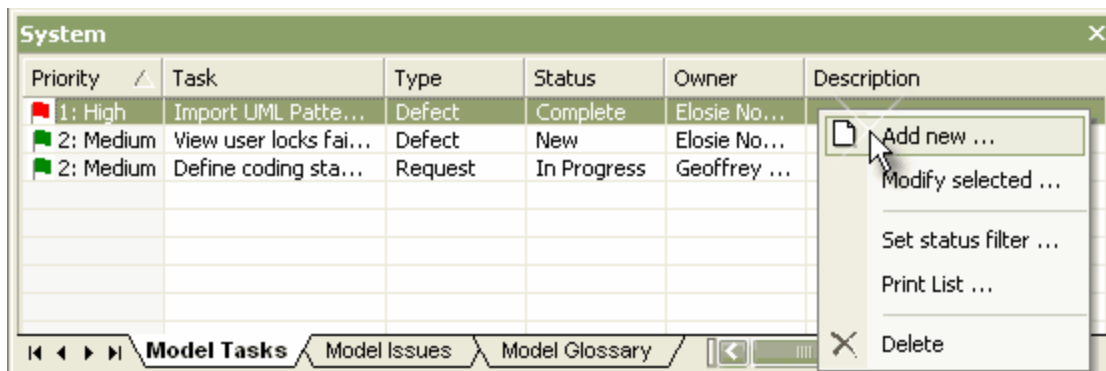
You can use the *Model Tasks Tab* to access your list, and *add, delete and edit list items* using the *Task Details* dialog.

9.6.1 Model Tasks Tab

Model Tasks

The *Model Tasks* list is a convenient 'To Do' list of major project work items. It can also be used to track things like requests or meetings.

The *Model Tasks* list is available as a tab on the *System* window. Open the *System* window by selecting *System* from the *View* menu. Alternatively, press *Alt+2*. Select the *Model Tasks* tab.



Right click on the list to view the context menu, which allows you to add, modify and delete list items, and to

set a status filter. You can also set the sort order by left clicking the title-bar of the column on which you wish to index the tasks.

For more information see the [Adding, Modifying and Deleting Tasks](#) topic.

Tip: Select the *Print List* menu option to print out the currently displayed items.

9.6.2 Adding, Modifying and Deleting Tasks

From the *Model Tasks* tab on the *System* window, use the *Task Detail* dialog to [Add](#), [Modify](#) and [Delete](#) tasks.

Add a Task

To add a task, follow the steps below:

1. Double click in the *Model Tasks* tab **-OR-** select *Add New* from the right click context menu. This will open the *Task Detail* dialog.

The screenshot shows the 'Task Detail' dialog box with the following fields and values:

- Task:** View user locks failed (with an 'Auto' button)
- Type:** Defect
- Owner:** Elosie Norman
- Start:** 24/07/2003
- Status:** New
- Assigned to:** (empty)
- End:** 24/07/2003
- Priority:** Medium
- Total Time:** (empty)
- Percent:** 0
- Phase:** (empty)
- Actual Time:** (empty)

Below the details are two empty text areas for 'Comments' and 'History'. At the bottom of the dialog are buttons for 'New', 'Save', 'OK', 'Cancel', and 'Help'.

2. Enter the details for the task - you can set the following:
 - The *Task* name
 - Press *Auto* if you have [auto counters](#) configured
 - The task *Type*
 - The task *Owner*
 - The expected *Start* and *End* date for the task
 - The current *Status* of the task
 - The person this task has been *Assigned to*

- The task *Priority* - high, medium or low
 - The expected *Total Time* for the task
 - The *Percent* complete
 - The *Phase* associated with this task
 - The *Actual Time* expended
3. Press *Save*.
 4. To create another entry, press *New*.
 5. To close, press *OK*.

Modify a Task

To modify a task, either:

1. Double click the task you want to modify in the list on the *Model Tasks* tab. This will open the *Task Detail* window for editing.

-OR-

1. Right click the entry you want to modify in the list on the *Model Tasks* tab. This will bring up the context menu.
2. Select the *Modify Selected* menu item. This will open the *Task Detail* window for editing.

Delete a Task

To delete a task, follow the steps below:

1. Right click the task you want to modify in the list on the *Model Task* tab. This will bring up the context menu.
2. Select the *Delete* menu item.

9.7 Maintenance Element Properties

Note: This page shows the properties dialog for defects. The dialogs for changes, issues and tasks differ only in minor details such as field names.

Element Defect details are recorded via the *Defect Details* dialog. Follow these steps to access this dialog:

1. Open the *Maintenance* window by selecting *Maintenance* from the *View* menu.
2. Open a diagram and select an element - all of the maintenance entries for that element will appear in the *Maintenance* window, under the various tabbed sections.
3. Ensure the *Element Defects* tab is selected.
4. Add a new item (select *Add new* from the right click context menu).
Alternatively, double click on an existing item or click the Show/Hide Properties toolbar icon.

The screenshot shows a 'Maintenance' dialog box with a table of defects and a detailed view of a specific defect. The table has columns for 'Defect' and 'Priority'. The detailed view includes fields for Name, Reported by, Resolved by, Version, Reported date, Resolved date, Status, and Priority. The description field contains the text: 'The threading model proposed requires rework'.

| Defect | Priority |
|--------------------------------|----------|
| Current thread model is broken | High |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Name: Current thread model is broken Auto

Reported by: Jobi Reported: 17/07/2006 Status: Complete

Resolved by: Gerrico Resolved: 17/07/2006 Priority: High

Version: 1

Description: The threading model proposed requires rework

History:

Element Defects | Element Changes | Element Issues | Element Tasks

The following fields are available:

| Control | Description |
|------------------|---|
| Name | Name of the defect. |
| Date | Date of maintenance. |
| Status | Indicate whether complete, approved etc. |
| Problem | Short description. |
| Notes | Long description. |
| Reported by | Who reported the fault. |
| Resolved by | Who fixed the fault. |
| Version number | The version number associated with this fix. |
| Date resolved | When fixed. |
| Resolver comment | Notes on the fix. |
| Problem List | List of problems and changes associated with element. |

9.8 Project and Model Issues

Any identified issues can be recorded against the current project. Issues are raised with a description, date, owner and status. You can also save a report on project issues in Rich Text Format.

See Also

- [Project Issues Dialog](#)
- [Model Issues Tab](#)
- [Add, Delete and Modify Issues](#)
- [Generate a Report](#)

9.8.1 Project Issues Dialog

The *Project Issues* dialog, accessed by selecting *Issues* tab in the System view. You can access the System view from the from the *Project* menu. This allows any identified issues to be recorded against the current project. Issues are raised with a description, date, owner and status. You can also save a report on project issues in Rich Text Format.

Tip: You can use the *Project Issues* dialog to [add, modify and delete issues](#). You can also [generate a report](#) of your issues.

Details

Issue:

Priority: Date:

Status: Owner:

Description:

Resolution:

Date: Resolved By:

Comments:

| Component | Description |
|-------------|--|
| Issue | The name of the issue. |
| Auto | Press Auto if you have auto counters configured. |
| Priority | The priority of this issue - low, medium or high. |
| Date | The date the issue arose. |
| Status | The issue's current status. |
| Owner | The person owning the issue. |
| Description | Description of the issue. |
| Resolution | Notes on the resolution of the issue. |
| Resolver | Person who resolved the issue. |
| Date | The date the issue was resolved. |
| Comments | Any comments regarding the resolution of the issue. |
| Close Issue | Click to close the issue. |
| New | Create a new issue. |
| Apply | Apply the currently shown issue. |
| Cancel | Cancel the currently shown issue. |

9.8.2 Model Issues Tab

The *Model Issues* tab in the *System* window allows any identified issues to be recorded against the current project. Issues are raised with a description, date, owner and status. You can also save a report on project issues in Rich Text Format.

Access this tab by opening the System window - select *System* from the *View |* menu or press *Alt+2*. Select the *Model Issues* tab.

Tip: You can right click on the list and select the *Print List* menu option to print out the currently displayed items.

| Priority | Issue | Date | Status | Owner | Description |
|-----------|-----------------------------|------------|-------------|--------------|---------------------------|
| 1: High | Test servers will be del... | 24/07/2003 | Closed | Elosie No... | The test server builds ha |
| 2: Medium | Public Holidays | 24/07/2003 | Open | Joanna S... | The schedule includes sta |
| 2: Medium | Compiler Version disparity | 24/07/2003 | Under Re... | Elosie No... | A number of the developi |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

You can use the *Issue Detail* dialog to [add](#), [modify](#) and [delete](#) issues. You can also [generate a report](#) of your issues.

9.8.3 Add, Delete and Modify Issues

Issues can be added, deleted and modified using either the [Project Issues dialog](#) or the [Model Issues tab](#).

9.8.3.1 Using the Project Issues Dialog

Select *Issues* from the *Project | Documentation* menu to open the *Issues* dialog.

Add an Issue

To add an issue, follow the steps below:

1. Enter the details for the issue - you can set the following:
 - *Issue* name: the name of the issue
 - Press *Auto* if you have [auto counters](#) configured
 - Issue *Priority* - high, medium or low
 - Issue *Date*: date arose
 - Current *Status*
 - Issue *Owner*: person owning issue
 - *Description* of the issue
 - *Resolution*: Text notes on resolution
 - *Resolved By*
 - *Date* resolved
 - *Close Issue*: click to close the issue
 - *Comments* when resolving the issue
2. Press *Save*.
3. To create another entry, press *New*.
4. To close, press *OK*.

Modify an Issue

To modify an issue, follow the steps below:

1. Select the issue you wish to modify in the issues list. This will show the issue's details.
2. Modify the details as required.
3. Press *Save*.

Delete an Issue

To delete an issue, follow the steps below:

1. Select the issue you wish to delete in the issues list.
2. Press *Delete*.

9.8.3.2 Using the Model Issues Tab

Open the System window by selecting *System* from the *View* menu. Alternatively, press *Alt+2*. Select the *Model Issues* tab.

Add an Issue

To add an issue, follow the steps below:

1. Double click in the *Model Issues* tab **-OR-** select *Add New* from the right click context menu. This will open the *Issue Detail* dialog.

The screenshot shows the 'Issue Detail' dialog box with the following fields and values:

- Issue:** Compiler Version disparity
- Priority:** Medium
- Date:** 24/07/2003
- Status:** Under Review
- Owner:** Elosie Norman
- Description:** A number of the developers have downloaded different version of a number the compilers. This has lead to unpredictable builds impacting on testing.
- Resolution Date:** 24/07/2003 (checked)
- Resolved By:** (empty)

Buttons at the bottom: Help, New, Save, OK, Cancel. A 'Close Issue' button is also present in the Resolution section.

2. Enter the details for the issue - you can set the following:
 - *Issue* name: the name of the issue
 - Press *Auto* if you have [auto counters](#) configured
 - Issue *Priority* - high, medium or low

- Issue *Date*: date arose
 - Current *Status*
 - Issue *Owner*: person owning issue
 - *Description* of the issue
 - *Resolution*: Text notes on resolution
 - *Resolved By*
 - *Date* resolved
 - *Close Issue*: click to close the issue
 - *Comments* when resolving the issue
3. Press *Save*.
 4. To create another entry, press *New*.
 5. To close, press *OK*.

Modify an Issue

To modify an issue, either:

1. Double click the issue you want to modify in the list on the *Model Issues* tab. This will open the *Issue Detail* window for editing.

-OR-

1. Right click the entry you want to modify in the list on the *Model Issues* tab. This will bring up the context menu.
2. Select the *Modify Selected* menu item. This will open the *Issue Detail* window for editing.

Delete an Issue

To delete an issue, follow the steps below:

1. Right click the issue you want to modify in the list on the *Model Issue* tab. This will bring up the context menu.
2. Select the *Delete* menu item.

9.8.4 Generate a Report

An RTF report of your issue list can be generated and viewed using either the [Project Issues dialog](#) or the [Model Issues tab](#).

Tip: You can view sample report output in the [Report Output Sample](#) topic.

9.8.4.1 Reports - Using the Project Issues Dialog

To generate an RTF document of your issue log using the *Project Issues* dialog, follow the steps below:

1. From the *Project | Documentation* menu, and select *Issues*, to bring up the issues dialog.
2. Press *Report*.
3. The *Save As* dialog will appear - select where to save your report and enter a name for it.

4. Press **Save**.
5. To view the report, press **View RTF**.

Tip: You can view sample report output in the [Report Output Sample](#) topic.

9.8.4.2 Reports - Using the Model Issues Tab

To generate an RTF document of your issue log using the **Model Issues** tab, follow the steps below:

1. Open the System window by selecting **System** from the **View** menu. Alternatively, press **Alt+2**.
2. Select the **Model Issues** tab.
3. Right click in the white area of the **Model Issues** tab and select **Create RTF Report** from the context menu.
4. Enter the file name and location to save your report to and press **Save**.
5. EA will generate your report. This should only take a few moments to complete.

Tip: You can view sample report output in the [Report Output Sample](#) topic.

9.8.4.3 Report Output Sample

An example of the output from a Issues report can be seen below:

List of Project Issues: 24-Jul-2003 9:47:00 AM

| Issue | Date/Owner | Description | Resolution |
|------------------------------|--------------------------|---|--|
| Test servers will be delayed | 24/07/2003 Elosie Norman | The test server builds have been delayed because the particular (unusual) memory requirements to match the customer's site are not available on shore. They are being sourced from Singapore but it will delay the builds and delivery of the machines. | Closed: 24/07/2003 Geoffrey Sparks The machines will be built and delivered using standard memory and the proprietary memory will be added later. All performance tests will be delayed until the memory is available. |
| Public Holidays | 24/07/2003 Joanna Stoa | The schedule includes staff working on public holidays. A number of staff have indicated that contrary to what they stated earlier they will not be available. | Open: 24/07/2003 |
| Compiler Version disparity | 24/07/2003 Elosie Norman | A number of the developers have downloaded different version of a number the compilers. This has lead to unpredictable builds impacting on testing. | Under Review: 24/07/2003 |

9.9 Model Glossary

The glossary allows you to set up a list of defined terms for your project. You may further divide the items by category - for example, Business terms, Technical terms. The glossary can be saved in Rich Text format for inclusion as part of a larger project document.

You can access the Model Glossary through the [Glossary Dialog](#) or through the [Model Glossary tab](#) on the *System* window.

Tip: Include a [Glossary Report](#) in your project requirements or functional specifications document(s).

9.9.1 The Glossary Dialog

To open the *Glossary* dialog, select Glossary from the *Project* menu. Use this dialog to [add](#), [modify](#) and [delete](#) glossary entries. You may also [limit the display](#) to show only technical or business related entries.

| Type | Term |
|-----------|----------------------|
| Business | Accounting Periods |
| Technical | Association |
| Technical | Component Model |
| Business | Customer |
| Technical | Deployment Model |
| Technical | Extends Relationship |

| Control | Description |
|------------------|---|
| Term | A term in the glossary. |
| Term type | Technical or business. |
| Description | Notes to describe the term. |
| Limit Display To | Filter glossary to Technical or Functional terms or both. |
| Report | Print a glossary report. |
| Defined Terms | List of defined glossary terms. |

9.9.2 Model Glossary Tab

The *Model Glossary* tab in the *System* window shows all of the items in your model's glossary. This tab lists all the defined technical and business terms already defined for a model. You can add to the list, delete or change items and filter the list to exclude by type.

Access this tab by opening the System window - select *System* from the *View* menu or press **Alt+2**. Select the *Model Glossary* tab.

Tip: You can right click on the list and select the *Print List* menu option to print out the currently displayed items.

| Term | Type | Meaning |
|----------------------|-----------|--|
| Accounting Periods | Business | A defined period of time whereby performance reports may be extracted. (norm |
| Association | Technical | A relationship between two or more entities. Implies a connection of some type |
| Component Model | Technical | The component model provides a detailed view of the various hardware and sof |
| Customer | Business | A person or a company that requests An entity to transport goods on their beha |
| Deployment Model | Technical | A model of the system as it will be physically deployed |
| Extends Relationship | Technical | A relationship between two use cases in which one use case 'extends' the behav |

You can use the *Glossary Detail* dialog to [add](#), [modify](#) and [delete](#) glossary entries. This can also be done from the [Glossary dialog](#).

Tip: Include a [Glossary Report](#) in your project requirements or functional specifications document(s).

9.9.3 Add, Delete and Modify Glossary Entries

Glossary entries can be added, deleted and modified using either the [Glossary dialog](#) or the [Model Glossary tab](#).

9.9.3.1 Using the Glossary Dialog

Select *Glossary* from the *Project* menu to open the *Glossary* dialog.

Add a Glossary Entry

To add an entry to the glossary, follow the steps below:

1. Enter the details for the glossary item - the *Term*, the *Type* and the *Meaning*.
2. Click *Save*.
3. To enter another item, click *New*.

Modify a Glossary Entry

To modify a glossary entry, follow the steps below:

1. Select the entry you wish to modify. Its details will appear in the fields in the top half of the window.
2. Change the details as required.
3. Press *Save*.

Delete a Glossary Entry

To delete a glossary entry, follow the steps below:

1. Select the entry you wish to modify. Its details will appear in the fields in the top half of the window.
2. Press *Delete*.

Limit the Display

You can alter which entry categories are displayed in the list. You can choose from:

- View all glossary entries - select the *All* option
- View Technical categorized entries only - select the *Technical* option
- View Business categorized entries only - select the *Business* option

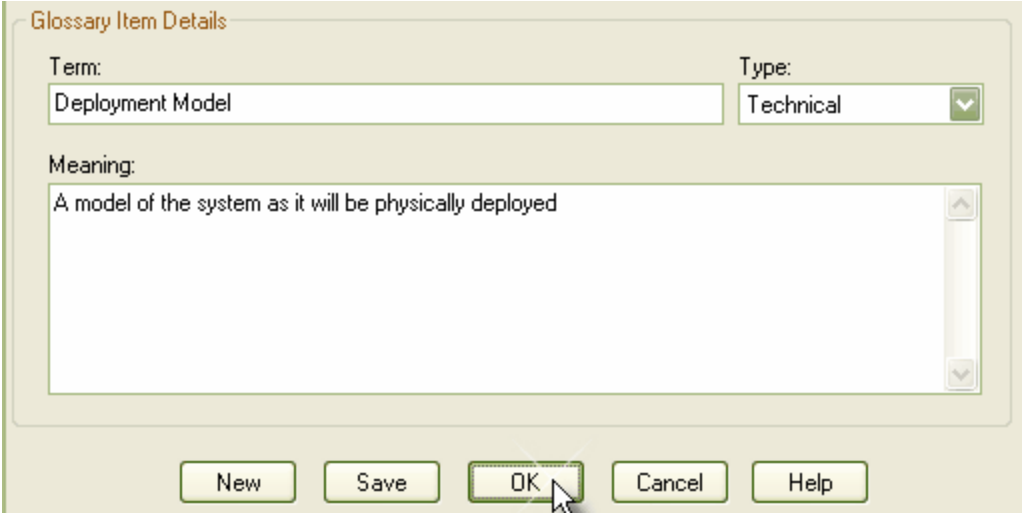
9.9.3.2 Using the Model Glossary Tab

Open the System window by selecting *System* from the *View | Other Windows* submenu. Alternatively, press *Alt+2*. Select the *Model Glossary* tab.

Add a Glossary Entry

To add an entry to the glossary, follow the steps below:

1. Double click in the *Model Glossary* tab **-OR-** select *Add New* from the right click context menu. This will open the *Glossary Detail* dialog.



2. Enter the details for the glossary item - the *Term*, the *Type* and the *Meaning*.
3. Press *Save*.
4. To create another entry, press *New*.
5. To close, press *OK*.

Modify a Glossary Entry

To modify a glossary entry, either:

1. Double click the entry you want to modify in the list on the *Model Glossary* tab. This will open the *Glossary Detail* window for editing.

-OR-

1. Right click the entry you want to modify in the list on the *Model Glossary* tab. This will bring up the context menu.

2. Select the *Modify Selected* menu item. This will open the *Glossary Detail* window for editing.

Delete a Glossary Entry

To delete a glossary entry, follow the steps below:

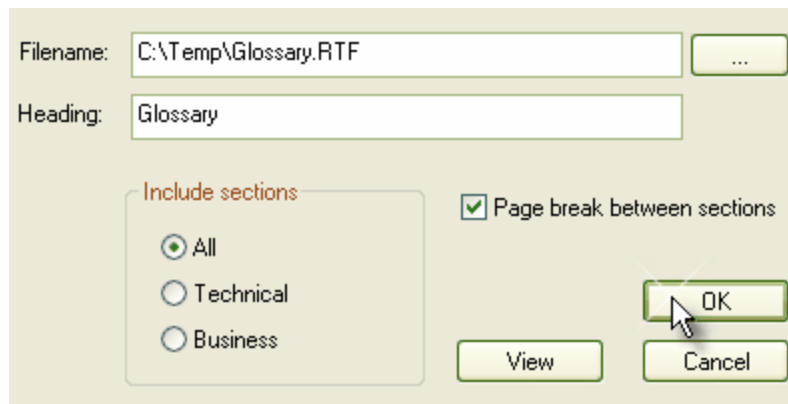
1. Right click the entry you want to modify in the list on the *Model Glossary* tab. This will bring up the context menu.
2. Select the *Delete* menu item.

9.9.4 Generate a Report

Generate a Report

You can generate a report of your model's glossary by following these steps:

1. Select Glossary from the *Project | Documentation* menu, to open the *Glossary* dialog.
2. Press *Report*. This will open the *Glossary Report* dialog.



Filename: C:\Temp\Glossary.RTF

Heading: Glossary

Include sections

All

Technical

Business

Page break between sections

View OK Cancel

3. Enter a filename and a heading for the glossary.
4. Select whether you want to include all sections, or only the Technical or Business sections.
5. If you want to include page breaks, check the *Page break between sections* option.
6. Press *OK* to generate the report.
7. Press *View* to open the report.

Tip: You can view sample report output in the [Glossary Report Output Sample](#) topic.

9.9.5 Glossary Report Output Sample

An example of the output from a Glossary report can be seen below:

Glossary

Business Terms

Accounting Periods

A defined period of time whereby performance reports may be extracted. (normally 4 week periods).

Customer

A person or a company that requests An entity to transport goods on their behalf.

Technical Terms

Association

A relationship between two or more entities. Implies a connection of some type - for example one entity uses the services of another, or one entity is connected to another over a network link.

Component Model

The component model provides a detailed view of the various hardware and software components that make up the proposed system. It shows both where these components reside and how they inter-relate with other components. Component requirements detail what responsibilities a component has to supply functionality or behavior within the system.

Deployment Model

A model of the system as it will be physically deployed

Extends Relationship

A relationship between two use cases in which one use case 'extends' the behavior of another. Typically this represents optional behavior in a use case scenario - for example a user may optionally request a list or report at some point in a performing a business use case.

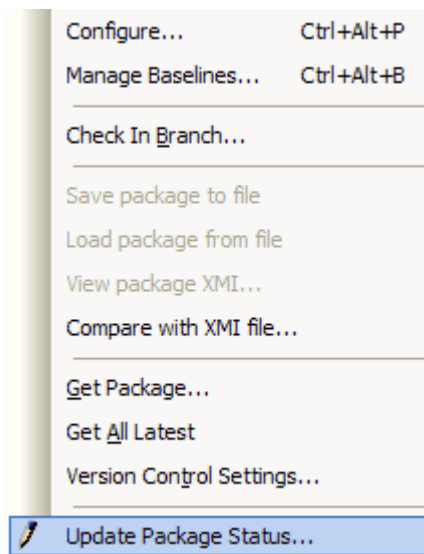
9.10 Update Package Status

Elements in EA may be assigned a current status, such as Proposed, Validated, Mandatory etc. Often a complete package structure will be updated from one status to another at the same time (or released). To help facilitate this, EA supports a 'bulk' update of element status at the same time

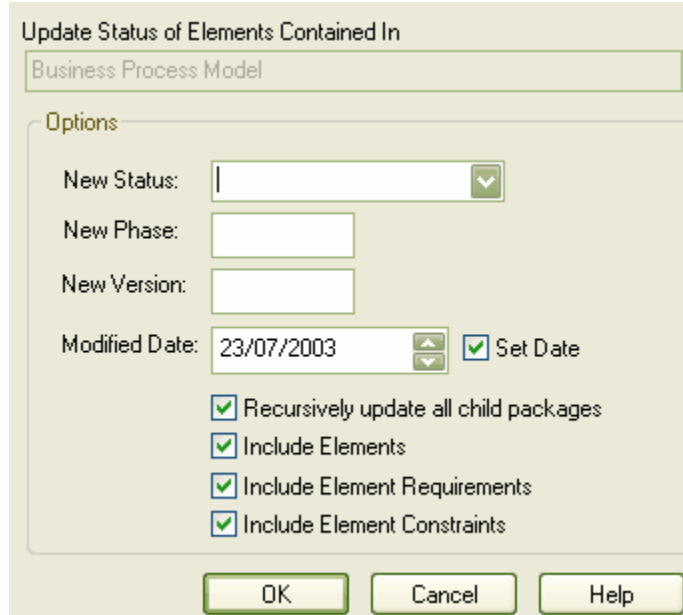
Update Element Status for a Complete Package Structure

To update element status for a complete package structure, follow the steps below:

1. In the Project Browser, right click on the package you need to update, to open the context menu.
2. Select **Update Package Status** from the **Package Control** submenu. This will open the **Status Update** dialog.



3. In the *Status Update* update dialog select:
 - The new status
 - Whether to recursively descend the package tree
 - Whether to include elements
 - Whether to include element requirements
 - Whether to include element constraints



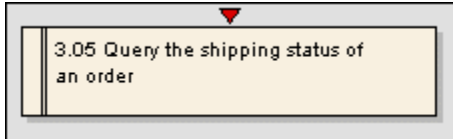
4. Press **OK** when you have configured the settings. EA will update all required elements to the new status.

9.11 Manage Bookmarks

Bookmarks are small red triangles that appear above elements in diagrams if the element has been 'bookmarked'. A Bookmark is a visual clue that something is different about an element - the meaning is up to you.

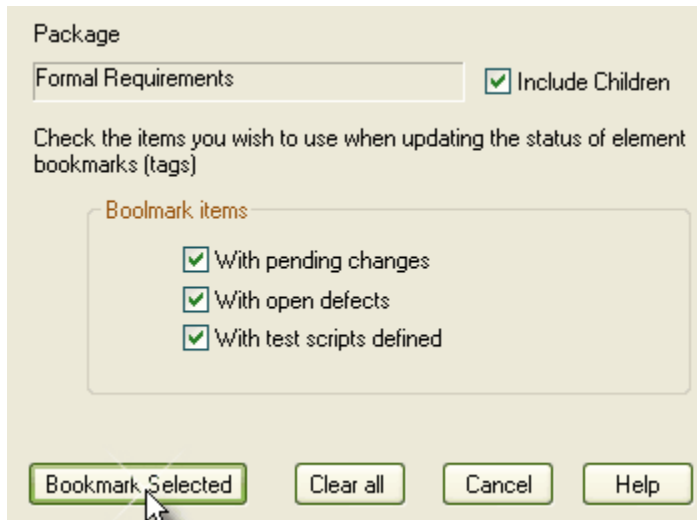
Tip: The [Search](#) dialog also allows searching based on bookmarked elements.

You can bookmark an element manually by pressing the **Shift+Space** keys:



Bookmark Multiple Elements

You can also bookmark all elements in a folder (and their children) using the [Manage Bookmarks](#) dialog. This is accessible from the package context menu in the [Project Browser](#) - select the [Bookmarks](#) menu item.



This dialog lets you automatically bookmark elements with changes, with defects or with test scripts defined. This is useful to highlight elements that have additional project information, or alternatively, those elements that do not.

You can press **Clear All** to clear all elements in the current tree of bookmarks.

Part

10

10 Code Engineering

Code Engineering is a process which includes the processes of code generation, reverse engineering of source code and synchronization between the source code and model. Code Engineering is only available in the Professional and Corporate Editions of Enterprise Architect, Desktop edition owners can upgrade to either of the aforementioned editions from the [EA Upgrade.htm](#) page.

Enterprise Architect allows you to generate source code from UML models. In particular you may generate C++, C#, Delphi, Java, PHP, Python, Action Script, Visual Basic and VB.NET source code. The source code generated includes class definitions, variables and function stubs for each attribute and method in the UML class.

Generating code is also known as Forward Engineering. Importing source code into model elements is known as Reverse Engineering the languages that are available for generation with EA are also available for reverse engineering.

Synchronization is when changes in the model are exported to the source and changes to source are imported into the model. This allows you to keep your model and your source code up to date as the project develops.

Round trip engineering occurs as a combination of reverse and forward generation of code and should include synchronization between the source code and the model except in all but the most trivial of code engineering projects.

See Also

- [Reverse Engineer Source Code](#)
- [Generate Source Code](#)
- [Code Engineering Settings](#)
- [Language Options](#)
- [Code Template Framework](#)
- [Modeling Conventions](#)
- [XML Technologies](#)

10.1 Reverse Engineer and Synchronizing

Reverse Engineering in EA allows the user to import existing source code from a variety of code languages into a UML model. Existing source code structures will be mapped into their UML representations, for example a Java class will be mapped into a UML class element with the variables being defined as attributes, methods are modeled as operations and the interactions between the Java classes being displayed in the UML model class diagram with the appropriate connectors.

Reverse Engineering allows users to examine legacy code and examine the functionality of code libraries for reuse or to bring the UML model up to date with the code that has been developed as part of a process called synchronization. Examining the code in a UML model allows user to identify the critical modules contained the code, allowing a starting point for understanding of the business and system requirements of the pre-existing system and to enable the developers to gain a better overall understanding of the source code.

To begin the process of importing existing code into EA an existing source of code needs to be [imported into EA](#), which may be a single directory or [directory structure](#). Several options are available when performing the reverse engineering process. The [Source Code Engineering options](#) page contains several options that effect the reverse engineering process. These include:

- If comments are reverse engineered into notes fields, and how they are formatted if they are.
- How property methods are recognized.
- If dependencies should be created for operation return and parameter types.

It is important to note that when a legacy system has been poorly designed that simply importing the source into EA will not create an easily understood UML model. When working with legacy system which is poorly designed it is useful to break down the code into manageable components by examining the code elements on a "per element" basis. This can be achieved by importing a specific class of interest into a diagram and then [inserting the related elements](#) at one level to determine immediate relationship to other classes. From

at this point it is possible to create use cases that identify the interaction between the legacy classes enabling an overview of the legacy systems operation.

Copyright ownership is an important issue to take into account when undertaking the process of reverse engineering. In some cases, software will have specific limitations which prohibit the process of reverse engineering, it is important that a user address the issue of copyright before beginning the process of reverse engineering of code. Situations which typically lend themselves to the reverse engineering of source code include source code which is:

- Source code which you have already developed
- That is part of a 3rd party library which you have obtained permission to use.
- Part of a framework which your organization uses.
- That is being developed on a daily basis by your developers.

Enterprise Architect currently supports reverse engineering in the following programming languages

- [Actionscript](#)
- [C#](#)
- [C++](#)
- [Delphi](#)
- [Java](#)
- [PHP](#)
- [Python](#)
- [Visual Basic](#)
- [Visual Basic .NET](#)

Enterprise Architect is also able to reverse engineer certain types of binary files: Java .jar files and .NET PE files. See [Import Binary Module](#) for more information.

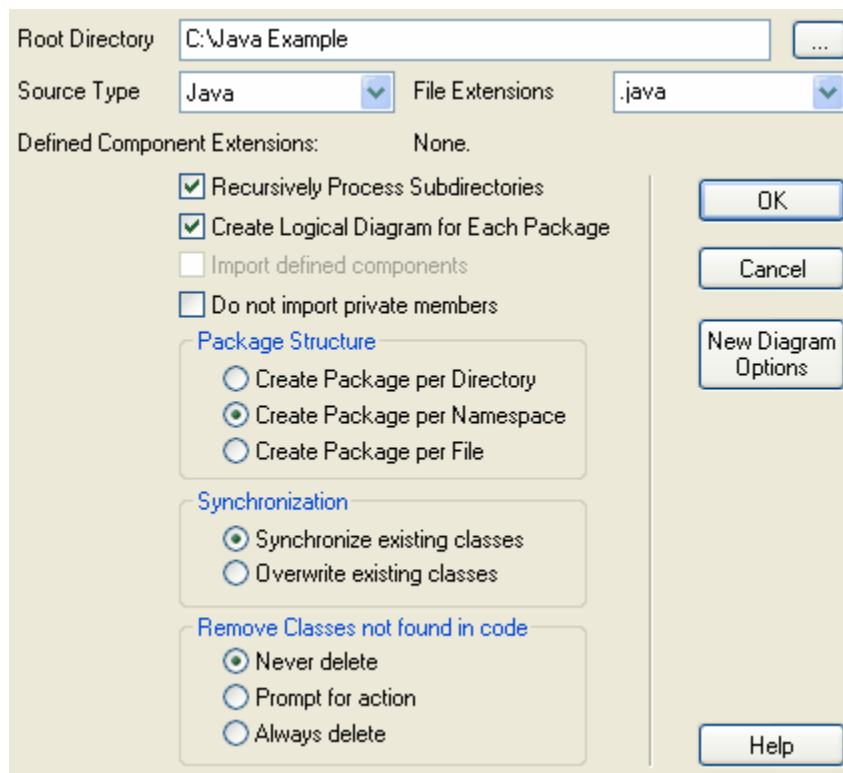
Note: Reverse Engineering of other languages including CORBA IDL is currently available through the use of MDG Technologies from www.sparxsystems.com.au/resources/mdg_tech/.

10.1.1 Import a Directory Structure

You may elect to import from all source files in a complete directory structure. This process allows you to import or synchronize multiple files in a directory tree in one pass. Enterprise Architect will create the necessary packages and diagrams during the import process.

To import a directory structure, follow the steps below:

1. In the Project Browser, right click on a package in the Project View.
2. From the context menu, select **Import Source Directory** from the **Code Engineering** submenu.
3. In the **Import Directory Structure** dialog, select the options you require. You may configure:
 - The source directory
 - The source type
 - The file extensions to look at
 - Whether to recurse sub directories
 - Whether to create a diagram for each package
 - To create a package for every directory, namespace or file. This may be restricted dependant on the source type selected.
 - Whether to Synchronize or Overwrite existing classes when found
 - Whether to import additional files as described in the Import Component Types dialog.
 - Whether to exclude private members from libraries being imported from the model.
 - How to handle classes not found during the import.
 - What is shown on diagrams created by the import.
4. Press **OK** to start



10.1.2 Import ActionScript

ActionScript code may be imported into Enterprise Architect. When you import ActionScript, you will need to select the appropriate source file (.as) as the source code to import. EA supports most ActionScript constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.3 Import C

C code may be imported into Enterprise Architect. When you import C, you will need to select the appropriate header (.h) file(s) and/or .c file(s) as the source code to import.

EA supports most C constructs and keywords; however it will not support some extensions and macros used by Visual Studio. Future releases of EA will support more of these non-standard extensions.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.4 Import C++

C++ code may be imported into Enterprise Architect. When you import C++, you will need to select the appropriate header (.h) file as the source code to import. EA does not use information in the .cpp file.

EA supports most C++ constructs and keywords; however it will not support some extensions and macros used by Visual Studio or Borland. Future releases of EA will support more of these non-standard extensions.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.5 Import C#

C# code may be imported into Enterprise Architect. When you import C#, you will need to select the appropriate source file (.cs) as the source code to import. EA supports most C# constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.6 Import Delphi

Delphi code may be imported into Enterprise Architect. When you import Delphi, you will need to select the appropriate source file (.pas) as the source code to import.

EA supports most Delphi constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.7 Import Java

Java code may be imported into Enterprise Architect. When you import Java, you will need to select the appropriate source file (.java) as the source code to import.

EA supports most Java constructs and keywords including the AspectJ language extensions.



Aspects are modelled using classes with the stereotype "aspect". These aspects can then contain attributes and methods the same as a normal class. If an intertype attribute or operation is required a tag "className"

with the value being the name of the class it belongs to can be added.

Pointcuts are defined as operations with the stereotype of "pointcut". These may occur in any java class, interface or aspect. The details of the pointcut are included in the behaviour field of the method.

Advice is defined as an operation with the stereotype "advice". The pointcut this advice operates on is in the behaviour field and acts as part of the methods unique signature. After advice may also have one of the tagged values "returning" or "throwing".

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.8 Import PHP

PHP code can now be imported into Enterprise Architect. When you import PHP, you will need to select the appropriate source file (.php, .php4, .inc) as the source code to import.

EA supports most PHP constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.9 Import Python

Python code may be imported into Enterprise Architect. When you import Python, you will need to select the appropriate source file (.py) as the source code to import.

EA supports most Python constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.10 Import Visual Basic

Visual Basic code may be imported into Enterprise Architect. When you import Visual Basic, you will need to select the appropriate class file (.cls) file as the source code to import.

EA supports most Visual Basic constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.11 Import VB.Net

VB.Net code may be imported into Enterprise Architect. When you import VB.Net, you will need to select the appropriate class file (.vb) file as the source code to import.

EA supports most VB.Net constructs and keywords.

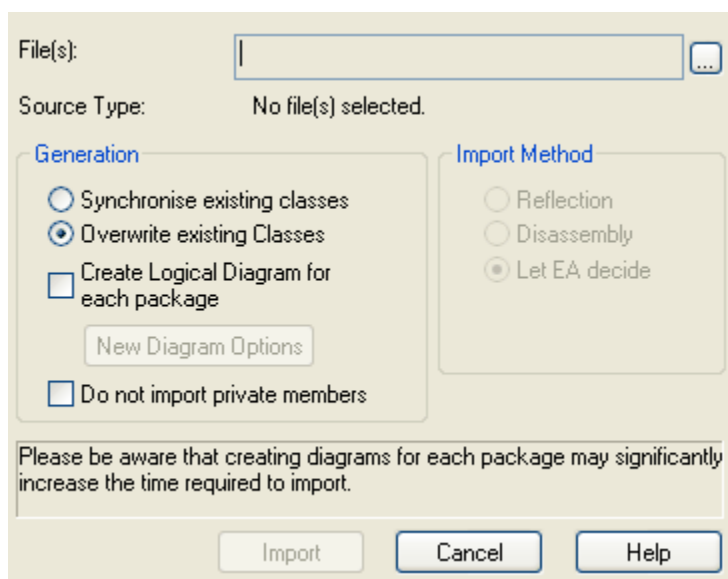
If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

See Also

- [Import Source Code](#)

10.1.12 ***Import Binary Module***

Enterprise Architect (Professional and Corporate editions) allows for certain types of binary modules to be reverse-engineered. To import a binary module, right-click the target package in the **Project View** window and select the "**Code Engineering** | **Import Binary Module**" command.



Currently the permitted types are as follows:

- **Java Archive** (.jar).
- **.Net PE file** (.exe, .dll). Native Windows DLL and Exe files are not supported. Only PE files containing .Net assembly data.
- **Intermediate Language file** (.il).

Enterprise Architect will create the necessary packages and diagrams during the import process. Checking the **Do not import private members** checkbox is used to exclude private members from libraries from being imported into the model.

When importing .Net files, the option is available to import via reflection or disassembly. The choice to let EA decide what the best method is may result in both types being used. The reflection based importer relies on a .Net program, and requires the .Net runtime environment to be installed. The disassembler based importer relies on a native Windows program called Ildasm.exe which is a tool provided with the MS .Net SDK. The SDK may be downloaded from the Microsoft website.

A choice of import methods is available because some files are not compatible with reflection (such as mscorlib.dll) and may only be opened using the disassembler. However, the reflection-based importer is generally much faster.

10.1.13 MDG Link and Code Engineering

The MDG Link for Eclipse and MDG Link for Visual Studio.NET are standalone products which provide an enhanced code engineering functionality between EA and the development environments.

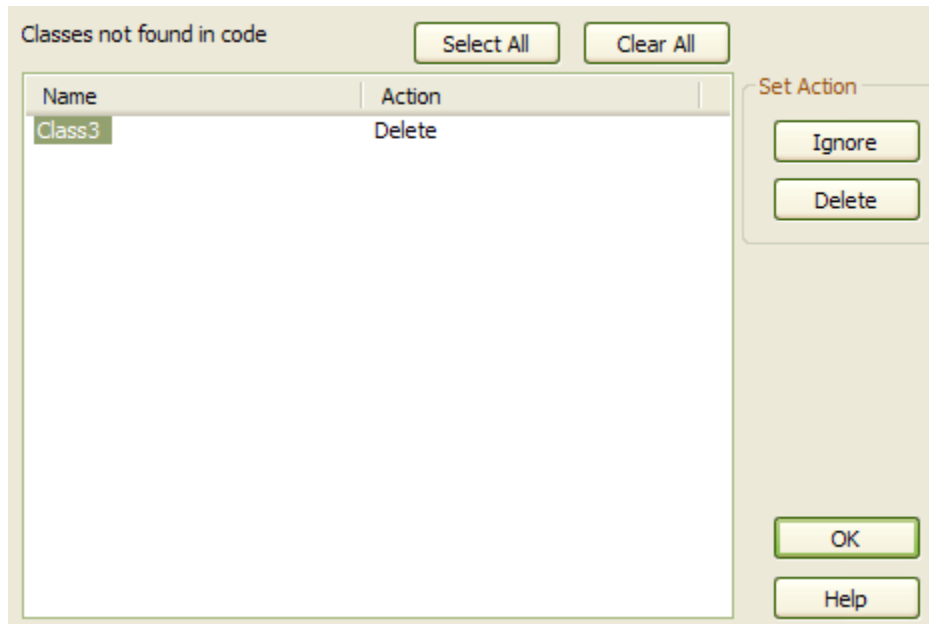
The MDG Link programs provide a lightweight bridge between EA and the development environment offering enhanced code generation, reverse engineering and synchronization between code and the UML model. Merging changes can be achieved with minimal effort and navigation between model and source code is significantly enhanced.

A trial version of MDG Link for Eclipse may be downloaded from www.sparxsystems.com.au/products/mdg_eclipse.html and MDG Link for Visual Studio.NET may be downloaded from www.sparxsystems.com.au/products/mdg_vs.html.

10.1.14 Handling of Classes not found during Import

When reverse synchronizing from your code, there are times when some classes may be deliberately removed from your source code. EA's import source directory functionality keeps track of the classes it expects to synchronize with and allows the option of how to handle the classes that weren't found. At the end of the import EA will either ignore the missing classes, automatically delete them or prompt you to handle them. You have the choice to ignore them, automatically delete them, or prompt you for how to handle them.

The dialog to delete classes looks like this.



By default, all classes are marked for deletion. To keep one or more classes select them and press the *Ignore* button.

10.1.15 Import Source Code

To import source code (reverse engineer) you will usually do the following:

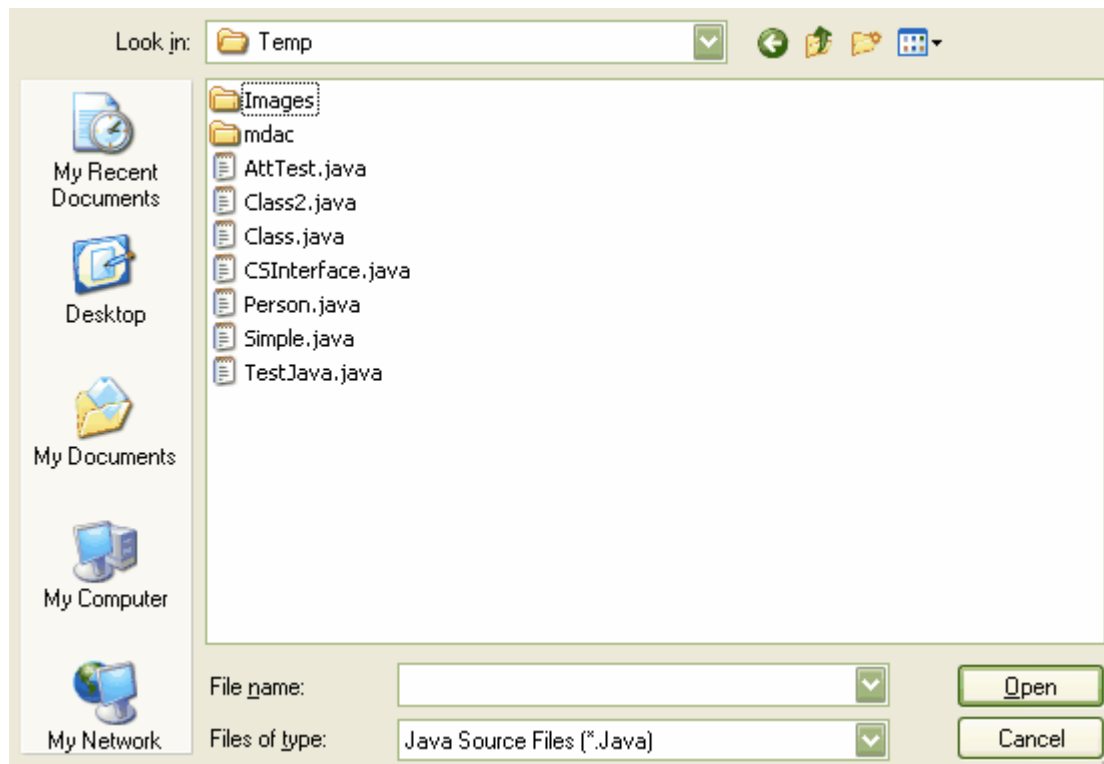
1. In the Project Browser, select (or add) a diagram into which the classes will be imported.
2. Right click on the diagram background to open the context menu. Select the language you wish to

import from the *Import from source file(s)* submenu.

-OR-

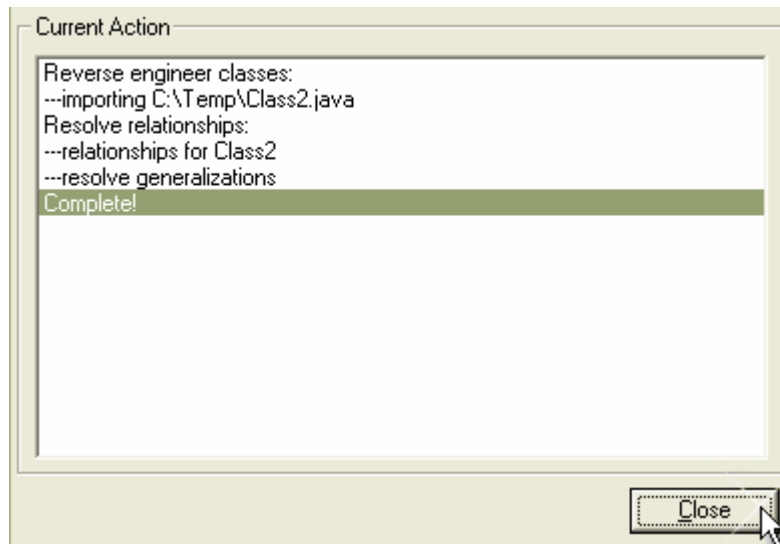
Use the drop down *Import Language* list in the *Code Generation* toolbar - select the *Import | Import xxx files* item, where *xxx* represents the language you wish to import

3. From the file browser that appears, select one or more source code files to import.



4. Press **OK** to start the import process.

As the import proceeds, EA will provide progress information. When all files are imported, EA makes a second pass to resolve and associations and inheritance relationships between the imported classes.



10.1.16 Synchronize Model and Code

In addition to generating code and importing code - EA has the option to synchronize the model and source code. This allows the model and the source code to be synchronized creating a model which is up to date with the latest changes in the source code and vice versa.

For example - you may have already generated some source code, but have made subsequent changes to the model. When you generate again, EA will add any new attributes or methods to the existing source code, leaving intact what already exists. This means that developers may work on the source code, and then generate additional methods as required from the model, without having their code overwritten or destroyed.

In another scenario, changes may have been made to a source file, but the model has detailed notes and characteristics you do not want to lose. By synchronizing from the source into the model, additional attributes and methods are imported, but other model elements are left alone.

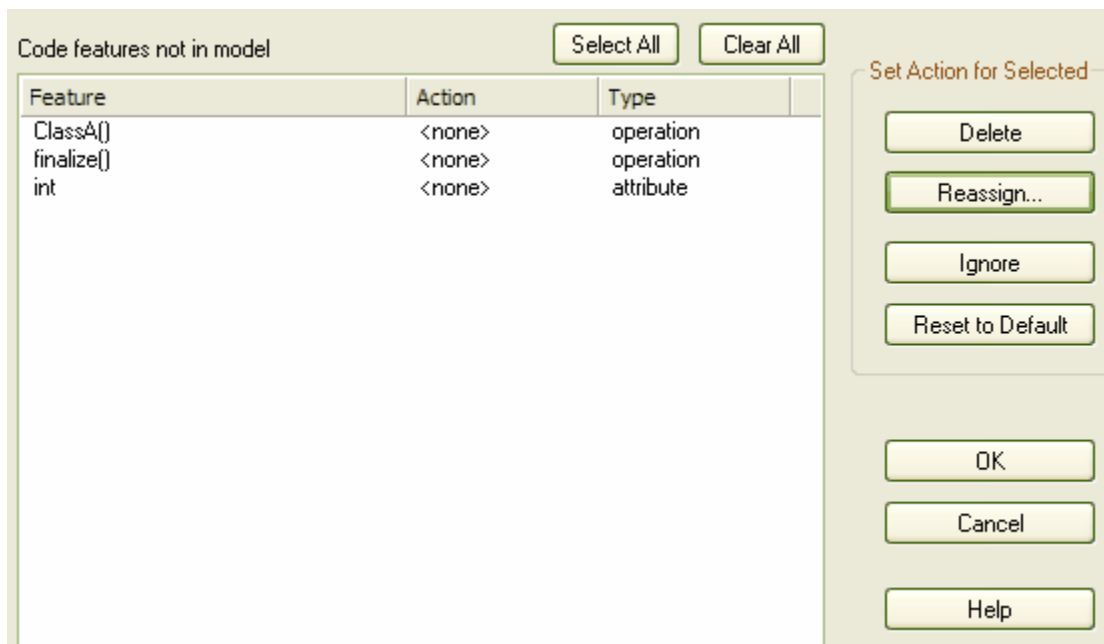
Using the two synchronization methods above, it is simple to keep source code and model elements up to date and synchronized.

Synchronizing Classes on Forward Generation

When there are features present in the code but not in the model the following options may be used during forward synchronization:

Note: these features are only available when the *On forward synch, prompt to delete code features not in model* in the [Options - Attributes and Operations](#).

- **Delete**, when the **Delete** button is pressed the selected code features will be removed from the code.
- **Reassign**, when the **Reassign** button is pressed it is possible to reassign code elements to elements in the model (this is only possible when an appropriate model element is present and is not already defined in the code).
- **Ignore**, when this button is selected the code elements not present in the model are ignored completely.
- **Reset to Default**, when this button is selected the settings for synchronizing during forward generation is set to **Ignore** meaning that the elements present in the code but not in the model are ignored completely.



10.2 Generate Source Code

Generating source code (forward engineering) takes the UML class or interface model elements and creates a source code equivalent for future elaboration and compilation. By forward engineering code from the model the mundane work involved with having to key in classes and attributes and methods is removed, and symmetry between model and code is ensured.

Code is generated from Class or Interface model elements, so you must create the required class and interface elements to generate from. Add attributes (which become variables) and operations (which become methods).

Before you generate code, you should ensure the default settings for code generation match your requirements. The default generation settings are located in the *Generation* section of the *Local Options* dialog (accessed by selecting *Options* from the *Tools* menu). Set up the defaults to match your required language and preferences. Preferences that may be defined include default constructors and destructors, methods for interfaces and the Unicode options for created languages. In addition to the default settings for generating code EA supports the following code languages with their own specific code generation options:

- [Actionscript](#)
- [C#](#)
- [C++](#)
- [Delphi](#)
- [Java](#)
- [PHP](#)
- [Python](#)
- [Visual Basic](#)
- [Visual Basic .NET](#)

The Code Template Framework (CTF) allows for the customization of the way EA generates source code and also allows for the generation of languages that are not specifically supported by EA.

When you have completed the design of your classes, you may generate source code.

See Also

- [How to Generate Code](#)
- [The Code Generation Dialog](#)

- [Generate a Group of Classes](#)
- [Generate a Package](#)
- [Generate Package Dialog](#)
- [Update Package Contents](#)
- [Namespaces](#)

10.2.1 How to Generate Code

Before you generate code, you should ensure the default settings for code generation match your requirements. The default generation settings are located in the *Generation* section of the *Local Options* dialog (accessed by selecting *Options* from the *Tools* menu). Set up the defaults to match your required language and preferences. Languages such as Java support namespaces and may be configured to specify a namespace root.

Code is generated from Class or Interface model elements, so you must create the required class and interface elements to generate from. Add attributes (which become variables) and operations (which become methods). When you have completed the design of your classes, you may generate source code.

Before generating code, you should also familiarize yourself with the way Enterprise Architect handles local path names. Local path names allow you to substitute tags for directory names (eg. %SRC% = C:\Source).

See Also

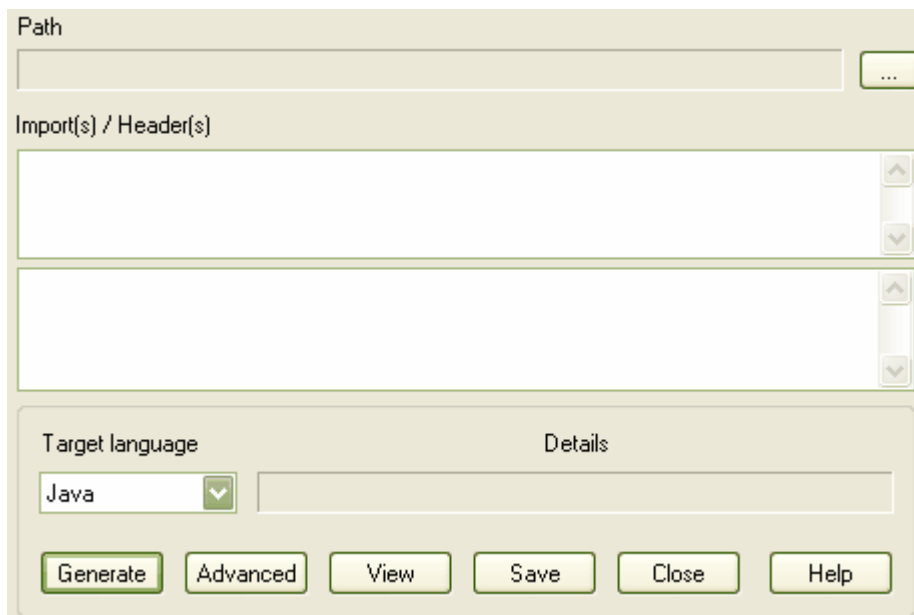
- [Generate a Single Class](#)
- [Generate a group of Classes](#)
- [Generate a Package](#)

10.2.2 Generate a Single Class

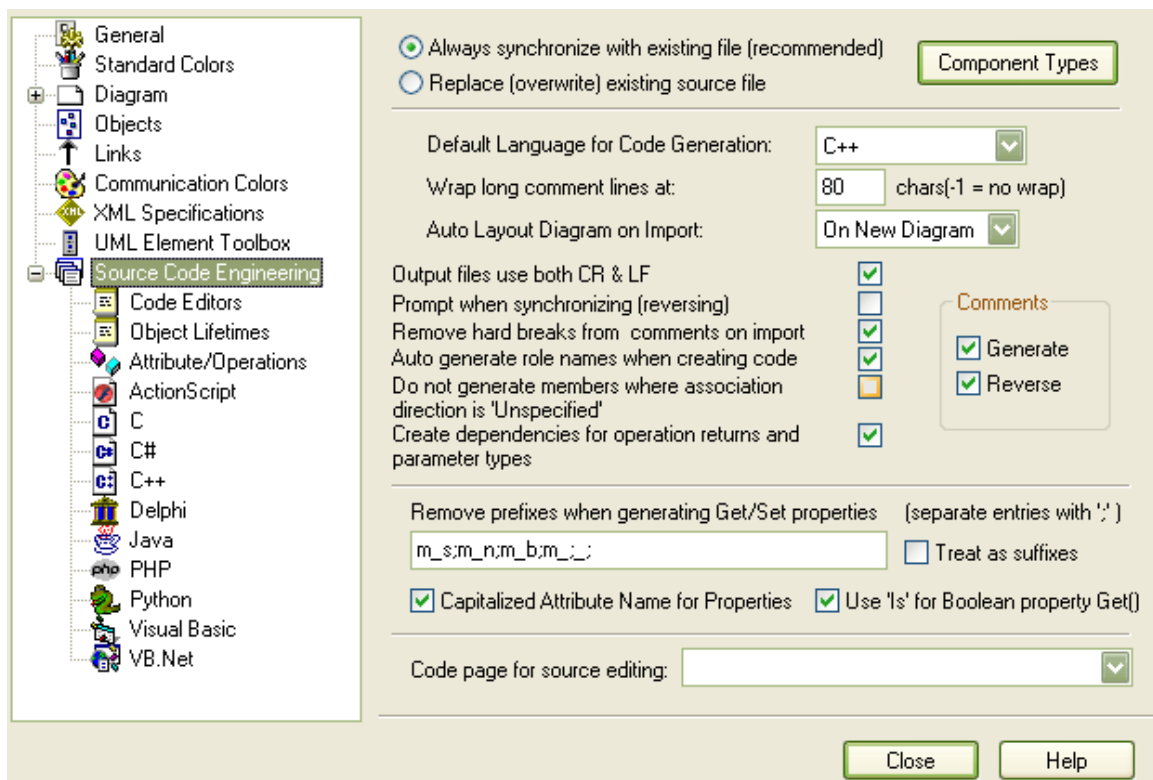
To generate code for a single class, first ensure the design of the model element (class or interface) is complete. Also ensure you have added inheritance links to parents and associations to other classes which are used. Also add inheritance links to Interfaces that your class implements - Enterprise Architect offers the option to generate function stubs for all interface methods that a class implements. Once the design is satisfactory, follow the steps below.

To Generate Code for a Single Class

1. Open the diagram containing the class or interface for which you wish to generate code.
2. Right click on the class/interface and select *Generate Code...* from the context menu.
3. The *Generate Code* dialog appears. Enter a *Path* name for your source code



4. Press *Advanced* to set any custom options (for this class alone).



5. Add any import statements, #includes or other header information in the spaces provided (note that in the case of Visual Basic this information is ignored, in the case of Java the two import text boxes are merged and in the case of C++ the first import text area is placed in the header file and the second in the body (.cpp) file)

6. Press **Generate** to create the source code

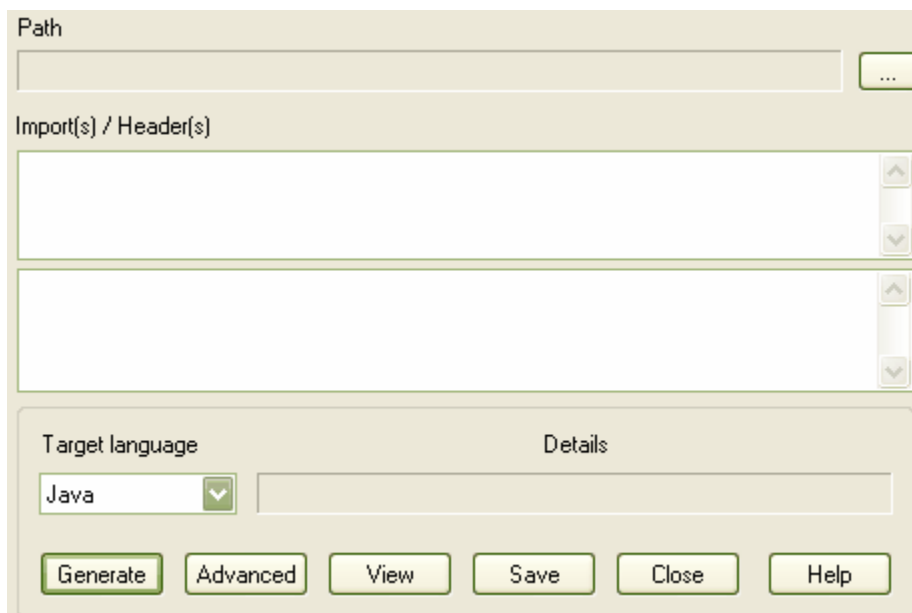
When complete, you may press **View** to see what has been generated. Note that you should set up your default viewer/editor for each language type first.

See Also

- [Source Code Viewer](#)

10.2.3 The Code Generation Dialog

The **Code Generation** dialog allows you to control how and where your source code is generated. Normally you will access this dialog from the context menu of a single class or interface. Right click on the class or interface and select **Generate Code** from the context menu. Alternatively, select the class or interface and press **Ctrl+G**.



This dialog allows you to set

- The **Path** where the source will be generated. Press the **Browse [...]** button to bring up a file browser dialog.
- The **Target Language** for generation. Select the language to generate - this will then become the permanent option for that class, so change it back if you only want to do one pass in another language.
- **Advanced** settings. Note that the settings you make here only apply to the current class.
- **Import statements #1**. An area for you to enter any special import statements (or #include in C++). For C++ this area is placed in the header file.
- **Import statements #2**. An area to define additional import or include statements (or even macros and #defines in C++). In C++ this area is placed in the CPP file, in Java it is appended to the first import statements and placed in the .java file.
- **Generate**. Press this to generate your source code - you will be advised of progress as the generation proceeds.
- **View**. Press this to view the generated source code in your default editor. You may also set up the default editor on the **Code Editors** section of the **Local Options** dialog (**Tools | Options**).

10.2.4 Generate a Group of Classes

In addition to being able to generate code for an individual class, you may also select a group of classes for batch code generation. When you do this, you will accept all the default code generation options for each class in the set.

To Generate Multiple Classes

1. Select a group of classes and/or interfaces in a diagram.
2. Right click on an element in the group to open the context menu.
3. Select *Generate all Selected objects with default options* from the *Code Generation* submenu.
4. If no output name has been specified for the class/interface already, EA will prompt you for a suitable name as the generation proceeds.

Note: If all the elements selected are not classes or interfaces the option to generate code will not be available.

10.2.5 Generate a Package

In addition to generating source code from single classes and groups of classes, you can also generate code from a package. This feature provides options to recursively generate child packages and automatically generate directory structures based on the package hierarchy. This allows you to generate a whole branch of your project model in one step.

The Package Code Generation dialog is accessed by selecting *Generate Package Source Code* from the *Project | Source Code Engineering* submenu. Alternatively, right click on a package from the Project Browser and select *Generate Source Code* from the *Code Engineering* submenu.

Root Package: Pages

Synchronize: Synchronize model and code

Generate:

Auto Generate Files Root Directory: ...

Retain Existing File Paths

Select Objects to Generate Include all Child Packages

| Object | Type | Target File |
|-----------------|-------|-------------|
| ContactListPage | Class | |
| JspPage | Class | |

Select All Select None

| Control | Description |
|----------------------------|--|
| Root Package | The name of the package to be generated |
| Synchronize | These options specify how existing files should be generated |
| Auto Generate Files | Specifies whether EA should automatically generate file names and directories, based on the package hierarchy |
| Root Directory | If Auto Generating Files, specifies the path under which the generated directory structure will be created |
| Retain Existing File Paths | If Auto Generating Files, specifies whether to use existing file paths associated with classes. If unselected, EA generates classes to automatically determined paths, regardless of whether source files are already associated with classes. |
| Include all Child Packages | If checked, all classes from all sub-packages of the target package are included in the list. This option facilitates recursive generation of a given package and its sub-packages. |
| Select Objects to Generate | Lists all classes that are available for generation under the target packages. Only selected (highlighted) classes will be generated. Classes are listed with their target source file. |
| Select All | Marks all classes in the list as selected |
| Select None | Marks all classes in the list as unselected |
| Generate | Starts the generation of all selected classes |
| Cancel | Exits the Code Template Editor dialog. No classes will be generated |

Generate a Package

To generate a package, follow these steps:

1. In the Project Browser, right click on the package you wish to generate code for.
2. In the context menu, select *Generate Source Code* from the *Code Engineering* submenu.
3. The *Generate Code* dialog will appear.
4. Select the appropriate synchronize option:
 - *Synchronize model and code* : Classes with existing files will be forward synchronized with that file. Classes with no existing file, will be generated to the displayed target file.
 - *Overwrite code* : All selected target files will be overwritten (forward generated).
 - *Do not generate* : Only selected classes that do not have an existing file will be generated. All other classes will be ignored.
5. Highlight the classes you wish to generate. Leave unselected any you do not wish to generate.
6. If you would like EA to automatically generate directories and filenames based on the package hierarchy, check the *Auto Generate Files* option. You will be prompted to select a root directory under which the source directories will be generated. By default, the Auto Generate feature ignores any file paths that are already associated with a class. You can change this behavior by checking the *Retain Existing File Paths* option.
7. If you wish to include all sub-packages in the output, check the *Include Child Packages* option. When you have made your selection, press *Generate* to start the process.

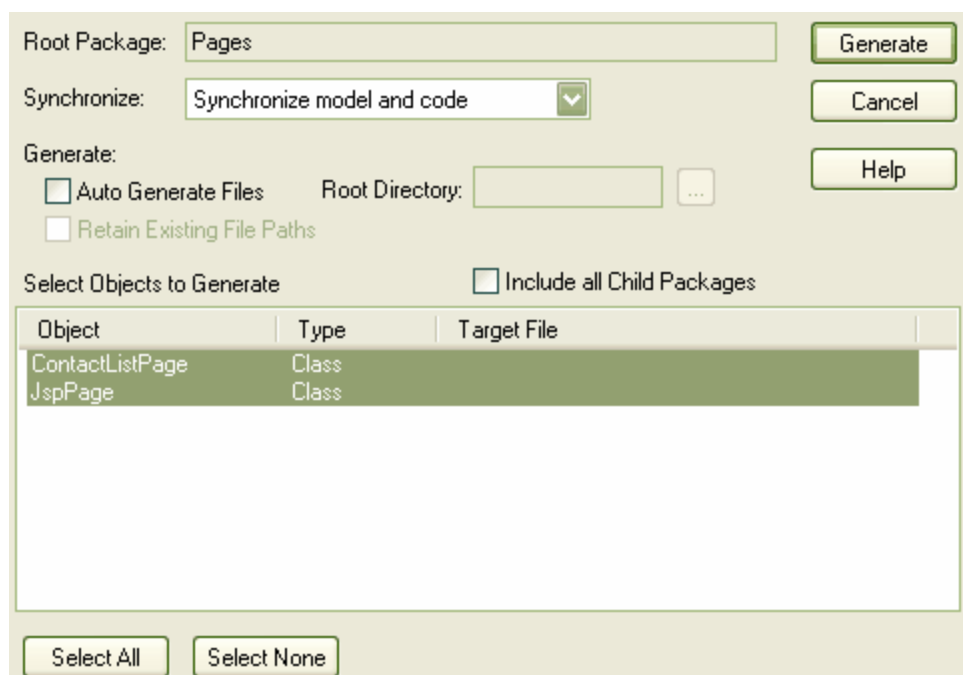
As the code generation proceeds you will be kept informed of progress. If a class requires an output filename you will be prompted to enter one at the appropriate time.

10.2.6 Generate Package Dialog

This dialog allows you to set up which classes will be generated into source code during a batch code generation run.

To Access the Generate Package Dialog

1. In the Project Browser, right click on the package to generate code for, to open the context menu.
2. Select *Generate Source Code* from the *Code Engineering* submenu.



3. If you wish to include classes in sub-packages, check the *Include Child Packages* option.
4. Next, highlight the classes in the list that you wish generated and leave unselected those you do not.
5. You may notice that many classes have no file name associated with them - as the generation proceeds you will be prompted to enter a suitable output file name for each class.
6. Finally, press *Generate* to start the process.

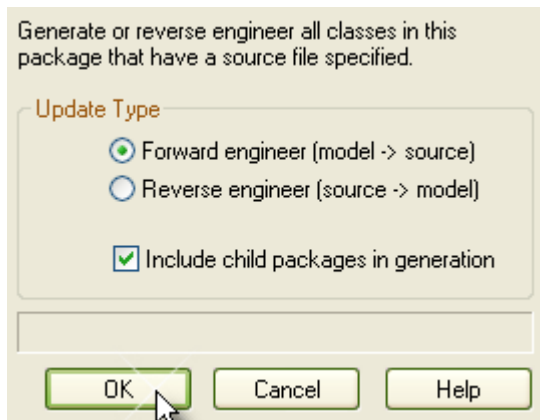
10.2.7 Update Package Contents

EA allows the synchronization of a directory tree. Follow the steps below:

1. Click on the Root package of the tree to synchronize in the Project Browser.
2. Select *Code Engineering* | *Synchronize Package Contents*.

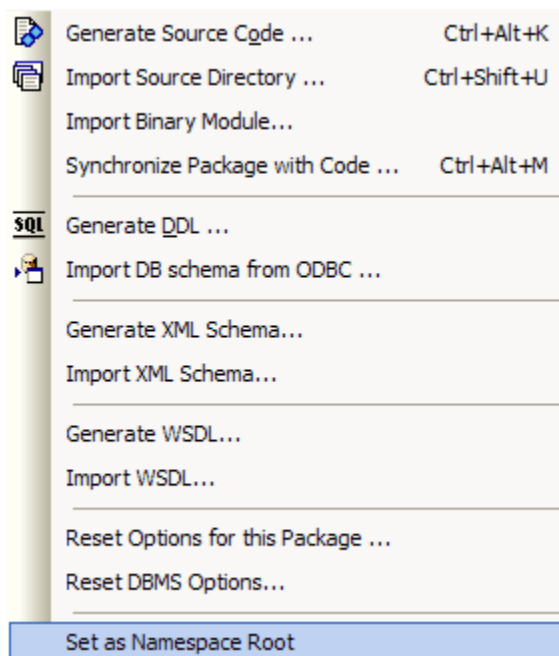
3. Select whether to *Forward Engineer* or to *Reverse Engineer* the package classes.
4. Select whether to *Include child packages* in the synchronization.
5. Press *OK* to start.

EA will use the directory names specified when the project source was first imported/generated and update either the model or the source code depending on the option chosen.



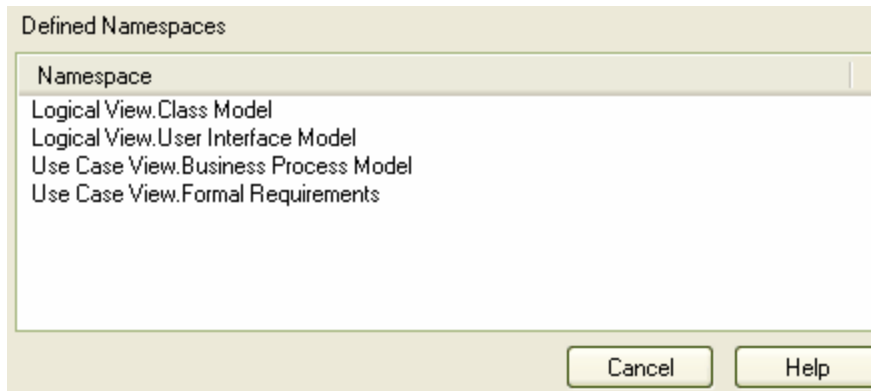
10.2.8 Namespaces

Languages such as Java support package structures or namespaces. EA lets you specify a package as a namespace root. To do this, right click on the package in the Project Browser, and select *Set as Namespace Root* from the *Code Engineering* submenu.

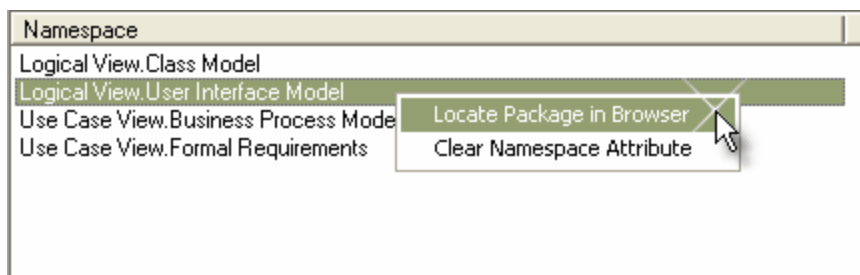


Once you have set a namespace root - Java code generated beneath this root will automatically add a package declaration at the head of the generated file indicating the current package location.

View a list of namespaces in the *Namespaces* dialog - select *Namespaces* from the *Project* menu.



Right click on an entry to remove the Namespace definition or locate the package in the main Project Browser



10.3 Code Engineering Settings

You can set the default code options such as the editors for each of the programming languages available for EA and special options for how source code is generated.

See Also

- [General Options](#)
- [Local Paths](#)
- [Local Path Dialog](#)
- [Language Macros](#)
- [Setting Collection Classes](#)

10.3.1 Source Code Engineering

The following topics describe general options that apply to all languages when generating code from EA. These options are all available under the *Source Code Engineering* section on the *Local Options* dialog (*Tools* | *Options*).

See Also

- [Source Code Options](#)
- [Options - Code Editors](#)

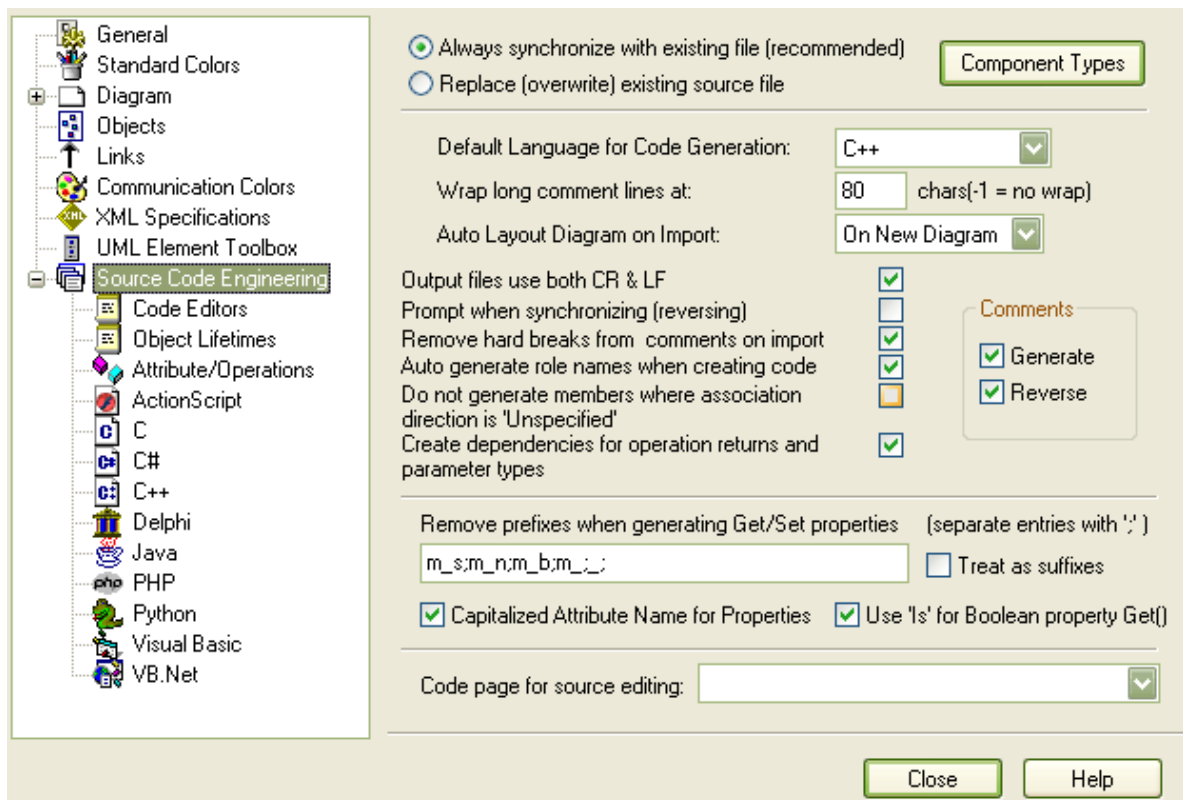
- [Options - Object Lifetimes](#)
- [Options - Attribute/Operations](#)
- [Synchronize Model and Code](#)
- [Code Page for Source Editing](#)

10.3.1.1 Source Code Options

When you generate code for a particular language, there are some options you may set. These include things like:

- Create a default constructor
- Create a destructor
- Generate copy constructor
- Select default language
- Generate methods for implemented interfaces
- Set the unicode options for code generation.

These options are accessed via the *Source Code Engineering* section of the *Local Options* dialog (*Tools / Options*). Below is the main dialog; navigate using the side browser to access other option pages. Most of the settings are self-explanatory. *Remove prefixes when generating Get/Set properties* provides a field to specify prefixes used in your variable naming conventions, if you want those prefixes removed in the variables' corresponding get/set functions.



Note: It is worthwhile to configure these settings, as they serve as the defaults for all classes in the model. You may override these on a per-class basis using the custom settings (from the [Code Generation](#) dialog).

10.3.1.1.1 Import Component Types

The import component types dialog allows you to configure what elements you would like to be created for files of any extension found while importing a source code directory. To access the [Import Component Types](#) dialog open the [Source Code Engineering](#) item from the [Local Options](#) dialog (access from [Tools | Options](#)) and press the [Component Type](#) button.

Specify for this model, the files by extension type, their UML equivalent and an optional stereotype for importing additional items when reverse engineering directories.

| Extension | Type | Stereotype |
|-----------|-----------|------------|
| sln | Artifact | solution |
| bmp | Component | bitmap |
| sln | Artifact | solution |
| rc | Artifact | resource |

Save New Delete Close

For each extension you can specify:

- The element type to be created.
- The stereotype to apply to these objects.

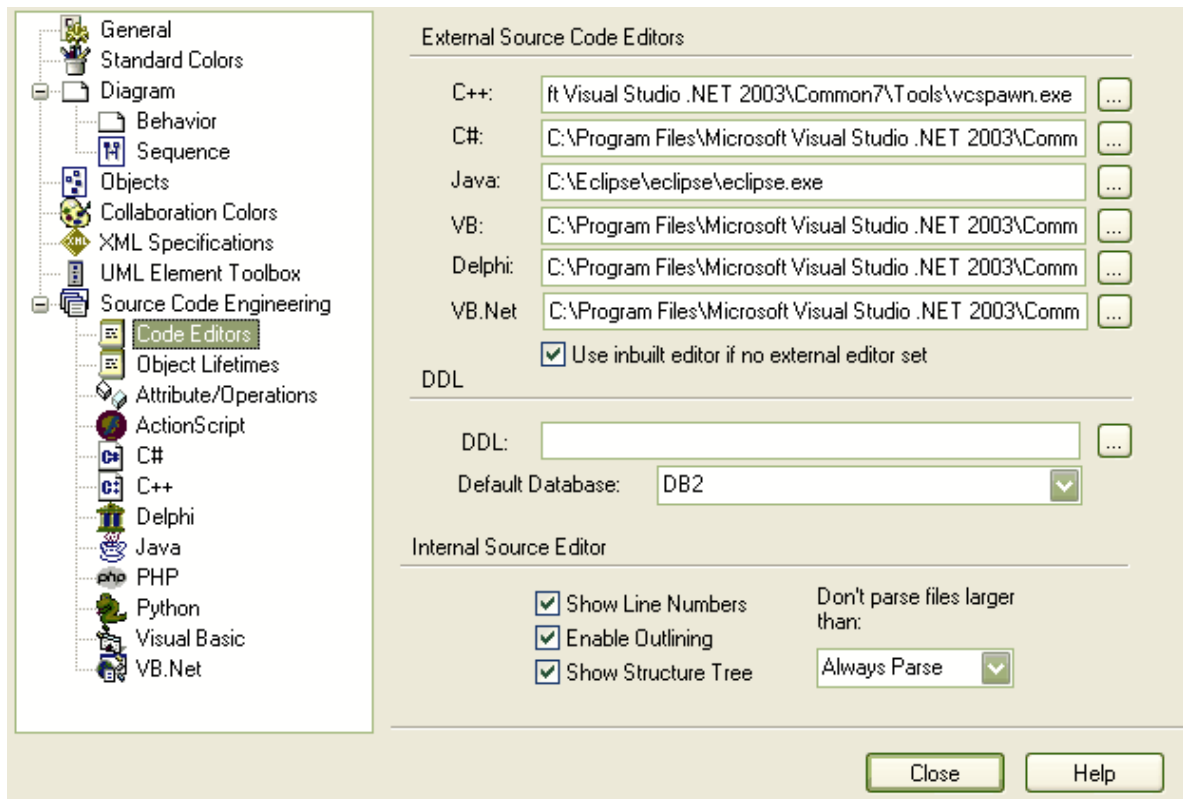
This information is stored in your current model, and can be transferred between different models by selecting [Import Component Types](#) from the [Export Reference Data](#) dialog.

10.3.1.2 Options - Code Editors

These options are accessed via the [Source Code Engineering](#) section of the Local Options dialog ([Tools | Options](#)). They allow you to configure options for EA's internal editor as well as default editors for each language and DDL scripts.

The options for the inbuilt editor are:

| Control | Description |
|-------------------------------|--|
| Show Line Numbers | Show line numbers in the editor. |
| Enable Outlining | Allow collapsible regions for standard languages. |
| Show Structure Tree | Show a tree with the results of parsing the open file (Requires that the file is parsed successfully). |
| Don't parse files larger than | Specify an upper limit on file size for parsing. Used to prevent performance decrease due to parsing very large files. |

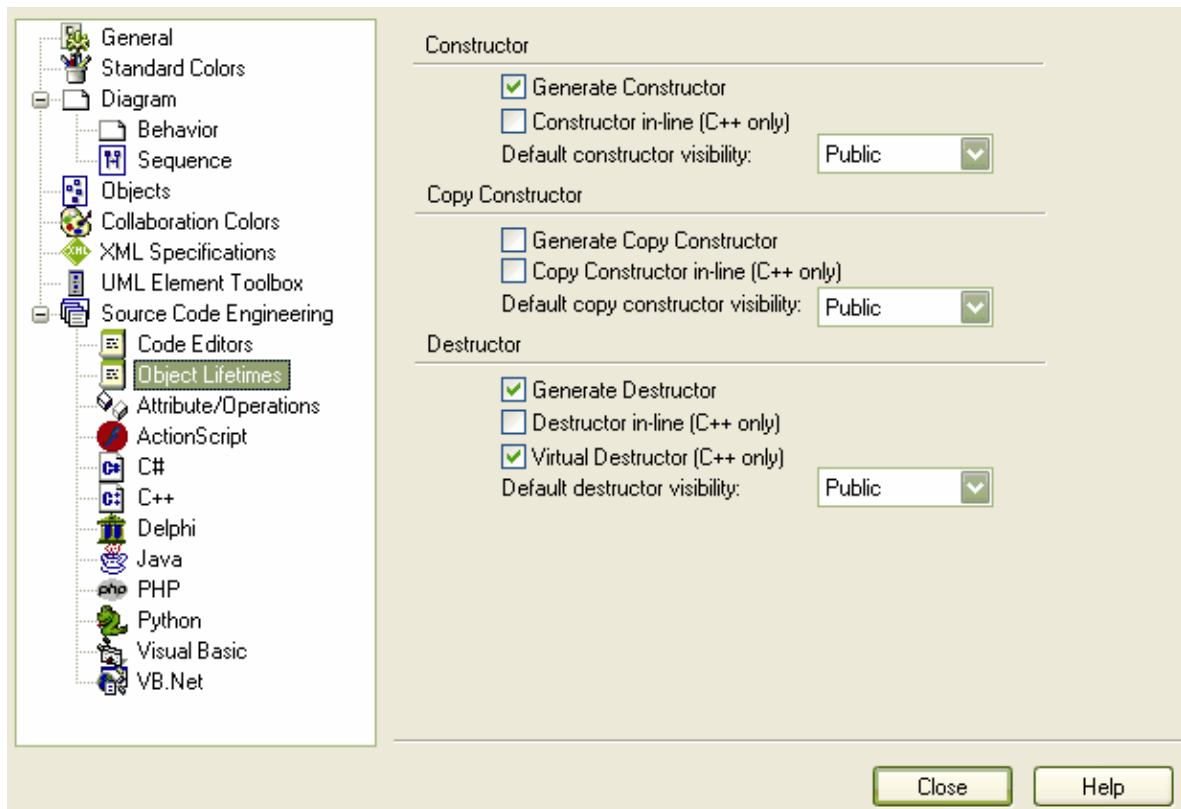


10.3.1.3 Options - Object Lifetimes

This set of options allows you to configure:

- Constructor details when generating code
- Whether to create a copy constructor
- Destructor details

This option is accessed via the *Source Code Engineering* section of the *Local Options* dialog (*Tools / Options*).



10.3.1.4 Options - Attribute/Operations

This set of options allows you to configure:

- The default name generated from imported attributes
- Generate methods for implemented interfaces
- Delete model attributes not included in the code during reverse synchronization
- Delete model methods not included in the code during reverse synchronization
- Delete code from features contained in the model during forward synchronization.
- Delete model associations and aggregations that correspond to attributes not included in the code during reverse synchronization.
- Whether to the bodies of methods will included and be saved in the EA model when reverse engineering.

This option is accessed via the *Source Code Engineering* section of the *Local Options* dialog (*Tools / Options*).

Attribute Specifications

Default name for associated attrib:

On reverse synch, delete model attributes not in code

On reverse synch, delete model associations not in code

Operation Specifications

Generate methods for implemented interfaces

On reverse synch, delete model methods not in code

Include method bodies in model when reverse engineering

Options

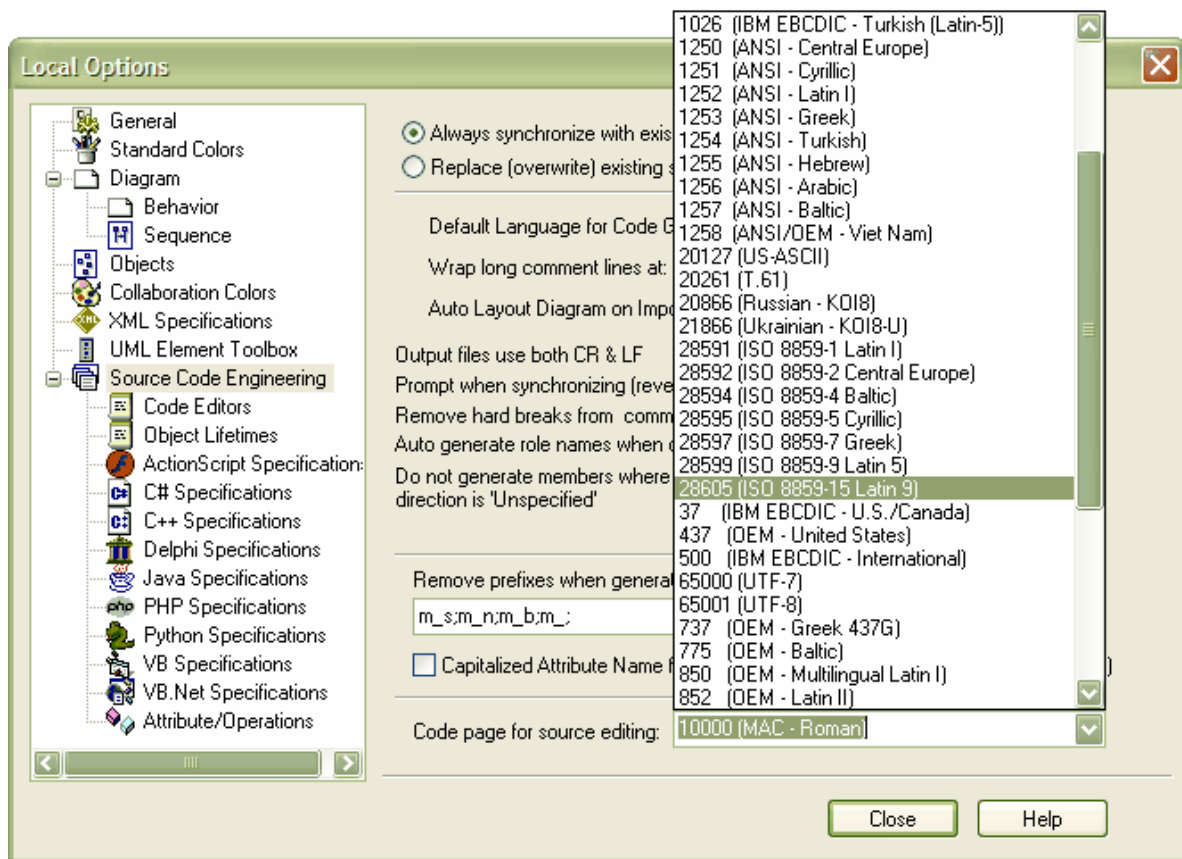
After save, re-select edited item

On forward synch, prompt to delete code features not in model

10.3.1.5 Code Page for Source Editing

The Unicode version of EA allows the user to define the Unicode character set for code generation. To set the Unicode character set use the following instructions:

1. Open the Local Options dialog by selecting *Tools | Options*.
2. Open the *Source Code Engineering* item and select the *Code page for source editing* drop down menu.



3. Select the appropriate Unicode character set.

Note: This feature is only applicable to the Unicode version of EA.

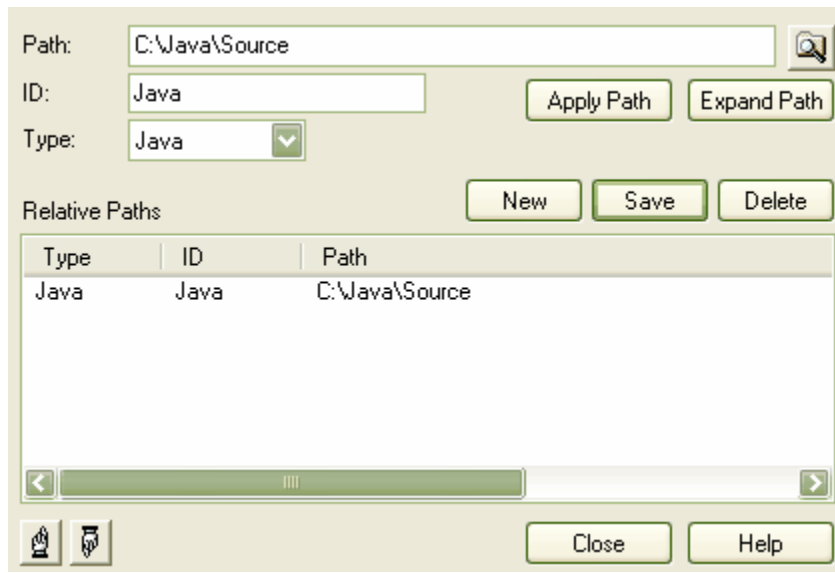
10.3.2 Local Paths

In many cases, a team of developers may be working on the same Enterprise Architect model. Often, each developer will store their version of the source code in their local file system, but not always at the same location as their fellow developers.

Local paths take a bit of setting up, but if you want to work collaboratively on source and model concurrently, the effort is well worthwhile.

For example: developer A stores their .java files in a 'C:\Java\Source' directory, while developer B stores theirs in 'D:\Source'. Meanwhile, both developers want to generate and reverse engineer into the same EA model located on a shared network drive (or replicated).

To handle this scenario, EA lets you define local paths for each EA user. In the above scenario developer A may define a local path of `JAVA_SOURCE = "C:\Java\Source"` - so that all classes generated and stored in the EA project will be stored as `%JAVA_SOURCE%\<xxx.java>`. Developer B now defines a `JAVA_SOURCE` local path as `D:\Source`. Now, EA will store all java files in these directories as `%JAVA_SOURCE%\<filename>`, and on each developer's machine, the filename is expanded to the correct local version.



To open the *Local Paths* dialog, from the main menu select the *Settings | Local Paths* command.

10.3.3 Local Path Dialog

The *Local Paths* dialog allows you to set up local paths for a single user on a particular machine. For a description of what Local Paths are used for, see [Local paths](#).

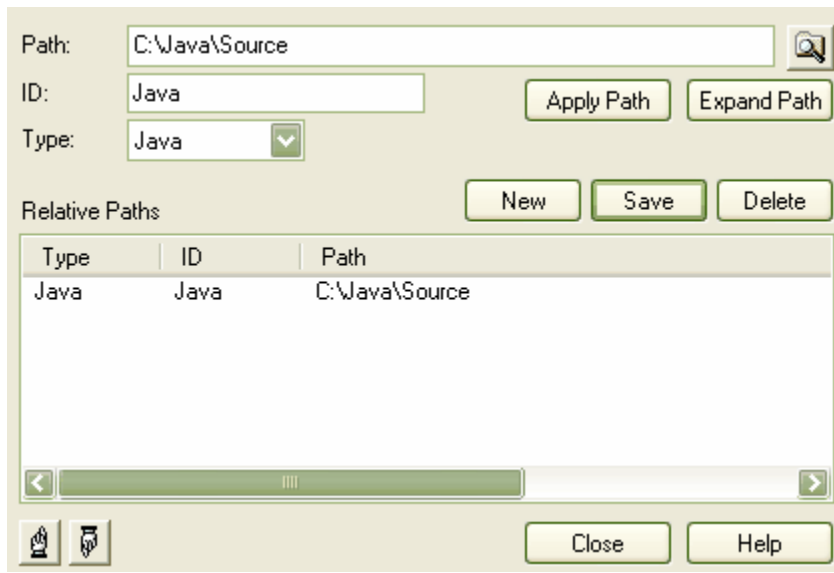
To open the *Local Paths* dialog, from the main menu select the *Settings | Local Paths* command.

The *Local Paths* dialog allows you to define:

- *Path* - this is the local directory in the file system (eg. d:\java\source)
- *ID* - this is the shared ID which will be substituted for the Local Path (eg. JAVA_SRC)
- *Type* - this is the language type- (eg. Java)

And also to:

- *Apply Path*. Select a path and press this to update any existing paths in the model (with full path names) to the shared relative path name (so "d:\java\source\main.java" may become "%JAVA_SRC%\main.java)
- *Expand Path*. The opposite of the item above. This lets you remove a relative path and substitute the full path name.
- Using the two above items you can update and change existing paths.



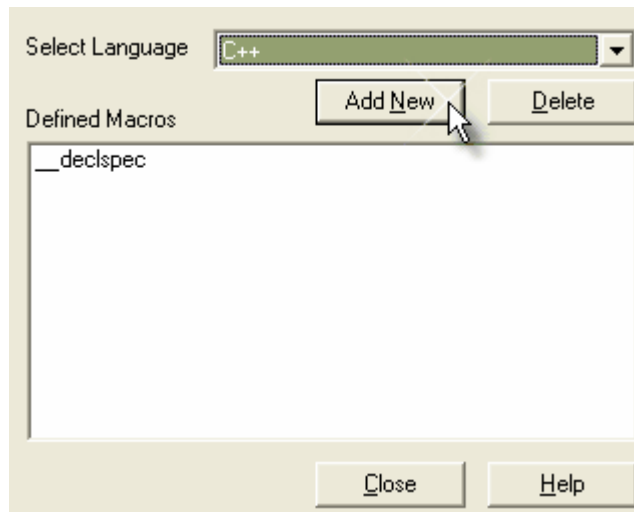
10.3.4 Language Macros

When reverse engineering a language like C++, often there are pre-processor directives scattered throughout the code. This may make code management easier - but can hamper parsing of the underlying C++ language.

To help remedy this, you may define any number of MACRO definitions, which will be ignored during the parsing phase of the reverse engineering. It is still preferable, if you have the facility, to pre-process the code using the appropriate compiler first - this way, complex macro definitions and defines are expanded out and may be readily parsed. If you don't have this facility, then this option provides a convenient substitute.

To Define a MACRO

1. Select *Preprocessor Macros* from the *Settings* menu.
2. On the *Language Macros* dialog, press *Add New*.
3. Enter details for your MACRO.
4. Press *OK*.



Macros Embedded within Declarations

Macros are sometimes used within the declaration of classes and operations as in the following examples:

```
class __declspec Foo
{
    int __declspec Bar(int p);
};
```

If declspec is defined as a C++ macro as outlined above, the imported class and operation will contain a tagged value named "DeclMacro1" with value "__declspec". (Subsequent macros would be defined as "DeclMacro2", "DeclMacro3" etc). During forward engineering, these tagged values are used to regenerate the macros in code.

Defining Complex Macros

It is sometimes useful to define rules for complex macros that may span multiple lines. EA will ignore the entire code section defined by the rule. Such macros can be defined in EA as in the following example:

```
BEGIN_INTERFACE_PART ^ END_INTERFACE_PART
```

where the ^ symbol represents the body of the macro. Note : The spaces surrounding the ^ symbol are required.

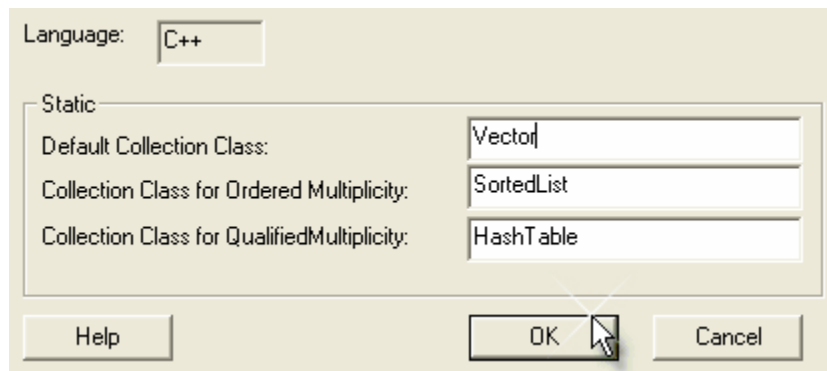
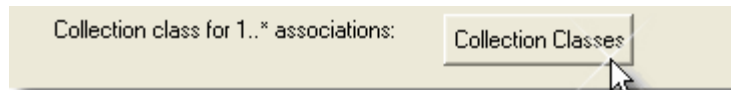
10.3.5 Setting Collection Classes

EA lets you define collection classes for generating code from association links where the target role has a multiplicity setting greater than 1.

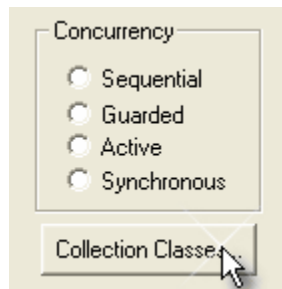
There are two options for doing this:

1. In the *Local Options* dialog (*Tools | Options*), on each page of the *Generation* section is a *Collection Classes* button. This opens a dialog where you can define:
 - The default collection class for 1..* roles
 - The ordered collection class to use for 1..* roles

- The qualified collection class to use for 1..* roles



2. A *Collection Classes* button can also be found on the *Detail* tab of the *Class Properties* dialog (accessible from the right click context menu of any class). This opens the same dialog as above, where you can define:
 - The default collection class for 1..* roles *for this class only*
 - The ordered collection class to use for 1..* roles *for this class only*
 - The qualified collection class to use for 1..* roles *for this class only*



When EA generates code for a link that has a multiplicity role >1, the collection class is calculated as follows:

1. If the *Qualifier* is set use the qualified collection:
 - for the class if set
 - else use the code language qualified collection
2. If the *Order* option is set use the ordered collection:
 - for the class if set
 - else use the code language ordered collection
3. Else use the default collection:
 - for the class if set
 - else use the code language default collection

Note: You can include the marker "#TYPE#" in the collection name, and EA will replace this with the name of the class being collected at source generation time (eg. Vector<#TYPE#> would become Vector<foo>)

Additionally, there is a *Member Type* field on both the *Source Role* and *Target Role* tabs on the *Association Property* dialog (accessible from the right click context menu of any association). If you set this, the value entered will override all the above options. The example below shows a defined PersonList - when code is generated, because this has a *Multiplicity* > 1 and the *Member Type* is defined, the variable created will be of type PersonList.

The screenshot shows the 'Source Role' tab of the 'Association Property' dialog. The 'Login Role' dropdown is set to 'm_PersonList'. The 'Role Notes' text area contains 'A list of staff in order'. Under the 'Multiplicity' section, the dropdown is set to '1..*', and the 'Ordered' checkbox is checked. The 'Member Type' text field contains 'PersonList'. Other dropdowns include 'Containment' (Unspecified), 'Access' (Protected), 'Aggregation' (none), 'Target Scope' (instance), 'Navigability' (Unspecified), and 'Changeable' (none). There are 'OK', 'Cancel', and 'Help' buttons at the bottom.

10.3.6 Language Options

You can set up various options for how EA will handle a particular language when generating code. These options are accessible on the *Local Options* dialog (select *Options* from the *Tools* menu). From the *Source Code Engineering* section, select the required language. The following topics outline the options available for each language.

See the code generation options for:

- [ActionScript](#)
- [C#](#)
- [C++](#)
- [Delphi](#)
- [Delphi Properties](#)
- [Java](#)
- [PHP](#)
- [Python](#)
- [Visual Basic](#)

- [VB.Net](#)
- [Reset Options](#)

10.3.6.1 Options - ActionScript

Configure options for ActionScript generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *ActionScript* in the *Source Code Engineering* section. The options you can specify include:

- The default file extension(s).
- Default source directory .
- You may also set the editor for ActionScript Code.
- Default ActionScript Version to generate.

| ActionScript Specifications | |
|---|-----|
| <input type="checkbox"/> Disable Language | |
| [-] Options for the current model | |
| Default Version | 2.0 |
| Default Extension | .as |
| [-] Options for the current user | |
| Default Source Directory | |
| Editor | |
| Collection class for 1..* associations: | |
| <input type="button" value="Collection Classes"/> | |

10.3.6.2 Options - C

Configure options for C code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *C* in the *Source Code Engineering* section. The options you can specify include:

- The default file extension(s).
- Default source directory.

C Specifications

Disable Language

| Options for the current model | |
|-------------------------------|----|
| Header Extension | .h |
| Source Extension | .c |

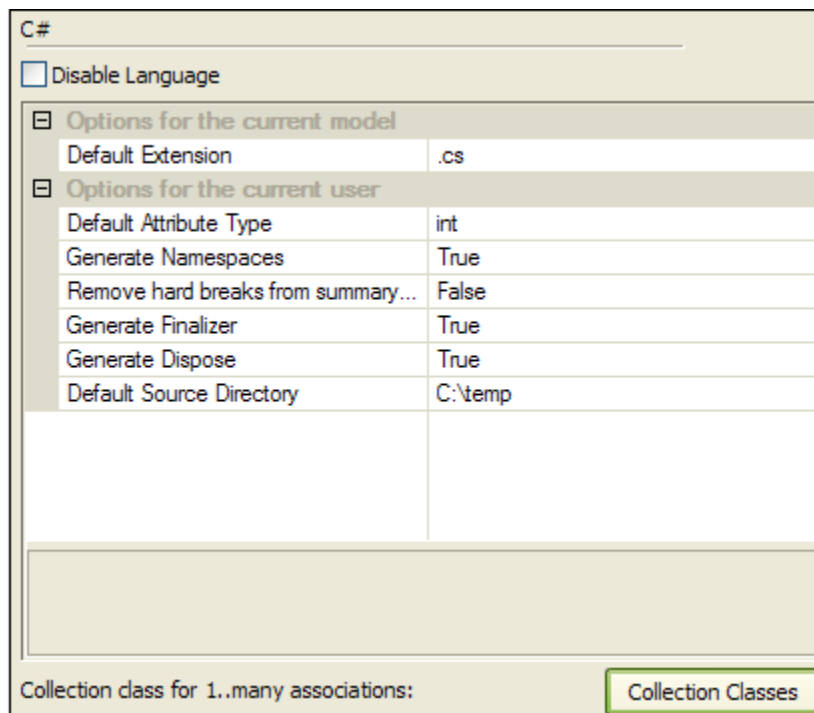
| Options for the current user | |
|--|-------|
| Default Attribute Type | int |
| Method Notes in Header | False |
| Method Notes in Implementation | True |
| Synchronise Notes in Generation | True |
| Synchronise Implementation file in ... | True |
| Default Source Directory | |
| Import File Extensions | .h; |

Collection class for 1..* associations: Collection Classes

10.3.6.3 Options - C#

Configure options for C# code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *C#* in the *Source Code Engineering* section. The options you can specify include:

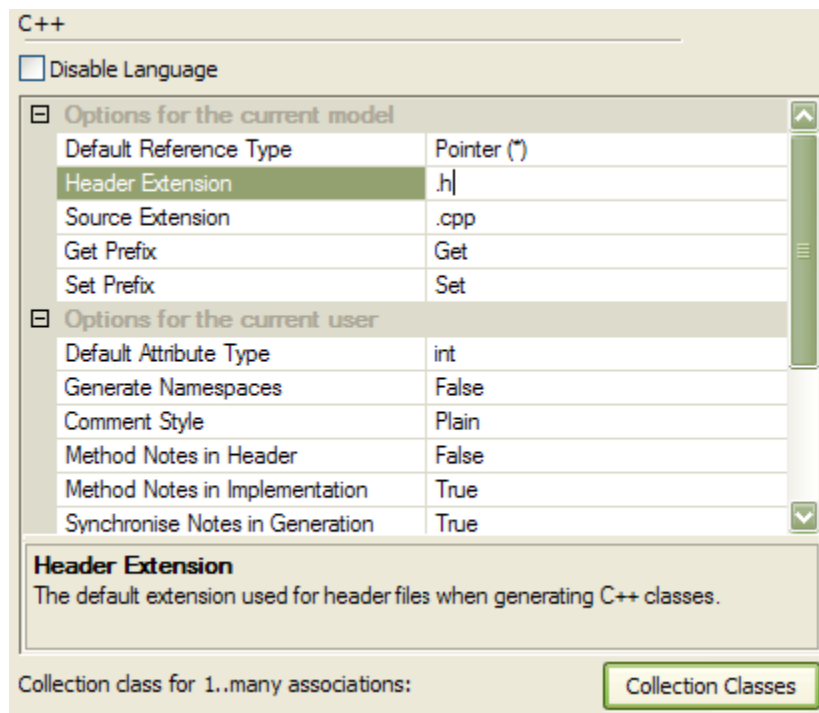
- The default file extension
- The default 'Get' prefix
- The default 'Set' prefix
- You may also set a default directory for opening and saving C# source code



10.3.6.4 Options - C++

Configure options for C++ code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *C++* in the *Source Code Engineering* section. The options you can specify include:

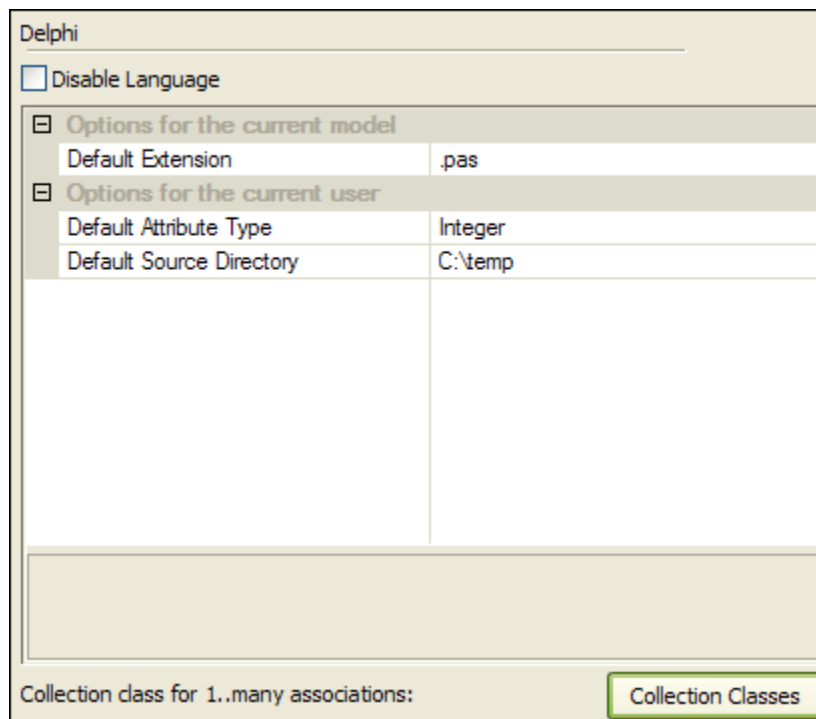
- The version of C++ to generate. This controls the set of templates used and how properties are created.
- The default reference type used when a type is specified as by reference.
- The default file extension(s).
- Default Get/Set prefixes.
- Default source directory.



10.3.6.5 Options - Delphi

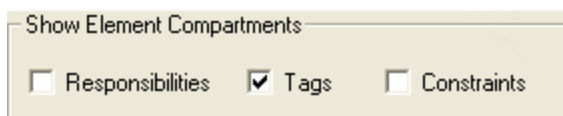
Configure options for Delphi code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *Delphi* in the *Source Code Engineering* section. The options you can specify include:

- Default file extension.
- Default source directory.
- You may also set a default directory for opening and saving Delphi source code.



10.3.6.5.1 Delphi Properties

Enterprise Architect has comprehensive support for Delphi properties. These are implemented as tagged values - with a specialized property editor to help create and modify class properties. The class image below illustrates the appearance of a Delphi class that has had properties added to it. These are stored as tagged values, and by using the *Specify Feature Visibility* element context menu option, you can display the 'tags' compartment which contains the properties. Imported Delphi classes with properties will automatically have this feature made visible for your convenience.

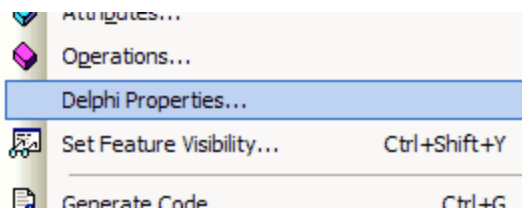


Note: when you use the *Create Property* dialog from the *Attribute* screen, EA will generate a pair of get and set functions, together with the required property definition as tagged values. You may manually edit these tagged values if required.

| TTestClass2 | |
|-------------|---|
| - | FTestField: Integer |
| - | m_Name: String |
| + | «property get» GetName(): String |
| + | «property set» SetName(String) |
| ..* | PrivateProcedure Test(Integer) |
| - | «function» PrivateFunctionTest(): string |
| - | «property get» GetPublicPropertyTest2(): string |
| - | «property set» SetPublicPropertyTest2(string) |
| # | ProtectedProcedure Test(WideString) |
| # | «function» ProtectedFunctionTest(): Boolean |
| # | «property get» GetPublishedPropertyTest4(): Extended |
| # | «property set» SetPublishedPropertyTest4(Extended) |
| + | «constructor» TestCreate(TObject) |
| + | «destructor» TestDestroy() |
| + | PublicProcedure Test(Double) |
| + | «function» PublicFunctionTest(): Word |
| tags | |
| property: | +PublicPropertyTest1:Integer read m_Name write GetName default 'Joe' |
| property: | +PublicPropertyTest2:string read GetPublicPropertyTest2 write SetPublicPropertyTest2 default 13 |
| property: | ^PublishedPropertyTest3:Integer read FTestField write FTestField |
| property: | ^PublishedPropertyTest4:Extended read GetPublishedPropertyTest4 write SetPublishedPropertyTest4 |

To manually activate the property editor

1. Ensure the class you have selected has the code generation language set to Delphi
2. Right click on the class and select *Delphi Properties* to open the editor



The Delphi Properties editor allows you to build properties in a simple and straightforward manner. From here you can:

- Change the name and scope (only Public and Published are currently supported).
- Change the property type(drop down list includes all defined classes in the project).
- Set the read and write information ... drop down list has all the attributes and operations from the current class. You may also enter free text.
- Set stored to True or False.
- Set the Implements information.
- Set the Default value is one exists.

Property Details

Name: Published

Type:

Read:

Write:

Stored:

Implements:

Default:

Definition:

Defined Properties

- Property Details
- ^Active:Boolean read FActive write SetActive default false
- ^Text:TStringList read FText write SetText
- ^Interval:Integer read GetInterval write SetInterval default 100
- ^Repetitions:integer read FR repetitions write SetRepetitions default 0
- ^Transparent:boolean read FTtransparent write SetTransparent default true
- ^WordWrap:boolean read GetWordWrap write SetWordWrap
- ^ScrollPixels:integer read FScrollPixels write SetScrollPixels default 1
- ^OnClick
- ^OnDbClick

Note: Public properties are displayed with a '+' symbol prefix and published with a '^'.

Note: When creating a property in the property wizard accessed through the Attributes dialog, you can set the scope to be Published if the property type is Delphi - see example below.

Property Details

Name:

Getter:

Setter:

Stereotype: Published

Get Scope: Set Scope:

Limitations

- Only Public and Published supported.
- If you change the name of a property and forward engineer, a new property is added, but the old one must be manually deleted from the source file.

10.3.6.6 Options - Java

Configure options for Java code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *Java* in the *Source Code Engineering* section. The options you can specify include:

- The default file extension
- The default 'Get' prefix
- The default 'Set' prefix
- You may also set a default directory for opening and saving Java source code

| Options for the current model | |
|-------------------------------|------|
| Default Extension | java |
| Get Prefix | get |
| Set Prefix | set |
| Default Collection Class | |

| Options for the current user | |
|------------------------------|-----|
| Default Attribute Type | int |
| Default Source Directory | |

Collection class for 1..many associations:

10.3.6.7 Options - PHP

Configure options for Java code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *PHP* in the *Source Code Engineering* section. The options you can specify include:

- The default source extension - Specify the extension to be used when creating files for PHP source.
- The default import extension - A semi-colon separated list of extensions to look at when doing a [directory code import](#) for PHP.
- The default PHP version - Version of PHP to generate.
- You may also set a default directory for opening and saving PHP source code

The screenshot shows the 'PHP' configuration dialog. At the top, there is a checkbox for 'Disable Language'. Below it, there are two expandable sections: 'Options for the current model' and 'Options for the current user'. The 'Options for the current model' section contains a table with the following data:

| | |
|-------------------|------|
| Default Extension | .php |
| Get Prefix | get |
| Set Prefix | set |
| Default Version | 4.0 |

The 'Options for the current user' section contains a table with the following data:

| | |
|--------------------------|------------------|
| Default Source Directory | C:\temp |
| Import File Extensions | .php;.php4;.inc; |
| Editor | |

At the bottom of the dialog, there is a text field labeled 'Collection class for 1..many associations:' and a button labeled 'Collection Classes'.

10.3.6.8 Options - Python

Configure options for Python generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *Python* in the *Source Code Engineering* section. The options you can specify include:

- The default file extension(s)
- Default source directory
- You may also set Editor for Python Code.

The screenshot shows the 'Python' configuration dialog. At the top, there is a checkbox for 'Disable Language'. Below it, there are two expandable sections: 'Options for the current model' and 'Options for the current user'. The 'Options for the current model' section contains a table with the following data:

| | |
|-------------------|-----|
| Default Extension | .py |
|-------------------|-----|

The 'Options for the current user' section contains a table with the following data:

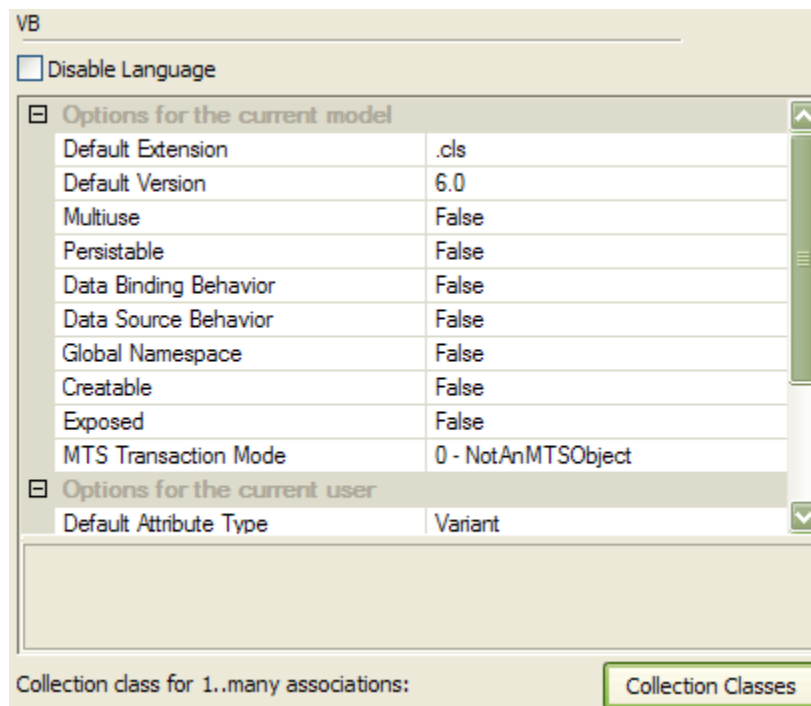
| | |
|--------------------------|--|
| Default Source Directory | |
| Editor | |

At the bottom of the dialog, there is a text field labeled 'Collection class for 1..many associations:' and a button labeled 'Collection Classes'.

10.3.6.9 Options - Visual Basic

Configure options for Visual Basic code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *Visual Basic* in the *Source Code Engineering* section. The options you can specify include:

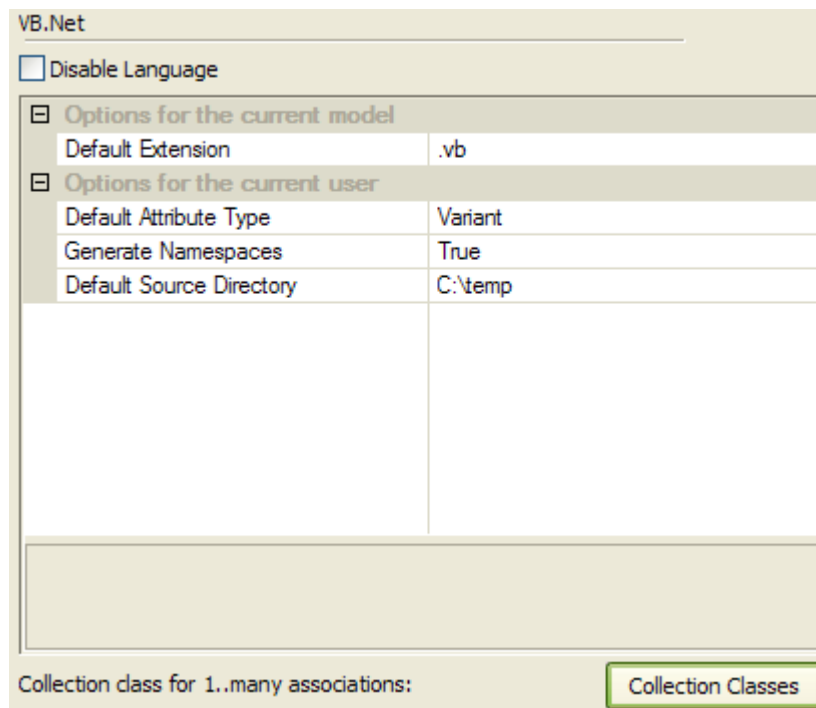
- The default file extension when reading/writing
- Default Visual Basic version
- MTS transaction mode for MTS objects
- Multi use (true or false)
- Persistable
- Data binding
- Global namespace
- Exposed
- Data source behavior
- Creatable



10.3.6.10 Options - VB.Net

Configure options for VB.Net code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *VB.Net* in the *Source Code Engineering* section. The options you can specify include:

- Default file extension
- Default source directory



10.3.6.11 Reset Options

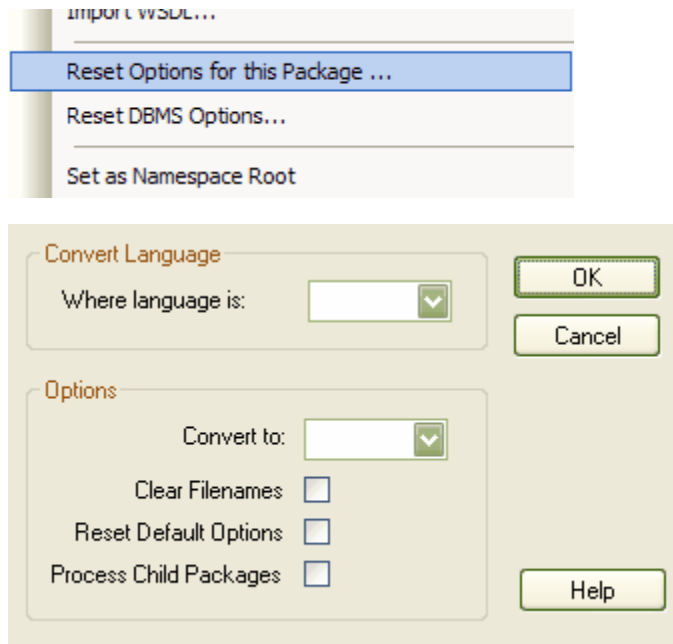
EA stores some of the options for a class when it is first created - and some are global. For example \$LinkClass is stored when you first create the class, so it won't automatically pick up the global change in the *Options* screen in existing classes. You need to modify the options for the existing class.

To Modify Options for a Single Class

1. Right click on the class to change and select *Generate Code* from the context menu.
2. Press *Advanced* on the Generate Code dialog.
3. In the Object Options dialog, select *Attributes/Operations*.
4. Change options and press *Close* to apply changes.

To Modify Options for All Classes Within a Package

1. Right click on the package Project Browser to view the context menu.
2. From the *Code Engineering* submenu, select *Reset Code Generation Options for Package*.
3. The *Manage Code Generation* dialog provides the ability to reset some defaults for each existing class.
4. Press *OK* to apply changes.



10.4 Code Template Framework

The **Code Template Framework** (CTF) is used during the forward engineering of UML models.

The CTF allows:

The Code Template Framework allows:

- Generation of source code from UML models.
- Customization of the way in which EA generates source code.
- Forward engineering of languages not specifically supported by EA.

The CTF consists of:

The Code Template Framework consists of:

- Default [Code Templates](#) which are built into EA for forward engineering supported languages.
- A [Code Template Editor](#) for creating and maintaining User-defined Code Templates.

See Also

- [Synchronizing Code](#)

10.4.1 Code Templates

EA's code templates specify the transformation from UML elements to the various parts of a given programming language. The templates are written as plain text with a syntax that shares some aspects of both mark-up languages and scripting languages. A simple example of a template used by EA is the "Class template". It is used to generate source code from a UML class:

```
%ClassNotes%
%ClassDeclaration%
%ClassBody%
```

The above template simply refers to three other templates, namely ClassNotes, ClassDeclaration and ClassBody. The enclosing percent (%) signs indicate a "macro." Code Templates consist of various types of macros, each resulting in a substitution in the generated output. For a language such as C++, the result of processing the above template might be:

```
/**
 * This is an example class note generated using code templates
 * @author Sparx Systems
 */
class ClassA : public ClassB
{
    ...
}
```

See Also

- [Base Templates](#)
- [Custom Templates](#)
- [Execution of Code Templates](#)
- [Code Template Syntax](#)

10.4.1.1 Base Templates

The CTF consists of a number of base templates. Each base template transforms particular aspects of the UML to corresponding parts of object-oriented languages. The following tables lists and briefly describe the base templates used in the CTF. The second table lists templates used for generating code for languages which have separate interface and implementation sections.

| Template | Description |
|-----------------------|---|
| Attribute | Top-Level template to generate member variables from UML attributes |
| Attribute Declaration | Used by Attribute template to generate member variable declaration |
| Attribute Notes | Used by Attribute template to generate member variable notes |
| Class | Top-Level template for generating classes from UML classes |
| Class Base | Used by Class template to generate a base class name in the inheritance list of a derived class where the base class doesn't exist in the model |
| Class Body | Used by Class template to generate the body of a class |
| Class Declaration | Used by Class template to generate the declaration of a class |
| Class Interface | Used by Class template to generate an interface name in the inheritance list of a derived class where the interface doesn't exist in the model |
| Class Notes | Used by Class template to generate the class notes |
| File | Top-Level template for generating the source file. For languages such as C++ this corresponds to the header file. |
| Import Section | Used in the File template to generate external dependencies |
| Linked Attribute | Top-Level template for generating attributes derived from UML associations. |

| | |
|------------------------------|--|
| Linked Attribute Notes | Used by Linked Attribute template to generate the attribute notes |
| Linked Attribute Declaration | Used by Linked Attribute template to generate the attribute declaration |
| Linked Class Base | Used by Class template to generate a base class name in the inheritance list of a derived class for a class element in the model that is a parent of the current class. |
| Linked Class Interface | Used by Class template to generate an interface name in the inheritance list of a derived class for an interface element in the model that is a parent of the current class |
| Namespace | Top-Level template for generating Namespaces from UML packages. (Although not all languages have namespaces, this template can be used to generate an equivalent construct, such as packages in Java.) |
| Namespace Body | Used by Namespace template to generate the body of a namespace |
| Namespace Declaration | Used by Namespace template to generate the namespace declaration |
| Operation | Top-Level template for generating operations from a UML class's operations |
| Operation Body | Used by Operation template to generate the body of a UML operations |
| Operation Declaration | Used by Operation template to generate the operation declaration |
| Operation Notes | Used by Operation template to generate documentation for an operation |
| Parameter | Used by Operation Declaration template to generate parameters. |

| Template | Description |
|----------------------------|--|
| Class Impl | Top-level template for generating the implementation of a class |
| Class Body Impl | Used by Class Impl to generate the implementation of class members. |
| File Impl | Top-Level template for generating the implementation file. |
| File Notes Impl | Used by the File Impl template to generate notes in the source file |
| Import Section Impl | Used by the File Impl template to generate external dependencies |
| Operation Impl | Top-Level template for generating operations from a UML class's operations |
| Operation Body Impl | Used by Operation template to generate the body of a UML operations |
| Operation Declaration Impl | Used by Operation template to generate the operation declaration |
| Operation Notes Impl | Used by Operation template to generate documentation for an operation |

The base templates form a hierarchy, which varies slightly across different programming languages. A typical template hierarchy relevant to a language like C# or Java (which do not have header files) is shown below. In this figure we have modeled the templates as classes (in reality they are just plain text). This hierarchy would

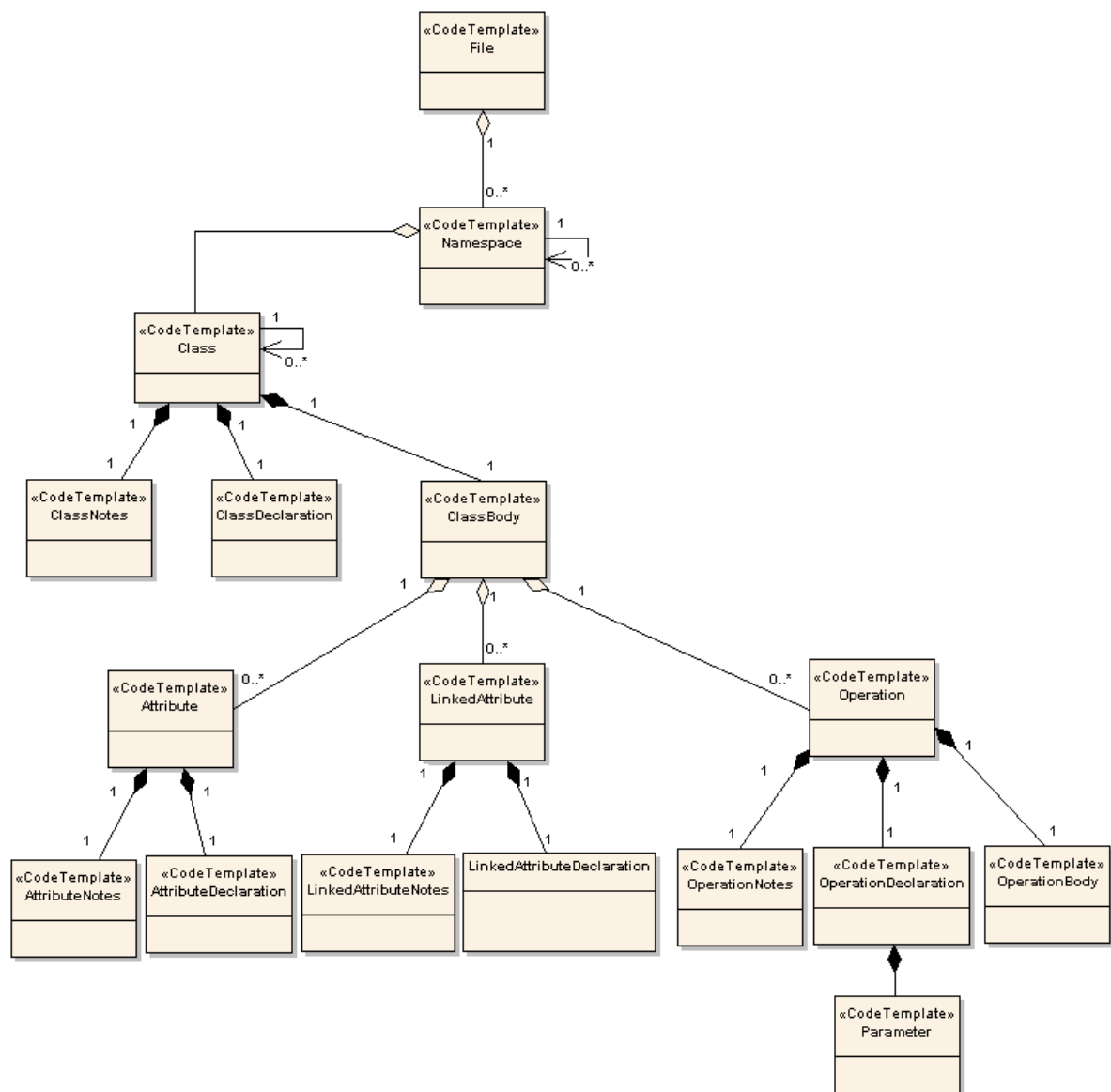
be slightly more complicated for languages like C++ and Delphi, which have separate implementation templates.

Each of the base templates need to be specialised to be of use in code engineering. In particular, each template is specialised for the supported languages (or "products"). For example, there is a ClassBody template defined for C++, and another for C#, and yet another for Java and so on. By specialising the templates, we can tailor the code generated for the corresponding UML entity.

Once the base templates are specialised for a given language, they may then be further specialised based on:

- A class's stereotype
- A feature's stereotype (where the feature can be an operation or attribute)

This type of specialisation allows a C# operation for example, which is stereotyped as <<property>>, to have a different OperationBody template than an ordinary operation. The OperationBody template may then be specialised further, based on the class stereotype.



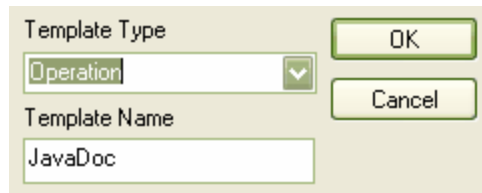
Above: Class Model showing the hierarchy of Code Generation templates for a language such as C# or Java. The aggregation links denote references between templates.

10.4.1.2 Custom Templates

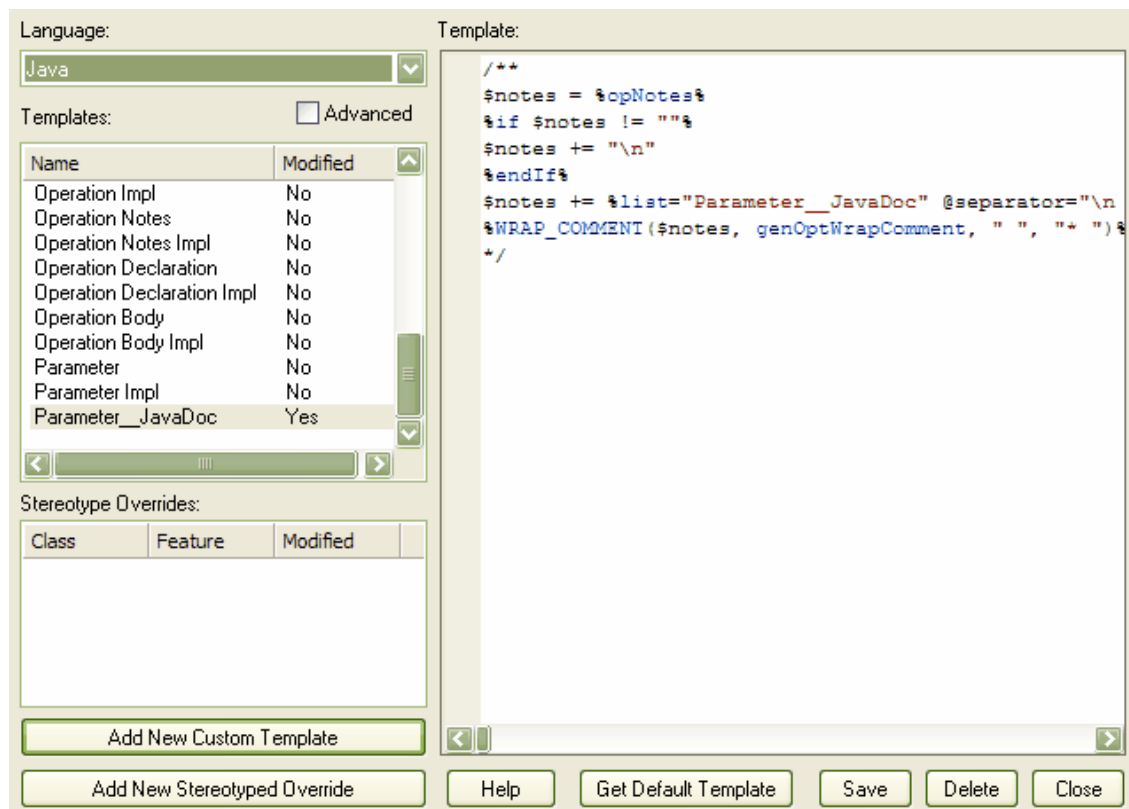
Custom templates give users the option to generate an element in multiple different ways. EA Allows you to define custom templates that are associated with given elements and call these templates from existing templates. You can even add stereotype overrides to your custom templates.

For example, you may want to list all of your parameters and their notes in your method notes.
To create a new custom template:

1. Select Code Generation Templates from the Settings menu to open the Code Templates Editor or use the **CTRL + Shift + P**.
2. Select the appropriate language, from the *Language* list.
3. Press **Add New Custom Template**.
4. This will open the Create New Custom Template dialog and choose the type from the Template Type list. The elements currently supported are:
 - Attribute
 - Class
 - Operation
 - Parameter



5. Give the template an appropriate name, then press the **OK** button.
6. The new template will appear in the templates list and be marked as modified. It will be called <Element Type>__<Template Name>.
7. Select the appropriate template from the list of templates and create a template which fulfils your requirements.



8. Press the **Save** button. This stores the new stereotyped template in the .EAP file. The template is now available from the list of templates and via direct substitution for use

10.4.1.3 Execution of Code Templates

A reference to a template (such as the %ClassNotes% macro, from our previous example) results in the execution of that template.

Each template is designed for use with a particular element. For example the ClassNotes template is to be used with UML Class elements.

The element that is currently being generated is said to be "in scope". If the element in scope is stereotyped EA looks for a template that has been defined for that stereotype. If a match is found, the specialised template is executed. Otherwise the default implementation of the base template is used.

Templates are processed sequentially, line by line, replacing each macro with its underlying text value from the model.

10.4.1.4 Code Template Syntax

Code Templates are written as plain text, using EA's code template editor (see [The Code Template Editor](#)). The template syntax centres around three basic constructs:

- [Literal Text](#)
- [Macros](#)

- [Variables](#)

Templates may contain any or all of these constructs, which are described in the following sections.

10.4.1.4.1 Literal Text

All text within a given template that is not part of a macro or a variable definition/reference, is considered literal text. With the exception of blank lines, which are ignored, literal text is directly substituted from the template into the generated code.

Consider the following excerpt from the java Class Declaration template:

```
%PI=" "%
%CONVERT_SCOPE(classScope)%

%classStereotype=="static"? "static" : ""%
%classStereotype=="final"? "final" : ""%
%classStereotype=="staticfinal"? "staticfinal" : ""%
%classAbstract=="T"? "abstract" : ""%
%PI=" "%
class %className%$bases
```

On the final line, the word "class ", including the subsequent space, would be treated as literal text and thus reproduced in the output. The blank line following the `CONVERT_SCOPE` macro however, would have no effect on the output.

The %, \$ and " characters have special meaning in the template syntax and cannot always be used as literal text. If these characters need to be generated from within the templates, they can be safely reproduced using the following direct substitution macros:

| Macro | Description |
|-------|---------------------------------|
| %dl% | produces a literal \$ character |
| %pc% | produces a literal % character |
| %qt% | produces a literal " character |

10.4.1.4.2 Macros

Macros provide access to element fields within the UML model and are also used to structure the generated output. All macros are enclosed within percent (%) signs. The CTF contains five basic types of macros:

- [Template substitution macros](#)
- [Field substitution macros](#)
- [Tagged value substitution macros](#)
- [Control macros](#)
- [Function macros](#)

In general, macros (including the % delimiters) are substituted with literal text in the output. For example consider the following snippet from the Class Declaration template:

```
... class %className% ...
```

The field substitution macro, `%className%`, would result in the current class' name being substituted in the output. So if the class being generated was named "Foo", the output would be:

```
...classFoo...
```


Each of the basic macro types are explained in the following topics.

10.4.1.4.2.1 Template Substitution Macros

The template substitution macros correspond to the templates listed in [Base Templates](#). These macros result in the execution of the named template. By convention, template macros are named according to Pascal casing.

Structure: %<TemplateName>%

where <TemplateName> may be one of the templates listed below.

When a template is referenced from within another template, it is generated with respect to the elements currently in scope. The specific template is selected based on the stereotypes of the elements in scope.

As noted previously, there is an implicit hierarchy among the various templates. Some care should be taken in order to preserve a sensible hierarchy of template references. For example, it does not make sense to use the %ClassInherits% macro within any of the attribute or operation templates. Conversely, the Operation and Attribute templates are designed for use within the ClassBody template.

The CTF contains the following template substitution macros:

- | | | |
|------------------------|------------------------------|----------------------------|
| • AttributeDeclaration | • ClassParameter | • NamespaceBody |
| • AttributeNotes | • File | • NamespaceDeclaration |
| • Attribute | • FileImpl | • NamespaceImpl |
| • Class | • ImportSection | • Operation |
| • ClassImpl | • ImportSectionImpl | • OperationBody |
| • ClassBase | • InnerClass | • OperationBodyImpl |
| • ClassBody | • InnerClassImpl | • OperationDeclaration |
| • ClassBodyImpl | • LinkedAttribute | • OperationDeclarationImpl |
| • ClassDeclaration | • LinkedAttributeNotes | • OperationImpl |
| • ClassDeclarationImpl | • LinkedAttributeDeclaration | • OperationNotes |
| • ClassInherits | • LinkedClassBase | • Parameter |
| • ClassInterface | • LinkedClassInterface | |
| • ClassNotes | • Namespace | |

10.4.1.4.2.2 Field Substitution Macros

The field substitution macros provide access to data in the model. In particular, they are used to access data fields from:

- Packages
- Classes
- Attributes
- Operations
- Parameters

Field substitution macros are named according to Camel casing. By convention, the macro is prefixed with an abbreviated form of the corresponding model element. For example, attribute-related macros begin with "att", as in the %attName% macro, to access the name of the attribute in scope.

The following table lists each of the field substitution macros with a description of the result. Note that macros which represent checkboxes return a value of "T" if the box is checked. Otherwise the value is empty.

| Macro Name | Description |
|------------------------|--|
| attAlias | Attribute dialog: "Alias" |
| attAllowDuplicates | Attribute Detail dialog: "Allow Duplicates" checkbox |
| attCollection | Attribute Detail dialog: "Attribute is a Collection" checkbox |
| attConst | Attribute dialog: "Const" checkbox |
| attContainerType | Attribute Detail dialog: "Container Type" |
| attContainment | Attribute dialog: "Containment" |
| attDerived | Attribute dialog: "Derived" checkbox |
| attGUID | The unique GUID for the current attribute |
| attInitial | Attribute dialog: "Initial" |
| attLowerBound | Attribute Detail dialog: "Lower Bound" |
| attName | Attribute dialog: "Name" |
| attNotes | Attribute dialog: "Notes" |
| attOrderedMultiplicity | Attribute Detail dialog: "Ordered Multiplicity" checkbox |
| attProperty | Attribute dialog: "Property" checkbox |
| attQualType | The attribute type qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). If the attribute classifier has not been set, is equivalent to the attType macro. |
| attScope | Attribute dialog: "Scope" |
| attStatic | Attribute dialog: "Static" checkbox |
| attStereotype | Attribute dialog: "Stereotype" |
| attType | Attribute dialog: "Type" |
| attUpperBound | Attribute Detail dialog: "Upper Bound" |
| attVolatile | Attribute Detail dialog: "Transient" checkbox |
| classAbstract | Class dialog: "Abstract" checkbox |
| classAlias | Class dialog: "Alias" |
| classArguments | Class Detail dialog: "C++ Templates: Arguments" |
| classAuthor | Class dialog: "Author" |
| classBaseName | Type Hierarchy dialog : "Class Name" (for use where no link exists between child and base class) |
| classComplexity | Class dialog: "Complexity" |
| classGUID | The unique GUID for the current class |
| classHasParent | True, if the class in scope has one or more base classes |
| classImports | Code Gen dialog: "Imports". For supported languages this also includes dependencies derived from associations |
| classIsActive | Class Advanced dialog: "Is Active" checkbox |
| classIsInstantiated | True, if the class is an instantiated template class |
| classIsLeaf | Class Advanced dialog: "Is Leaf" checkbox |
| classIsRoot | Class Advanced dialog: "Is Root" checkbox |
| classIsSpecification | Class Advanced dialog: "Is Specification" checkbox |
| classKeywords | Class dialog: "Keywords" |
| classLanguage | Class dialog: "Language" |
| classMacros | A space separated list of macros defined for the class |
| classMultiplicity | Class Advanced dialog: "Multiplicity" |
| className | Class dialog: "Name" |
| classNotes | Class dialog: "Note" |
| classParamDefault | Class Detail dialog |
| classParamName | Class Detail dialog |
| classParamType | Class Detail dialog |
| classPersistence | Class dialog: "Persistence" |
| classPhase | Class dialog: "Phase" |
| classQualName | The class name prefixed by its outer classes. Class names are separated by double colons (:). |

| Macro Name | Description |
|-----------------------------|--|
| classScope | Class dialog: "Scope" |
| classStereotype | Class dialog: "Stereotype" |
| classStatus | Class dialog: "Status" |
| classVersion | Class dialog: "Version" |
| connectorDestAggregation | Connector target role properties: "Aggregation" |
| connectorDestAccess | Connector target role properties: "Access" |
| connectorDestContainment | Connector target role properties: "Containment" |
| connectorDestElem* | For any "class" substitution macro that is supported, get the same information about the element at the target end of the connector. |
| connectorDestMemberType | Connector target role properties: "Member Type" |
| connectorDestMultiplicity | Connector target role properties: "Multiplicity" |
| connectorDestNotes | Connector target role properties: "Notes" |
| connectorDestOrdered | Connector target role properties: "Multiplicity is Ordered" |
| connectorDestRole | Connector target role properties: "Role" |
| connectorDestScope | Connector target role properties: "Target Scope" |
| connectorDestStereotype | Connector target role properties: "Stereotype" |
| connectorDirection | Connector properties: "Direction" |
| connectorEffect | StateFlow Constraints dialog: "Effect" |
| connectorGuard | Object Flow and StateFlow Constraints dialog: "Guard" |
| connectorGUID | The unique GUID for the current connector |
| connectorName | Connector properties: "Name" |
| connectorNotes | Connector properties: "Notes" |
| connectorSourceAggregation | Connector source role properties: "Aggregation" |
| connectorSourceAccess | Connector source role properties: "Access" |
| connectorSourceContainment | Connector source role properties: "Containment" |
| connectorSourceElem* | For any "class" substitution macro that is supported, get the same information about the element at the source end of the connector. |
| connectorSourceMemberType | Connector source role properties: "Member Type" |
| connectorSourceMultiplicity | Connector source role properties: "Multiplicity" |
| connectorSourceNotes | Connector source role properties: "Notes" |
| connectorSourceOrdered | Connector source role properties: "Multiplicity is Ordered" |
| connectorSourceRole | Connector source role properties: "Role" |
| connectorSourceScope | Connector source role properties: "Target Scope" |
| connectorSourceStereotype | Connector source role properties: "Stereotype" |
| connectorStereotype | Connector properties: "Stereotype" |
| connectorTrigger | StateFlow Constraints dialog: "Trigger" |
| connectorType | The connector type. eg. Association or Generalization. |
| connectorWeight | Object Flow Constraints dialog: "Weight" |
| eaDateTime | The current time with format: DD-MMM-YYYY HH:MM:SS AM/PM |
| eaGUID | A unique GUID for this generation |
| eaVersion | Program Version (Located in an EA dialog by selecting Help About EA) |
| elemType | The element type: "Interface" or "Class" |
| fileExtension | The file type extension of the file being generated |
| fileName | The name of the file being generated |
| fileNameImpl | The filename of the implementation file for this generation if applicable |

| Macro Name | Description |
|-----------------------------------|--|
| fileHeaders | Code Gen dialog: "Headers" |
| fileImports | Code Gen dialog: "Imports" |
| genOptActionScriptVersion | ActionScript Specifications dialog: "Default Version" |
| genOptCPPCommentStyle | C++ Specifications dialog: "Comment Style" |
| genOptCPPDefaultAttributeType | C++ Specifications dialog: "Default Attribute Type" |
| genOptCPPDefaultReferenceType | C++ Specifications dialog: "Default Reference Type" |
| genOptCPPDefaultSourceDirectory | C++ Specifications dialog: "Default Source Directory" |
| genOptCPPGenMethodNotesInHeader | C++ Specifications dialog: "Method Notes In Header" checkbox |
| genOptCPPGenMethodNotesInBody | C++ Specifications dialog: "Method Notes In Body" checkbox |
| genOptCPPGetPrefix | C++ Specifications dialog: "Get Prefix" |
| genOptCPPHeaderExtension | C++ Specifications dialog: "Header Extension" |
| genOptCPPSetPrefix | C++ Specifications dialog: "Set Prefix" |
| genOptCPPSourceExtension | C++ Specifications dialog: "Source Extension" |
| genOptCPPSynchCPPFile | C++ Specifications dialog: "Synchronize Notes" |
| genOptCPPSynchNotes | C++ Specifications dialog: "Synchronize CPP File" |
| genOptCSDefaultAttributeType | C# Specifications dialog: "Default Attribute Type" |
| genOptCSSourceExtension | C# Specifications dialog: "Default file extension" |
| genOptCSGenDispose | C# Specifications dialog: "Generate Dispose" |
| genOptCSGenFinalizer | C# Specifications dialog: "Generate Finalizer" |
| genOptCSGenNamespace | C# Specifications dialog: "Generate Namespace" |
| genOptCSDefaultSourceDirectory | C# Specifications dialog: "Default Source Directory" |
| genOptDefaultAssocAttributeName | Attribute Specifications dialog: "Default name for associated attrib" |
| genOptDefaultConstructorScope | Object Lifetimes dialog: "Default Constructor Visibility" |
| genOptDefaultCopyConstructorScope | Object Lifetimes dialog: "Default Copy Constructor Visibility" |
| genOptDefaultDatabase | Code Editors dialog: "Default Database" |
| genOptDefaultDestructorScope | Object Lifetimes dialog: "Default Destructor Constructor Visibility" |
| genOptGenCapitalisedProperties | Source Code Engineering Dialog: "Capitalize Attribute Names for Properties" checkbox |
| genOptGenComments | Source Code Engineering Dialog: "Generate Comments" checkbox |
| genOptGenConstructor | Object Lifetimes dialog: "Generate Constructor" checkbox |
| genOptGenConstructorInline | Object Lifetimes dialog: "Constructor Inline" checkbox |
| genOptGenCopyConstructor | Object Lifetimes dialog: "Generate Copy Constructor" checkbox |
| genOptGenCopyConstructorInline | Object Lifetimes dialog: "Copy Constructor Inline" checkbox |
| genOptGenDestructor | Object Lifetimes dialog: "Generate Destructor" checkbox |
| genOptGenDestructorInline | Object Lifetimes dialog: "Destructor Inline" checkbox |

| Macro Name | Description |
|----------------------------------|---|
| genOptGenDestructorVirtual | Object Lifetimes dialog: "Virtual Destructor" checkbox |
| genOptGenImplementedInterfaceOps | Attribute/Operations Specifications dialog: "Generate methods for implemented interfaces" checkbox |
| genOptGenPrefixBooleanProperties | Source Code Engineering Dialog: "Use is prefix for boolean property Get()" checkbox |
| genOptGenRoleNames | Source Code Engineering Dialog: "Autogenerate role names when creating code" |
| genOptGenUnspecAssocDir | Source Code Engineering Dialog: "Do not generate members where Association direction is unspecified" checkbox |
| genOptJavaDefaultAttributeType | Java Specifications dialog: "Default attribute type" |
| genOptJavaGetPrefix | Java Specifications dialog: "Get Prefix" |
| genOptJavaDefaultSourceDirectory | Java Specifications dialog: "Default Source Directory" |
| genOptJavaSetPrefix | Java Specifications dialog: "Set Prefix" |
| genOptJavaSourceExtension | Java Specifications dialog: "Source code extension" |
| genOptPHPDefaultSourceDirectory | PHP Specifications dialog: "Default Source Directory" |
| genOptPHPGetPrefix | PHP Specifications dialog: "Get Prefix" |
| genOptPHPSetPrefix | PHP Specifications dialog: "Set Prefix" |
| genOptPHPSourceExtension | PHP Specifications dialog: "Default file extension" |
| genOptPHPVersion | PHP Specifications dialog: "PHP Version" |
| genOptPropertyPrefix | Source Code Engineering Dialog: "Remove prefixes when generating Get/Set properties" |
| genOptVBMultiUse | VB Specifications dialog: "Multiuse" checkbox |
| genOptVBPersistable | VB Specifications dialog: "Persistable" checkbox |
| genOptVBDataBindingBehavior | VB Specifications dialog: "Data binding behavior" checkbox |
| genOptVBDataSourceBehavior | VB Specifications dialog: "Data source behavior" checkbox |
| genOptVBGlobal | VB Specifications dialog: "Global namespace" checkbox |
| genOptVBCreatable | VB Specifications dialog: "Creatable" checkbox |
| genOptVBExposed | VB Specifications dialog: "Exposed" checkbox |
| genOptVBMTS | VB Specifications dialog: "MTS Transaction Mode" |
| genOptVBNetGenNamespace | VB.Net Specifications dialog: "Generate Namespace" |
| genOptVBVersion | VB Specifications dialog: "Default Version" |
| genOptWrapComment | Source Code Engineering Dialog: "Wrap length for comment lines" |
| importClassName | The name of the class being imported. |
| importFileName | The filename of the class being imported. |
| importFromAggregation | 'T' if the class has an aggregation link to a class in this file, 'F' otherwise. |
| importFromAssociation | 'T' if the class has an association link to a class in this file, 'F' otherwise. |
| importFromAtt | 'T' if an attribute of a class in the current file is of the type of this class, 'F' otherwise. |
| importFromDependency | 'T' if the class has an dependency link to a class in this file, 'F' otherwise. |
| importFromGeneralization | 'T' if the class has an generalization link to a class in this file, 'F' otherwise. |
| importFromMeth | 'T' if a method return type of a class in the current file is the type of this class, 'F' otherwise. |
| importFromParam | 'T' if an method parameter of a class in the current file is of the type of this class, 'F' otherwise. |
| importFromRealization | 'T' if the class has an realization link to a class in this file, 'F' otherwise. |

| Macro Name | Description |
|--------------------------------|--|
| importInFile | 'T' if the class is in the current file, 'F' otherwise. |
| importPackagePath | The package path with a '.' separator of the class being imported. |
| linkAttAccess | Association Properties Target Role dialog: "Access" |
| linkAttCollectionClass | The collection appropriate for the linked attribute in scope |
| linkAttContainment | Association Properties Target Role dialog: "Containment" |
| linkAttName | Association Properties dialog: "Target" |
| linkAttNotes | Association Properties Target Role dialog: "Role Notes" |
| linkAttQualName | The Association target qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). |
| linkAttRole | Association Properties Target Role dialog: "Role" |
| linkAttStereotype | Association Properties Target Role dialog: "Stereotype" |
| linkAttTargetScope | Association Properties Target Role dialog: "Target Scope" |
| linkCard | Link Properties Target Role dialog: "Multiplicity" |
| linkGUID | The unique GUID for the current link |
| linkParentName | Generalization Properties dialog : "Target" |
| linkParentQualName | The Generalization target qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). |
| linkStereotype | The stereotype of the current link |
| linkVirtualInheritance | Generalization Properties dialog : "Virtual Inheritance" |
| opAbstract | Operation dialog: "Virtual" checkbox |
| opAlias | Operation dialog: "Alias" |
| opBehaviour | Operation Behavior dialog: "Behavior" |
| opCode | Operation Behavior dialog: "Initial Code" |
| opConcurrency | Operation dialog: "Concurrency" |
| opConst | Operation dialog: "Const" checkbox |
| opGUID | The unique GUID for the current operation |
| opImplMacros | A space separated list of macros defined in the implementation of this operation. |
| opIsQuery | Operation dialog: "IsQuery" checkbox |
| opMacros | A space separated list of macros defined in the declaration for this operation |
| opName | Operation dialog: "Name" |
| opNotes | Operation dialog: "Notes" |
| opPure | Operation dialog: "Pure" checkbox |
| opReturnArray | Operation dialog: "Return Array" checkbox |
| opReturnQualType | The operation return type qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). If the return type classifier has not been set, is equivalent to the opReturnType macro. |
| opReturnType | Operation dialog: "Return Type" |
| opScope | Operation dialog: "Scope" |
| opStatic | Operation dialog: "Static" checkbox |
| Operation dialog: "Stereotype" | Operation dialog: "Stereotype" |
| opSynchronized | Operation dialog: "Synchronized" checkbox |
| packageAlias | Package dialog: "Alias" |
| packageName | Package dialog: "Name" |
| packageGUID | The unique GUID for the current package |
| packagePath | The string representing the hierarchy of packages, for the class in scope. Each package name is separated by a dot (.) |
| paramDefault | Operation Parameters dialog: "Default" |
| paramFixed | Operation Parameters dialog: "Fixed" checkbox |
| paramGUID | The unique GUID for the current parameter |
| paramIsEnum | True, if the parameter uses the enum keyword (C++) |

| Macro Name | Description |
|---------------|--|
| paramKind | Operation Parameters dialog: "Kind" |
| paramName | Operation Parameters dialog: "Name" |
| paramNotes | Operation Parameters dialog: "Notes field" |
| paramQualType | The parameter type qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). If the parameter classifier has not been set, is equivalent to the paramType macro. |
| paramType | Operation Parameters dialog: "Type" |

Field substitution macros may be used in one of two ways:

Use 1: Direct Substitution

This form directly substitutes the corresponding value of the element in scope into the output.

Structure: %<macroName>%

where <macroName> can be any of the macros listed above.

Examples:

- %className%
- %opName%
- %attName%

Use 2: Conditional Substitution

This form of the macro allows alternative substitutions to be made depending on the macro's value.

Structure: %<macroName> [== "<test>"] ? <subTrue> [: <subFalse>]%

Where:

- [<text>] denotes that <text> is optional.
- <test> is a string representing a possible value for the macro.
- <subTrue> and <subFalse> may be a combination of quoted strings and the keyword value. Where the value is used, it gets replaced with the macro's value in the output.

Examples:

- %classAbstract=="T" ? "pure" : ""%
- %opStereotype=="operator" ? "operator" : ""%
- %paramDefault != "" ? " = " value : ""%

The above three examples will output nothing if the condition fails. In this case the false condition can be omitted resulting in the following usage.

Examples:

- %classAbstract=="T" ? "pure"%
- %opStereotype=="operator" ? "operator"%
- %paramDefault != "" ? " = " value%

The third example of both blocks shows a comparison checking for a non-empty value or existence. This test can also be omitted.

- %paramDefault ? " = " value : ""%
- %paramDefault ? " = " value%

All of the above examples containing paramDefault are equivalent. If the parameter in scope had a default value of 10, we would expect the output from each of them to be:

= 10

Note: In a conditional substitution macro, any whitespace following <macroName> is ignored. If whitespace is required in the output, it should be included within the quoted substitution strings.

10.4.1.4.2.3 Tagged Value Macros

Tagged value macros are a special form of field substitution macros, which provide access to element tags and the corresponding tagged values.

Use 1: Direct Substitution

This form of the macro directly substitutes the value of the named tag into the output.

Structure: %<macroName>:"<tagName>"%

where <macroName> can be one of:

- attTag
- classTag
- opTag
- packageTag
- paramTag
- connectorTag
- connectorSourceTag
- connectorDestTag
- linkTag
- linkAttTag

which corresponds to the tags for attributes, classes, operations, packages, parameters, connectors with both ends and links including the attribute end respectively. <tagName> is a string representing the specific tag name.

Examples:

- %opTag:"attribute"%

Use 2: Conditional Substitution

This form of the macro mimics the conditional substitution defined for field substitution macros.

Structure: %<macroName>:"<tagName>" [== "<test>"] ? <subTrue> [: <subFalse>]%

Where:

- <macroName> and <tagName> are as defined above.
- [<text>] denotes that <text> is optional.
- <test> is a string representing a possible value for the macro.
- <subTrue> and <subFalse> may be a combination of quoted strings and the keyword value. Where the value is used, it gets replaced with the macro's value in the output.

Examples:

- %opTag:"opInline" ? "inline" : ""%
- %opTag:"opInline" ? "inline" %
- %classTag:"unsafe" == "true" ? "unsafe" : ""%
- %classTag:"unsafe" == "true" ? "unsafe" %

Tagged value macros use the same naming convention as field substitution macros.

10.4.1.4.2.4 Function Macros

Function macros are a convenient way of manipulating and formatting various element data. Each function macro returns a result string. There are two primary ways to use the result of function macros:

- Direct substitution of the returned string into the output, such as: %TO_LOWER(attName)%
- Storing the returned string as part of a variable definition such as: \$name = %TO_LOWER(attName)%

Function macros can take parameters. Parameters may be passed to function macros as:

- string literals, enclosed within double quotation marks.
- direct substitution macros without the enclosing percent signs.
- variable references.
- numeric literals

Multiple parameters are passed using a comma separated list.

The available function macros are described below. Parameters are denoted by angle brackets, as in:
FUNCTION_NAME(<param>)

CONVERT SCOPE(<umlScope>)

Converts <umlScope> to the appropriate scope keyword for the language being generated. The following table shows the conversion of <umlScope> with respect to the given language.

| Language | Package | Public | Private | Protected |
|----------|-----------|--------|---------|-----------|
| C++ | public | public | private | protected |
| C# | internal | public | private | protected |
| Delphi | protected | public | private | protected |
| Java | | public | private | protected |
| PHP | public | public | private | protected |
| VB | Protected | Public | Private | Protected |
| VB .Net | Friend | Public | Private | Protected |

The CONVERT_SCOPE macro is for use with supported languages.

COLLECTION CLASS(<language>)

Gives the appropriate collection class for the language specified for the current linked attribute.

CSTYLE COMMENT(<wrap length>)

Converts the notes for the element currently in scope to plain c style comments, using /* and */.

DELPHI PROPERTIES(<scope>, <separator>, <indent>)

Generates a Delphi property.

DELPHI COMMENT(<wrap length>)

Converts the notes for the element currently in scope to Delphi comments.

EXEC ADD IN(<addin name>, <function name>, <prm 1>, ..., <prm n>)

Invokes an EA add-in function, which may return a result string. <addin_name> and <function_name> specify the names of the add-in and function to be invoked. Parameters to the add-in function may be specified via parameters <prm_1> to <prm_n>.

Example:

```
$result = %EXEC_ADD_IN("MyAddin", "ProcessOperation", classGUID, opGUID)%
```

Any function that is to be called by the EXEC_ADD_IN macro must have two parameters, an EA.Repository object, and a Variant array which contains any additional parameters from the EXEC_ADD_IN call. Return type should be Variant.

Example:

```
Public Function ProcessOperation(Repository As EA.Repository, args As Variant) As Variant
```

FIND(<src>, <subString>)

Position of the first instance of <subString> in <src>, -1 if none.

GET ALIGNMENT()

Gives a string where all of the text on the current line of output is converted into spaces and tabs.

JAVADOC COMMENT(<wrap length>)

Converts the notes for the element currently in scope to javadoc style comments.

LEFT(<src>, <count>)

The first <count> characters of <src>.

LENGTH(<src>)

Length of <src>.

MID(<src>, <count>)**MID(<src>, <start>, <count>)**

Substring of <src> starting at <start> and including <count> characters. Where count is omitted the rest of the string is included.

REMOVE DUPLICATES(<source>, <separator>)

Where <source> is a <separator> separated list, this will remove any duplicate or empty strings.

REPLACE(<string>, <old>, <new>)

Replaces all occurrences of <old> with <new> in the given string <string>.

RESOLVE OP NAME()

Resolves clashes in interface names where two method from interfaces have the same name.

RESOLVE QUALIFIED TYPE()**RESOLVE QUALIFIED TYPE(<separator>)****RESOLVE QUALIFIED TYPE(<separator>, <default>)**

Generates a qualified type for the current attribute, linked attribute, linked parent, operation, or parameter. Allows the specification of a separator other than . and a default value for when some value is required.

RIGHT(<src>, <count>)

The last <count> characters of <src>.

TO LOWER(<string>)

Converts <string> to lower case.

TO UPPER(<string>)

Converts <string> to upper case.

TRIM(<string>)**TRIM(<string>, <trimChars>)**

Removes trailing and leading white spaces from <string>. If <trimChars> is specified, all leading and trailing

characters in the set of <trimChars> are removed.

TRIM LEFT(<string>)

TRIM LEFT(<string>, <trimChars>)

Removes the specified leading characters from <string>.

TRIM RIGHT(<string>)

TRIM RIGHT(<string>, <trimChars>)

Removes the specified trailing characters from <string>.

VB COMMENT(<wrap length>)

Converts the notes for the element currently in scope to Visual Basic style comments.

WRAP COMMENT(<comment>, <wrap length>, <indent>, <start string>)

Wraps the text <comment> at width <wrap_length> putting <indent> and <start_string> at the beginning of each line.

Example:

```
$behaviour = %WRAP_COMMENT(opBehaviour, "40", " ", "//")%
```

Note that <wrap_length> must still be pass as a string, even though WRAP_COMMENT treats this parameter as an integer.

WRAP LINES(<text>, <wrap length>, <start string>[, <end string>])

Wraps <text> as designated to be <wrap_length>, adding <start_string> to the beginning of every line and <end_string> to the end of the line if it is specified.

XML COMMENT(<wrap length>)

Converts the notes for the element currently in scope to xml style comments.

Function macros are named according to the All-Caps style, as in: %CONVERT_SCOPE (opScope)%

10.4.1.4.2.5 Control Macros

Control macros are used to control the processing and formatting of the templates. The basic types of control macros include:

- The list macro, for generating multiple elements, such as attributes and operations
- The branching macros, which form if-then-else constructs to conditionally execute parts of a template
- The PI macro for formatting newlines in the output
- The synchronization macros

In general, control macros are named according to Camel casing.

Lists

The list is used to generate multiple elements. The basic structure is:

```
%list=<TemplateName>@separator=<string>@indent=<string> [<conditions>]%
```

where <string> is a double-quoted literal string and <TemplateName> may refer to one the following template names:

- Namespace

- Class
 - ClassImpl
 - Attribute
 - InnerClass
 - InnerClassImpl
 - Operation
 - OperationImpl
 - Parameter
 - ClassBase
 - ClassInterface
 - Custom Template, Custom templates allow the user to define their own templates more information is found in [Adding a Custom Template](#).
- <conditions> is optional and appears the same as the conditions for if and elseif statements.

Example:

```
%list="Attribute" @separator="\n" @indent="  "%
```

The separator attribute, denoted above by @separator, specifies the space that should be used between each list item. This excludes the last item in the list.

The indent attribute, denoted by @indent, specifies the space by which each line in the generated output should be indented.

The above example would output the result of processing the Attribute template, for each attribute element of the class in scope. The resultant list would separate and indent its items by a single newline and two spaces respectively. If the class in scope had any stereotyped attributes, they would be generated using the appropriately specialised template.

There are some special cases to consider when using the list macro:

- If the Attribute template is used as an argument to the list macro, this will also generate attributes derived from associations by executing the appropriate LinkedAttribute template
- If the ClassBase template is used as an argument to the list macro, this will also generate class bases derived from links in the model by executing the appropriate LinkedClassBase template
- If the ClassInterface template is used as an argument to the list macro, this will also generate class bases derived from links in the model by executing the appropriate LinkedClassInterface template
- If InnerClass or InnerClassImpl is used as an argument to the list macro, these classes are generated using the Class and ClassImpl templates respectively. These arguments tell EA that it should process the templates based on the inner classes of the class in scope.

Branching (if-then-else Constructs)

The CTF supports a limited form of branching through the following macros:

- if
- elseif
- endif
- endTemplate

The basic structure of the if and elseif macros is:

```
%if <test> <operator> <test>%
```

where <operator> may be one of:

- ==
- !=

and <test> may be one of:

- a string literal, enclosed within double quotation marks.
- a direct substitution macro, without the enclosing percent signs
- a variable reference.

Branches may be nested, and multiple conditions may be specified using one of:

- and
- or

Note: When specifying multiple conditions, "and" and "or" have the same order of precedence and conditions are processed left to right.

The `endif` or `endTemplate` macros must be used to signify the end of a branch. In addition, the `endTemplate` macro causes the template to return immediately, if the corresponding branch is being executed.

Example:

```
%if elementType == "Interface"%
;
%else%
%OperationBody%
%endif%
```

Example:

```
$bases=%list="ClassBase"@separator=", "%
$interfaces=%list="ClassInterface"@separator=", "%
%if $bases != "" and $interfaces != ""%
: $bases, $interfaces
%elseif $bases != ""%
: $bases
%elseif $interfaces != ""%
: $interfaces
%endif%
```

The PI Macro

There are two primary means of generating whitespace from the templates:

- Explicitly using the newline, space and tab characters (`\n`, `\t`) as part of Literal Text
- Using the PI macro to format lines in the template that result in non-empty substitutions in the output

By default, each template line that generates a non-empty substitution also results in a newline being produced in the output. This behavior can be changed through the PI macro.

To demonstrate the use of the PI macro, consider the default C# Operation template:

```
%opTag:"Attribute"%
```

Default PI is \n, so any attributes would be on their own line

```
%PI=" "%
```

Blank lines have no effect on the output

```
%opTag:"unsafe"=="true"? "unsafe" : ""%
%CONVERT_SCOPE(opScope)%
%opTag:"new"=="true"? "new" : ""%
%opAbstract=="T"? "abstract" : ""%
%opConst=="T"? "sealed" : ""%
%opStatic=="T"? "static" : ""%
%opTag:"extern"=="true"? "extern" : ""%
%opTag:"delegate"=="true"? "delegate" : ""%
%opTag:"override"=="true"? "override" : ""%
%opTag:"virtual"=="true"? "virtual" : ""%
%opReturnType%opReturnArray=="T"? "[]" : ""%
%opStereotype=="operator"? "operator" : ""%
%opName%(%list="Parameter"@separator=", "%)
```

Set the PI, so keywords are separated by a space

Any keyword that does not apply, ie. the macro produces an empty result, will not result in a space

Only one space is generated for this line

The final line in the template will not generate a space

In the above example we want to arrange macros for the various keywords vertically for readability. In the output however, we want each relevant keyword to be separated by a single space. This is achieved by the line:

```
%PI=" "%
```

Notice how we do not need to specify the space between each of the possible keywords. This space is already implied by setting the PI to a single space. Essentially the PI acts as a convenience mechanism for formatting the output from within the templates.

The structure for setting the processing instruction is:

```
%PI=<value>%
```

where <value> may be a literal string enclosed by double quotes

The following points apply the PI macro:

- The value of the PI is not accessed explicitly
- Only template lines which result in a non-empty substitution cause the PI to be generated
- The last non-empty template line does not cause the PI to be generated
- The PI is not appended to the last substitution, regardless of which template line caused that substitution

The Synchronization Macros

The synchronization macros are used to provide formatting hints to EA when inserting new sections into the source code, during forward synchronization. The values for synchronization macros must be set in the File templates.

The structure for setting synchronization macros is:

```
%<name>=<value>%
```

where <name> may be one of the macros listed below and <value> is a literal string enclosed by double quotes.

| Macro Name | Description |
|-----------------------------|---|
| synchNewClassNotesSpace | Space to append to a new class note. Default value: \n |
| synchNewAttributeNotesSpace | Space to append to a new attribute note. Default value: \n |
| synchNewOperationNotesSpace | Space to append to a new operation note. Default value: \n |
| synchNewOperationBodySpace | Space to append to a new operation body. Default value: \n |
| synchNamespaceBodyIndent | Indent applied to classes within non-global namespaces. Default value: \t |

10.4.1.4.3 Variables

Template variables provide a convenient way of storing and retrieving data within a template. The following topics explain how variables are defined and referenced.

See Also

- [Variable Definitions](#)
- [Variable References](#)

10.4.1.4.3.1 Variable Definitions

Variable definitions take the basic form:

```
$<name> = <value>
```

where <name> may be any alpha-numeric sequence and <value> is derived from a macro or another variable

A simple example definition would be:

```
$foo = %className%
```

Variables may be defined, using values from:

- Substitution, function or list macros.
- String literals, enclosed within double quotation marks
- Variable references

The following rules apply to variable definitions:

- Variables have global scope within the template in which they are defined and are not accessible to other templates
- Each variable must be defined at the start of a line, without any intervening whitespace
- Variables are denoted by prefixing the name with \$, as in \$foo
- Variables do not need to be declared, prior to being defined
- Variables must be defined using either the assignment operator (=), or the addition-assignment operator (+=)
- Multiple terms may be combined in a single definition using the addition operator (+)

Examples

Using a substitution macro:

```
$foo = %opTag: "bar" %
```

Using a literal string:

```
$foo = "bar"
```

Using another variable:

```
$foo = $bar
```

Using a list macro :

```
$ops = %list="Operation" @separator="\n\n" @indent="\t" %
```

Using the addition-assignment operator (+=) :

```
$body += %list="Operation" @separator="\n\n" @indent="\t" %
```

The above definition is equivalent to the following:

```
$body = $body + %list="Operation" @separator="\n\n" @indent="\t" %
```

Using multiple terms:

```
$templateArgs = %list="ClassParameter" @separator=" , " %  
$template = "template<" + $templateArgs + ">"
```

10.4.1.4.3.2 Variable References

Variable values may be retrieved by using a reference of the form:

```
$<name>
```

where <name> may refer to a previously defined variable.

Variables references may be used in one of the following ways:

- As part of a macro, such as the argument to a function macro
- As a term in a variable definition
- As a direct substitution of the variable value into the output

Note: It is legal to reference a variable before it is defined. In this case, the variable is assumed to contain an empty string value: ""

Example:

Using variables as part of a macro. The following is an excerpt from the default C++ ClassNotes template.

```
$wrapLen = %genOptWrapComment%
$style = %genOptCPPCommentStyle%

%if $style == "XML.NET"%
%XML_COMMENT($wrapLen)%
%else%
%CSTYLE_COMMENT($wrapLen)%
%endif%
```

Define variables to store the style and wrap length options

Reference to \$style as part of a condition

Reference to \$wrapLen as an argument to function macro

Example:

Using variable references as part of a variable definitions:

```
$foo = "foo"
$bar = "bar"

$foobar = $foo + $bar
```

Define our variables

\$foobar now contains the value "foobar"

Example:

Substituting variable values into the output

```
$bases=%ClassInherits%
```

...

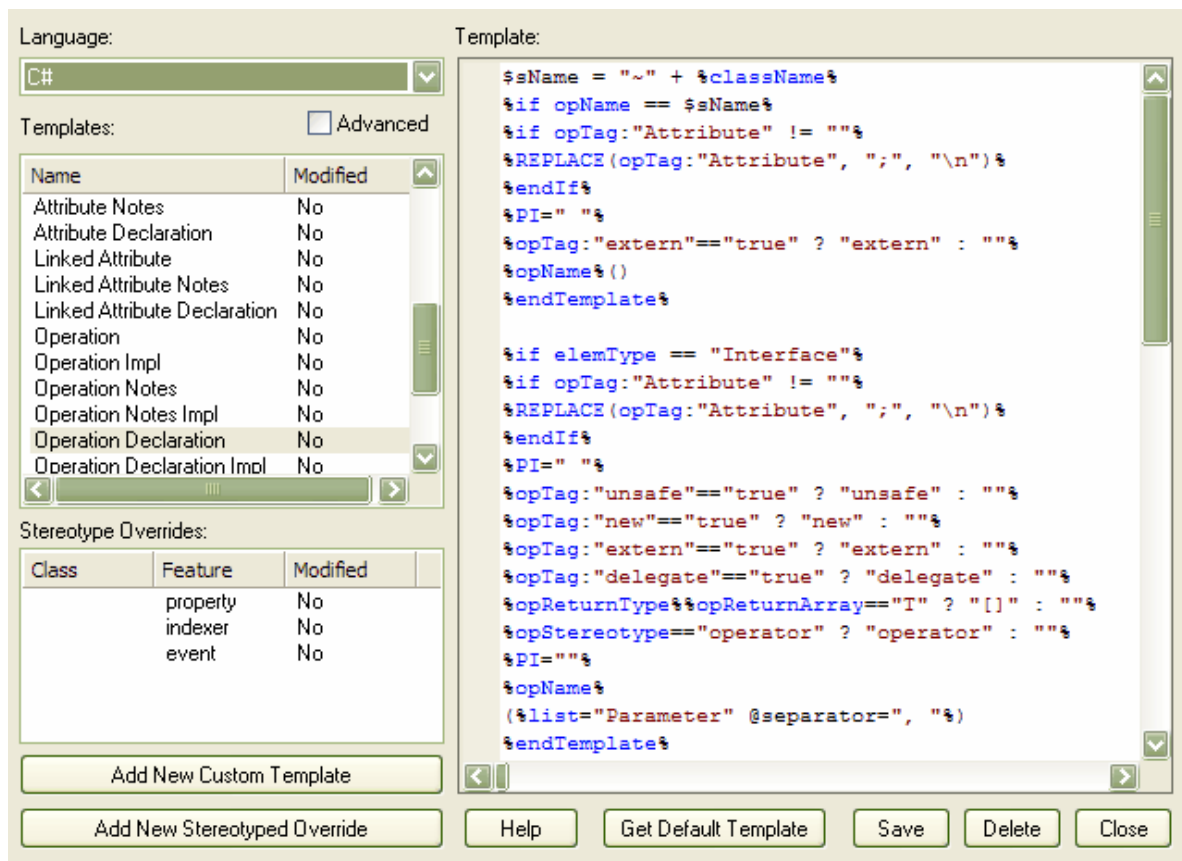
```
class %className%$bases
```

Store the result of the ClassInherits template in \$bases

Now output the value of \$bases after the class name

10.4.2 The Code Template Editor

The *Code Templates Editor* is accessed by selecting *Code Generation Templates* from the *Settings* menu.



| Control | Description |
|------------------------------|---|
| Language | Selects the programming language. |
| Template | Displays the contents of the active template. Provides the editor for modifying templates. |
| Templates | Lists the base code templates. The active template is highlighted. The modified field indicates whether the user has changed the default template for the current language. |
| Stereotype Overrides | Lists the stereotyped templates, for the active base template. The modified field indicates whether a default stereotyped template has been modified by the user. |
| Add New Stereotyped Override | Invokes a dialog for adding a stereotyped template, for the currently selected base template. |
| Add New Custom Template | Invokes a dialog for creating a custom stereotyped template. |
| Get Default Template | Updates the editor display with the default version of the active template. |
| Save | Overwrites the active templates with the contents of the editor. |
| Delete | If the active template has been overridden by the user, the override is deleted and replaced by the corresponding default code template. |
| Close | Exits the Code Template Editor dialog. |

See Also

- [Overriding Default Templates](#)
- [Adding New Stereotyped Templates](#)
- [Creating Templates For Custom Languages](#)
- [Importing and Exporting Code Templates](#)

10.4.2.1 Overriding Default Templates

EA has a set of built-in or default code generation templates. The *Code Templates Editor* allows users to modify these default templates, hence customizing the way in which EA generates code. Users may choose to modify any or all of the base templates to achieve their desired coding style.

Any templates that are overridden by the user are stored in the .EAP file. When generating code, EA first checks whether a template has been modified and if so, uses that template. Otherwise the appropriate default template is used.

Override a Default Template

To override a default code generation template, follow the steps below.

1. Select *Code Generation Templates* from the *Configuration* menu to open the *Code Templates Editor*.
2. Select the appropriate language from the *Language* list.
3. Select one of the base templates from the *Templates* list.
4. If the base template has stereotyped overrides, you may select one of these from the *Stereotype Overrides* list.
5. In the *Template* editor, make the desired modifications.
6. Press *Save*. This stores the modified version of the template to the .EAP file. The template will be marked as modified.

When generating code, EA will now use the overridden template, instead of the default template.

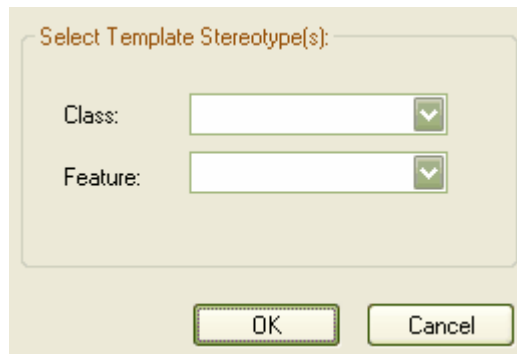
10.4.2.2 Adding New Stereotyped Templates

Sometimes it is useful to define a specific code generation template for use with elements of a given stereotype. This allows different code to be generated for elements, depending on their stereotype. EA provides some default templates, which have been specialised for commonly used stereotypes in supported languages. For example the Operation Body template for C# has been specialized for the property stereotype, so as to automatically generate its constituent get and set methods. Users may override the default stereotyped templates as described in the previous section. Additionally users may define templates for their own stereotypes, as described below.

Add A New Stereotyped Template

To override a default code generation template, follow the steps below.

1. Select *Code Generation Templates* from the *Configuration* menu to open the *Code Templates Editor*.
2. Select the appropriate language, from the *Language* list.
3. Select one of the base templates, from the *Templates* list.
4. Press *Add New Stereotyped Override*.
5. This opens the *New Template Override* dialog.



6. Select the desired *Feature* and/or *Class* stereotype and press *OK*.
7. The new stereotyped template override will appear in *Stereotype Overrides* list and be marked as modified.
8. Make the desired modifications in the *Template* editor.
9. Press *Save*. This stores the new stereotyped template in the .EAP file.

EA will now use the stereotyped template, when generated code for elements of that stereotype.

Note that class and feature stereotypes can be combined to provide a further level of specialization for features. For example if properties should be generated differently when the class has a stereotype "MyStereotype", then both "property" and "MyStereotype" should be specified in the *New Template Override* dialog.

10.4.2.3 Creating Templates For Custom Languages

EA can forward generate code for languages that it does not specifically support, if the appropriate code generation templates are defined for that language. This section outlines the steps required to define templates for custom languages.

Define a Template for a Custom Language

1. Create the custom language as a new product. To do this, go to the Language Datatypes dialog (select *Language Datatypes* from the *Configuration* menu), enter the name of the new language in the *Product Name* field and enter a datatype (one is enough to declare that the new language exists). See [Data Types](#) for more details.
2. Select *Code Generation Templates* from the *Configuration* menu to open the *Code Templates Editor*.
3. Select the custom language, from the *Language* list.
4. Select one of the base templates, from the *Templates* list.
5. Define the template using the *Template* editor.
6. Press *Save*. This stores the template in the .EAP file.
7. Repeat steps 1 to 6 for each of the relevant base templates for the custom language.

Note: The File template must be defined for the custom language. The File template can then refer to the *Import Section*, *Namespace* and *Class* templates.

10.4.2.4 Importing and Exporting Code Templates

User defined Code Templates can be imported and exported as Reference Data (see [Import and Export Reference Data](#)). The templates defined for each language appear in list of tables with the language name suffixed with "_Code_Templates".

10.4.3 Synchronizing Code

EA uses code templates during the forward synchronization of the following programming languages:

- C++
- C#
- Delphi
- Java
- VB
- VB.Net

Only a subset of the code templates are used during synchronization. This subset corresponds to the distinct sections that EA recognises in the source code. The following table lists the code templates and their corresponding code sections, which may be synchronized.

| Code Template | Code Section |
|----------------------------|---|
| Class Notes | Comments preceding class declaration |
| Class Declaration | Up to and including, class parents |
| Attribute Notes | Comments preceding Attribute declaration |
| Attribute Declaration | Up to and including terminating character |
| Operation Notes | Comments preceding operation declaration |
| Operation Notes Impl | As for "Operation Notes" |
| Operation Declaration | Up to and including, terminating character |
| Operation Declaration Impl | Up to and including terminating character |
| Operation Body | Everything between and including the braces |
| Operation Body Impl | As for "Operation Body" |

Three types of changes can occur in the source when it is synchronized with the UML model:

- [Synchronizing Existing Sections](#): for example, changing the return type in an operation declaration.
- [Adding New Sections to Existing Features](#): for example, adding notes to a class declaration, where there were previously none.
- [Adding New Features and Elements](#): for example, adding a new operation to a class.

Each of these changes must be handled differently by EA and their effect on the CTF is described in the following sections.

10.4.3.1 Synchronizing Existing Sections

When an existing section in the source code differs from the result generated by the corresponding template, that section is replaced. Consider for example, the following C++ class declaration:

```
[asm] class A : public B
```

Now assume we add an inheritance relationship from class A to class C, the entire class declaration would be replaced with something like:

```
[asm] class A : public B, public C
```

10.4.3.2 Adding New Sections to Existing Features

The following can be added as new sections, to existing features in the source code:

- Class Notes
- Attribute Notes
- Operation Notes
- Operation Notes Impl
- Operation Body
- Operation Body Impl

Assume class "A" from the previous example had no note when we originally generated the code. Now assume that we specify a note in the model for class "A." EA will attempt to add the new note from the model during synchronization. It will do this by executing the Class Notes template.

To make room for the new section to be inserted, we can specify how much whitespace to append to the section via synchronization macros. These macros are described in the section: [Control Macros](#).

10.4.3.3 Adding New Features and Elements

The following features and elements can be added to the source code during synchronization:

- Attributes
- Inner Classes
- Operations

These are added by executing the relevant templates for each new element or feature in the model. EA attempts to preserve the appropriate indenting of new features in the code, by finding the indents specified in list macros of the Class. For languages that make use of namespaces, the `synchNamespaceBodyIndent` macro is available. Classes defined within a (non-global) namespace will be indented according to the value set for this macro, during synchronisation. This value is ignored for classes defined within a package setup as a root namespace, or if the Generate Namespace option is turned off.

10.5 Modeling Conventions

In order to get the most out of the round trip engineering in EA, you need to be familiar with the modeling conventions used when generating and reverse engineering the languages you use. This section describes the stereotypes, tagged values and other conventions used by code engineering in EA for the following supported languages.

- [Actionscript](#)
- [C#](#)
- [C++](#)
- [Delphi](#)
- [Java](#)
- [PHP](#)
- [Python](#)
- [VB.Net](#)

- [Visual Basic](#)

10.5.1 Actionscript Conventions

EA supports the round trip engineering of Actionscript 2 and 3 where the following conventions are used.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|---|
| property get | Operation | A read property. |
| property set | Operation | A write property. |
| literal | Operation | A literal method referred to by a variable. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|----------------|--|--|
| dynamic | Class or Interface | The dynamic keyword. |
| intrinsic | Actionscript 2: Class | The intrinsic keyworded |
| attribute_name | Operation with stereotype "property get" or "property set" | The name of the variable behind this property. |
| namespace | Actionscript 3: Class, Interface, Attribute, Operation | The namespace of the current element. |
| prototype | Actionscript 3: Attribute | The prototype keyword. |
| final | Actionscript 3: Operation | The final keyword. |
| override | Actionscript 3: Operation | The override keyword. |
| rest | Actionscript 3: Parameter | The rest parameter (...) |

Common conventions

- Package qualifiers (ActionScript 2) and Packages (ActionScript 3) are generated when the current package is not a [namespace root](#).
- An unspecified type is modeled as *var* or an empty *Type* field.

Actionscript 3 conventions

- The *Is Leaf* property of a class corresponds to the sealed keyword.
- If a namespace tag is specified it overrides the *Scope* that is specified.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [ActionScript Options](#)

10.5.2 C Conventions

EA supports the round trip engineering of C where the following conventions are used:

Stereotype

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|--|
| Enumeraton | Inner class | An enum type |
| struct | Inner class | An struct type |
| | attribute | A keyword "struct" in variable definition |
| union | Inner class | An union type |
| | attribute | A keyword "union" in variable definition |
| typedef | Inner class | A typedef statement, where the parent is the original type name. |

Tagged Values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|------------|--|--|
| typedef | Class with stereotype other than typedef | This class being defined in a typedef statement. |
| anonymous | Class also containing the tagged value typedef | The name of this class being only defined by the typedef statement |

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|--------------|-------------------|---|
| bodyLocation | Operation | The location the method body is generated to. Expected values are header, classDec or classBody |

C Code generation for UML model

| <i>UML</i> | <i>C Code</i> | <i>Notes</i> |
|----------------------------------|---|-------------------------------------|
| A class | A pair of C files (.h + .c) | File name is the same as class name |
| Operation (public & protected) | Function declaration in .h file and definition in .c file | |
| Operation (private) | Function definition in .c file only | |
| Attribute (public & protected) | Variable definition in .h file | |
| Attribute (private) | Variable definition in .c file | |
| Inner class (without stereotype) | (N/A) | This inner class would be ignored |

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [C Options](#)

10.5.3 C# Conventions

EA supports the round trip engineering of C# where the following conventions are used.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|--|
| enumeration | Class | An enum type. |
| struct | Class | A struct type. |
| indexer | Operation | A property acting as an index for this class. |
| event | Operation | An event. |
| property | Operation | A property possibly containing both read and write code. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|--------------------|---|---|
| unsafe | Class, Interface, Operation | The unsafe keyword. |
| partial | Class, Interface | The partial keyword. |
| static | Class | The static keyword. |
| new | Class, Interface, Operation | The new keyword. |
| genericConstraints | Templated Class or Interface, Operation with tag generic. | The constraints on the generic parameters of this type or operation. |
| const | Attribute | The const keyword. |
| delegate | Operation | The delegate keyword. |
| extern | Operation | The extern keyword. |
| generic | Operation | The generic parameters for this Operation. |
| sealed | Operation | The sealed keyword. |
| override | Operation | The override keyword. |
| virtual | Operation | The virtual keyword. |
| Implements | Operation | The name of the method this implements, including the interface name. |
| ImplementsExplicit | Operation | The presence of the source interface name in this method declaration. |
| initializer | Operation | A constructor initialization list. |
| params | Parameter | A parameter list using the params keyword. |
| attribute_name | Operation with stereotype "property" or "event" | The name of the variable behind this property or event. |
| readonly | Operation with stereotype "property" | This property only defining read code. |

| | | |
|-----------|--------------------------------------|---|
| writeonly | Operation with stereotype "property" | This property only defining write code. |
|-----------|--------------------------------------|---|

Other conventions

- Namespaces are generated for each package below a [namespace root](#).
- The **Const** property of an attribute corresponds to the readonly keyword, while the tag const corresponds to the const keyword.
- The value of **inout** for the **Kind** property of a parameter corresponds to the ref keyword.
- The value of **out** for the **Kind** property of a parameter corresponds to the out keyword.
- Partial classes can be modeled as two separate classes with the partial tag.
- The **Is Leaf** property of a class corresponds to the sealed keyword.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [C# Options](#)

10.5.4 C++ Conventions

EA supports the round trip engineering of C++ including the [Managed C++](#) and [C++/CLI](#) extensions where the following conventions are used.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|--|
| enumeration | Class | An enum type. |
| struct | Class | A struct type. |
| property get | Operation | A read property. |
| property set | Operation | A write property. |
| union | Class | A union type. |
| typedef | Class | A typedef statement, where the parent is the original type name. |
| friend | Operation | The friend keyword. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|----------------|--|--|
| typedef | Class with stereotype other than typedef | This class being defined in a typedef statement. |
| anonymous | Class also containing the tagged value typedef | The name of this class being only defined by the typedef statement |
| attribute_name | Operation with stereotype "property get" or "property set" | The name of the variable behind this property. |
| mutable | Attribute | The mutable keyword. |
| inline | Operation | The inline keyword and inline generation of the method body. |
| explicit | Operation | The explicit keyword. |
| callback | Operation | A reference to the CALLBACK macro. |
| initializer | Operation | A constructor initialization list. |
| bodyLocation | Operation | The location the method body is generated to. Expected values are header, classDec or classBody. |
| typeSynonyms | Class | The typedef name and/or fields of this type. |
| throws | Operation | The exceptions that are thrown by this method. |
| afx_msg | Operation | The afx_msg keyword. |
| volatile | Operation | The volatile keyword. |

Other conventions

- Namespaces are generated for each package below a [namespace root](#).
- **By Reference** attributes correspond to a pointer to the type specified.
- The **Transient** property of an attribute corresponds to the volatile keyword.
- The **Abstract** property of an attribute corresponds to the virtual keyword.

- The *Const* property of an operation corresponds to the const keyword, specifying a constant return type.
- The *Is Query* property of an operation corresponds to the const keyword, specifying the method doesn't modify any fields.
- The *Pure* property of an operation corresponds to a pure virtual method using the "= 0" syntax.
- The *Fixed* property of a parameter corresponds to the const keyword.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [C++ Options](#)

10.5.4.1 Managed C++ Conventions

The following are the conventions used for the managed extensions to C++ prior to C++/CLI. In order to set EA to generate managed C++ you need to modify the C++ version in the [C++ Options](#).

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|---|
| reference | Class | The <code>__gc</code> keyword. |
| value | Class | The <code>__value</code> keyword. |
| property | Operation | The <code>__property</code> keyword. |
| property get | Operation | The <code>__property</code> keyword and a read property. |
| property set | Operation | The <code>__property</code> keyword and a write property. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------|--|--|
| managedType | Class with stereotype "reference", "value" or "enumeration", Interface | The keyword used in declaration of this type. Expected values are class or struct. |

Other conventions

- The typedef and anonymous tags from native C++ are not supported.
- The *Pure* property of an operation corresponds to the keyword `__abstract`.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [C++ Options](#)
- [C++/CLI Conventions](#)

10.5.4.2 C++/CLI Conventions

The following are the conventions used for modeling the C++/CLI extensions to C++. In order to set EA to generate managed C++/CLI you need to modify the C++ version in the [C++ Options](#).

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|----------------------|---|
| reference | Class | Corresponds to the ref class or ref struct keyword. |
| value | Class | Corresponds to the value class or value struct keyword. |
| property | Operation, Attribute | This is a property possibly containing both read and write code. |
| event | Operation | Defines an event to provide access to the event handler for this class. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------|--|--|
| managedType | Class with stereotype "reference", "value" or "enumeration", Interface | Corresponds to either the class or struct keyword. |

| | | |
|--------------------|---|---|
| generic | Operation | Defines the generic parameters for this Operation. |
| genericConstraints | Templated Class or Interface, Operation with tag generic. | Defines the constraints on the generic parameters for this Operation. |
| attribute_name | Operation with stereotype "property" or "event" | The name of the variable behind this property or event. |
| initonly | Attribute | Corresponds to the initonly keyword. |
| literal | Attribute | Corresponds to the literal keyword. |

Other conventions

- The typedef and anonymous tags are not used.
- The property get/property set stereotypes are not used.
- The *Pure* property of an operation corresponds to the keyword abstract.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [C++ Options](#)
- [Managed C++ Conventions](#)

10.5.5 Delphi Conventions

EA supports the round trip engineering of Delphi where the following conventions are used.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|-----------------------|
| struct | Class | A record type. |
| dispinterface | Class, Interface | A dispatch interface. |
| constructor | Operation | A constructor. |
| destructor | Operation | A destructor. |
| operator | Operation | An operator. |
| enumeration | Class | An enumerated type. |
| property get | Operation | A read property. |
| property set | Operation | A write property. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|----------------|--|---|
| packed | Class | The packed keyword. |
| property | Class | A property. See Delphi Properties for more information. |
| overload | Operation | The overload keyword. |
| reintroduce | Operation | The reintroduce keyword. |
| override | Operation | The override keyword. |
| attribute_name | Operation with stereotype "property get" or "property set" | The name of the variable behind this property. |

Other conventions

- The *Static* property of an attribute or operation corresponds to the class keyword.
- The *Fixed* property of a parameter corresponds to the const keyword.
- The value of *inout* for the *Kind* property of a parameter corresponds to the Var keyword.
- The value of *out* for the *Kind* property of a parameter corresponds to the Out keyword.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [Delphi Options](#)

10.5.6 Java Conventions

EA supports the round trip engineering of Java including [AspectJ](#) extensions where the following conventions are used.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|---|---|
| static | Class or Interface | The static keyword. |
| annotation | Interface | An annotation type. |
| enum | Attributes within a class stereotyped enumeration | An enumerated option, distinguished from other attributes which have no stereotype. |
| operator | Operation | An operator. |
| property get | Operation | A read property. |
| property set | Operation | A write property. |
| enumeration | Class | An enum type. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|----------------|--|--|
| annotations | Anything | The annotations on the current code feature. |
| dynamic | Class or Interface | The dynamic keyword. |
| generic | Operation | The generic parameters to this operation. |
| parameterList | Parameter | A parameter list with the ... syntax. |
| arguments | Attribute with stereotype "enum" | The arguments that apply to this enumerated value. |
| attribute_name | Operation with stereotype "property get" or "property set" | The name of the variable behind this property. |
| throws | Operation | The exceptions that are thrown by this method. |

Other conventions

- Package statements are generated when the current package is not a [namespace root](#).
- The *Const* property of an attribute or operation corresponds to the final keyword.
- The *Transient* property of an attribute corresponds to the volatile keyword.
- The *Fixed* property of a parameter corresponds to the final keyword.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [AspectJ Conventions](#)
- [Java Options](#)

10.5.6.1 AspectJ Conventions

The following are the conventions used for support of the AspectJ extensions to Java.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|---|
| aspect | Class | An AspectJ aspect. |
| advice | Operation | A piece of advice in an AspectJ aspect. |
| pointcut | Operation | A pointcut in an AspectJ aspect. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|------------|--|---|
| className | Attribute or operation within a class stereotyped aspect | The class(es) this AspectJ intertype member belongs to. |

Other conventions

- The specification of a pointcut are included in the *Behaviour* field of the method.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [Java Conventions](#)

10.5.7 PHP Conventions

EA supports the round trip engineering of PHP 4 and 5 where the following conventions are used.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|-----------------------|
| property get | Operation | A read property. |
| property set | Operation | A write property. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|----------------|--|--|
| final | Operations in PHP 5. | The final keyword. |
| attribute_name | Operation with stereotype "property get" or "property set" | The name of the variable behind this property. |

Common conventions

- An unspecified type is modeled as *var*.
- Methods returning a reference are generated by setting the *Return Type* to *var**.
- Reference parameters are generated from parameters with the parameter *Kind* set to *inout* or *out*.

PHP 5 Conventions

- The final class modifier is corresponds to from the *Is Leaf* property.
- The abstract class modifier is corresponds to the *Abstract* property.
- Parameter type hinting is supported by setting the *Type* of a parameter.
- The value of *inout* or *out* for the *Kind* property of a parameter corresponds to a reference parameter.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [PHP Options](#)

10.5.8 Python Conventions

EA supports the round trip engineering of Python where the following conventions are used.

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|------------|-------------------|---|
| decorators | Class, Operation | The decorators applied to this element in the source. |

Other conventions

- Model members with *Private Scope* correspond to code members with two leading underscores.
- Attributes are only generated when the *Initial* value is not empty.
- All types are reverse engineered as *var*.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [Python Options](#)

10.5.9 VB.Net Conventions

EA supports the round trip engineering of Visual Basic .Net where the following conventions are used. Earlier versions of [Visual Basic](#) are supported as a different language.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|--|
| module | Class | A module. |
| import | Operation | An operation to be imported from another library. |
| property | Operation | A property possibly containing both read and write code. |
| event | Operation | An event declaration. |
| operator | Operation | An operator overload definition. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-----------------|--------------------------------------|---|
| partial | Class, Interface | The Partial keyword. |
| shadows | Class, Interface, Operation | The Shadows keyword. |
| Shared | Attribute | The Shared keyword. |
| delegate | Operation | The Delegate keyword. |
| Overloads | Operation | The Overloads keyword. |
| Overrides | Operation | The Overrides keyword. |
| NotOverrideable | Operation | The NotOverrideable keyword. |
| MustOverride | Operation | The MustOverride keyword. |
| Implements | Operation | The implements clause on this operation. |
| Handles | Operation | The handles clause on this operation. |
| Lib | Operation with stereotype "import" | The library this import comes from. |
| Alias | Operation with stereotype "import" | The alias for this imported operation. |
| Charset | Operation with stereotype "import" | Corresponds to the character set clause for this import. One of the values Ansi, Unicode or Auto. |
| attribute_name | Operation with stereotype "property" | The name of the variable behind this property. |
| readonly | Operation with stereotype "property" | This property only defining read code. |
| writable | Operation with stereotype "property" | This property only defining write code. |
| parameterArray | Parameter | A parameter list using the ParamArray keyword. |

Other conventions

- Namespaces are generated for each package below a [namespace root](#).
- The *Is Leaf* property of a class corresponds to the NotInheritable keyword.
- The *Abstract* property of a class corresponds to the MustInherit keyword.
- The *Static* property of an attribute or operation corresponds to the Shared keyword.
- The *Abstract* property of an operation corresponds to the MustOverride keyword.
- The value of *in* for the *Kind* property of a parameter corresponds to the ByVal keyword.
- The value of *inout* or *out* for the *Kind* property of a parameter corresponds to the ByRef keyword.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [Visual Basic Conventions](#)
- [VB.Net Options](#)

10.5.10 Visual Basic Conventions

EA supports the round trip engineering of Visual Basic 5 and 6 where the following conventions are used. [Visual Basic .Net](#) is supported as a different language.

Stereotypes

| <i>Stereotype</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|-------------------|-------------------|-------------------------|
| with events | Attribute | The WithEvents keyword. |
| global | Attribute | The Global keyword. |
| property get | Operation | A property get. |

| | | |
|--------------|-----------|---|
| property set | Operation | A property set. |
| property let | Operation | A property let. |
| import | Operation | An operation to be imported from another library. |

Tagged values

| <i>Tag</i> | <i>Applies to</i> | <i>Corresponds To</i> |
|----------------|--|--|
| New | Attribute | The New keyword. |
| attribute_name | Operation with stereotype "property get", "property set" or "property let" | The name of the variable behind this property. |
| Lib | Operation with stereotype "import" | The library this import comes from. |
| Alias | Operation with stereotype "import" | The alias for this imported operation. |

Other conventions

- The value of *in* for the *Kind* property of a parameter corresponds to the ByVal keyword.
- The value of *inout* or *out* for the *Kind* property of a parameter corresponds to the ByRef keyword.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
- [VB.Net Conventions](#)
- [Visual Basic Options](#)

Part

11

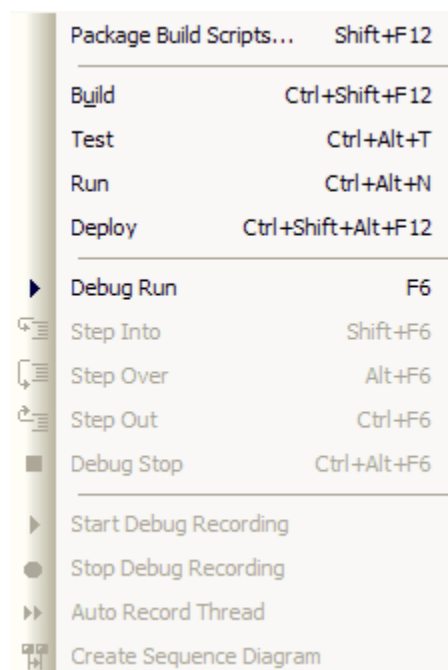
11 Build and Run

Enterprise Architect provides the ability to **build, test, debug, run and execute deployment scripts** - all from within the EA development environment.

This capability is available in the Professional and Corporate versions of EA, and provides developers with tools to **integrate their UML development** and modeling with their **source development** and compilation. With the ability to generate NUnit and JUnit test classes from source classes using MDA Transformations and integrate the test process directly into the EA IDE, it is now possible to integrate UML and modeling into the build/test/execute/deploy process.

In addition to build/test and execute functionality, EA also includes debugging capabilities for .NET and Java. The debuggers being built into EA are specifically designed to allow the capture of stack trace information as a developer or tester "walks through" the executing code. The final stack trace history can then be used to generate Sequence diagrams within EA - converting the actual code execution and calls into visual diagrams. This capability provides an excellent means of managing complexity within a project, of documenting existing code and ensuring that the code written performs as intended by the original architect/developer.

The *Build and Run* menu is accessed from the *Project* menu or from the context menu of a package in the *Project View*. It allows users to create and store custom scripts that specify how to build, test, run and deploy to code associated with a package.



With the appropriate scripts setup EA can

- Call a compiler to build your application, parse the compiler output to and open the internal editor to the location of errors and warnings given.
- Call a unit testing program to run the defined unit tests and parse the output of JUnit or NUnit and open the internal editor to failed tests.
- Debug source code using a customizable interface appropriate to the programming language.

See Also

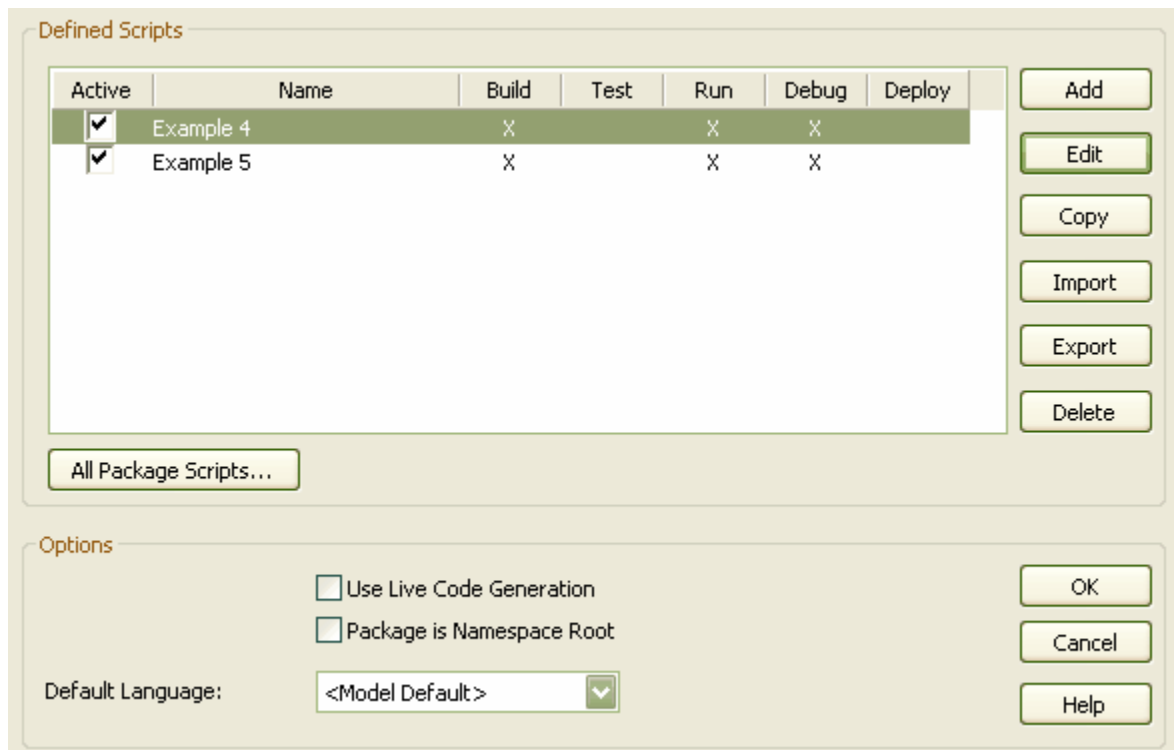
- [Source Code Configuration](#)
- [Build Scripts](#)

- [Testing Scripts](#)
- [Run Scripts](#)
- [Deploy Scripts](#)
- [Managing Compile Scripts](#)
- [Debugging with EA](#)
- [Unit Testing](#)

11.1 Setup for Build and Run

In Enterprise Architect, **any package within the UML Model** can be configured to act as the "root" of a source code project. By setting compilation scripts, xUnit commands, debuggers and other configuration settings for a package, all contained source code and elements can be built, tested or debugged according to the currently active configuration. Each package may have multiple scripts, but only will be active at any one time. This dialog allows you to create and manage those scripts.

To access the dialog, go to the project menu and select *Project | Build and Run | Package Build Scripts*. Alternatively, right click on a package in the project browser, and select *Build and Run | Package Build Scripts*



Scripts are assigned to packages and although a package may have only one active script at any time, you may assign multiple scripts and select from them as needed. The table above shows which script is active for the current package, and whether or not the script contains Build, Test and Run components

- To *create* a new script, click on the *Add* button. This will open the [Build Script Dialog](#).
- To *modify* an existing script, highlight the one you wish to edit, and press the *Edit* button.
- To *copy* a script with a new name, highlight the one you wish to copy, and press the *Copy* button. You will be prompted to enter a name for the new copy. Enter the name in the dialog and press *Ok*. The new copy will appear in the list and can be modified as per usual.
- To *delete* a script, highlight the one you wish to delete, press the *Delete* button, and press *Ok*.
- To *export* your scripts, click the *Export* button to choose the scripts to export for this package.
- To *import* build scripts, click the *Import* button to choose an xml file of the scripts to import.

The *Default Language* drop-down sets the default language for source code generation of all new elements within this package and its descendents.

The *Live Code Generation* feature updates your source code instantly as you make changes to your model. To enable this feature, check the Live Code Generation box.

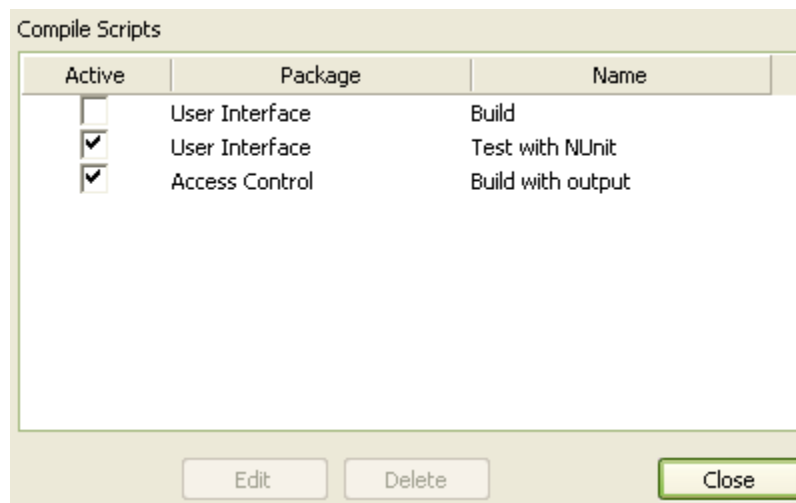
Use Package as Namespace Root sets the source code namespace root (ie. Java/C# namespace) to be the current package. Once you have set a namespace root, code generated beneath this root will add a package declaration at the head of the generated file indicating the current package location.

The *All Package Scripts* button opens a new window which displays all scripts in the current project. (see next section [Managing scripts](#))

Once you have created new scripts or made any changes to existing ones, press *Ok* to confirm the changes, otherwise press *Cancel* to quit the Source Code Configuration dialog without saving any changes.

11.1.1 Managing Scripts

The *All Package Scripts* dialog displays a list of every script in the current project.



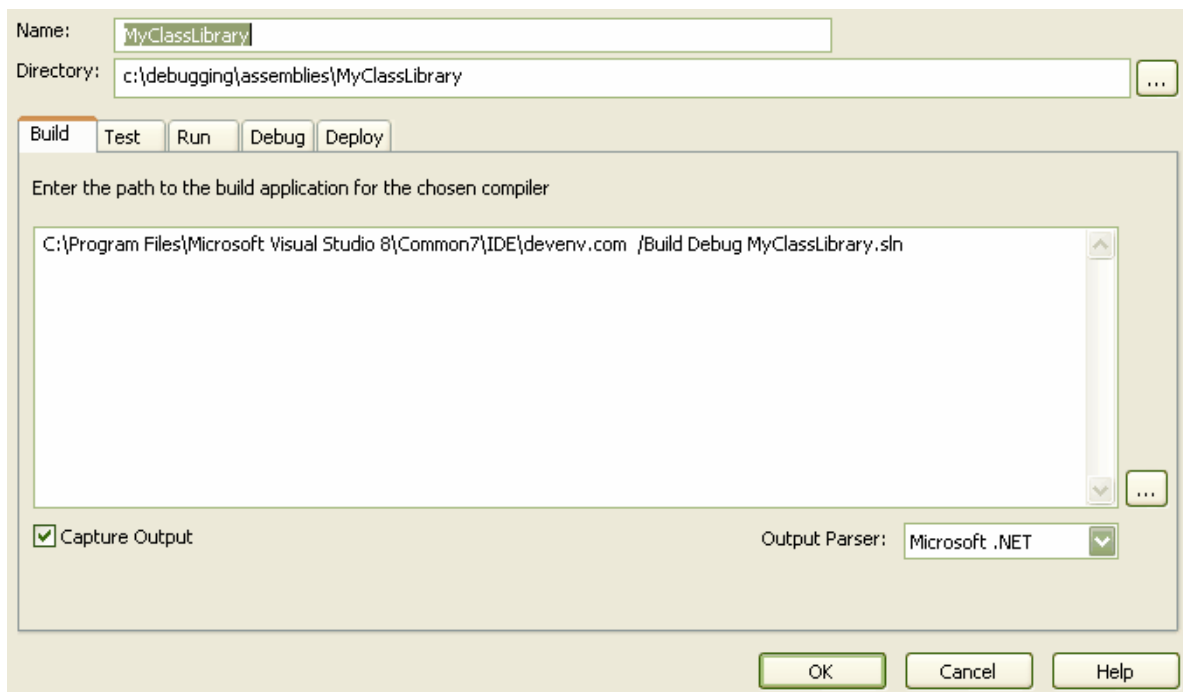
To edit a script double-click on its name, or highlight the script and press the *Edit* button. This will open the [Build Script](#) dialog.

To delete a script, highlight the script and press the *Delete* button. You will be prompted if you are sure you want to delete the script. Press *Yes* to continue and delete the script.

11.1.2 Build Script

The build script dialog allows you to maintain the runtime components of a package. This is where you configure how a package is built, assign any debugger, configure tests and detail how it should be deployed.

Each script requires a name and a working directory.



11.1.2.1 Build Command

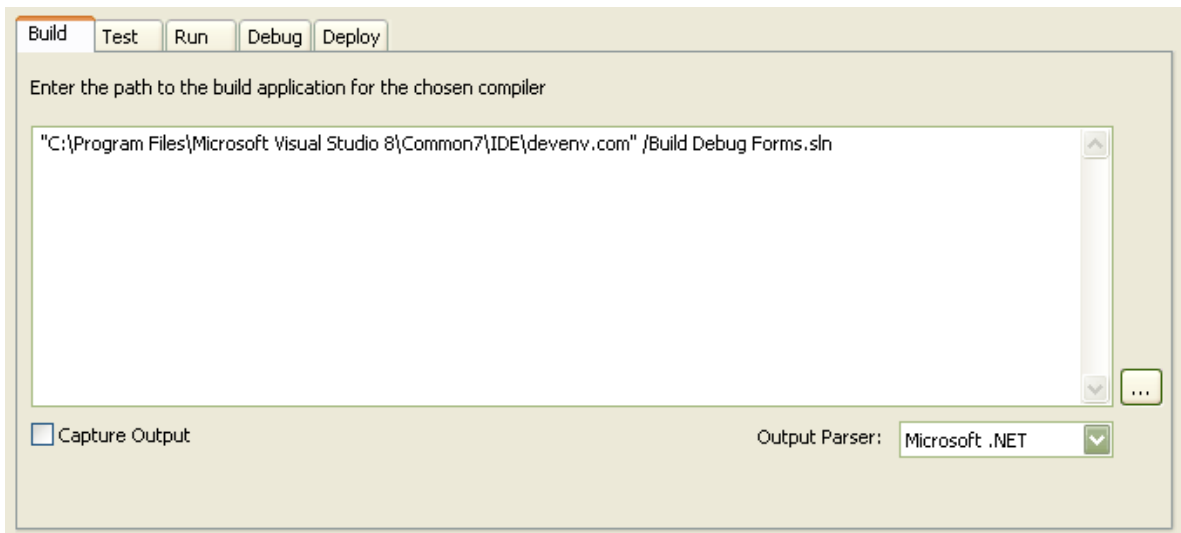
This tab allows you to enter a command for building the current package.

These are the commands that will be executed when you select build from the project menu, *Project | Build and Run | Build*.

Next, write your script in the large text box using the standard Windows Command Line commands. For example, you can specify compiler and linker options, the names of output files etc. The format and content of this section will depend on the actual compiler, make system, linker & etc. you use to build your project. You can also wrap up all these commands into a convenient batch file and call that here instead.

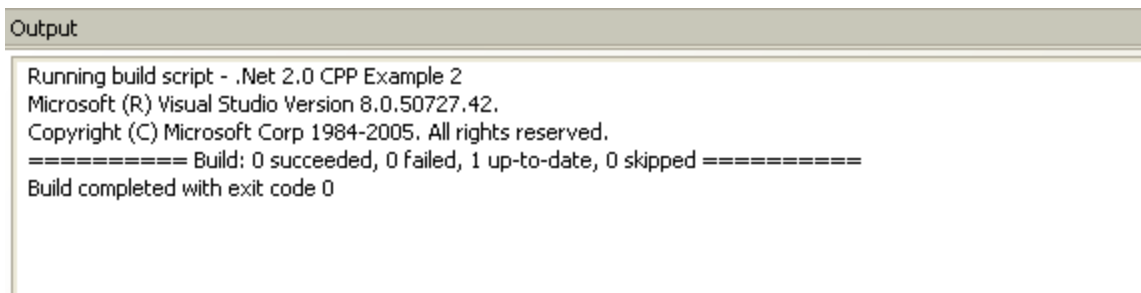
If the capture output box is checked, output from the script will be logged in Enterprise Architect's Output window. This can be activated from *View | Output*

The **Output Parser**: drop down list allows you to define a method for automatically parsing the compiler output. When **Capture Output** is checked, Enterprise Architect will parse the output of the compiler, so that by clicking on an error message in the output window, the user will be taken directly to the corresponding line of code.



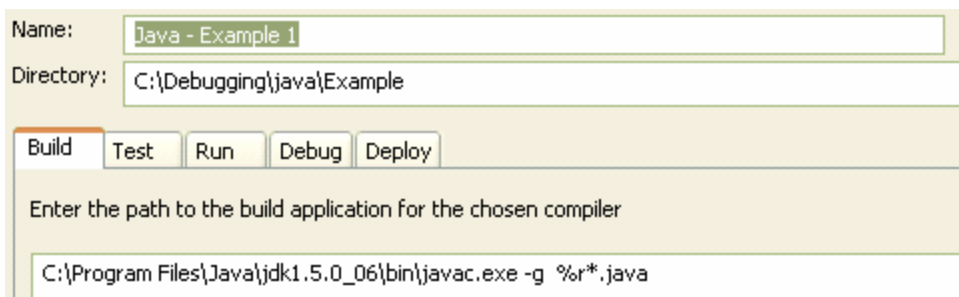
Note: The command listed in this field is executed as if from the command prompt. As a result, if the executable path or any arguments contain spaces, they must be surrounded in quotes.

When you run the compile command inside EA, output from the compiler is piped back to the output window and displayed as in the image below: By double clicking on error lines, EA will load the appropriate source file and position the cursor on the line where the error has been reported.



11.1.2.1.1 Recursive Builds

For any project you can apply the command entered in the build script to all sub folders of the initial directory by specifying the token %r immediately preceding the files to be built. The effect of this is that EA will iteratively replace the token with the any subpath found under the root and execute the command again.



The output from this Java example is shown below

```

Output
Running build script - Java - Example 1
error: cannot read: *.java
1 error
c:\debugging\java\example Build completed with exit code 1
error: cannot read: common\*.java
1 error
c:\debugging\java\example\common Build completed with exit code 1
c:\debugging\java\example\common\draw Build completed with exit code 103
c:\debugging\java\example\common\toolbars Build completed with exit code 0
c:\debugging\java\example\source Build completed with exit code 0

```

Note: The path being built is displayed along with the exit code. Also, some errors are presented. This is due to there being no java source files in some of the sub directories.

11.1.2.2 Test Command

Here you can create a command for performing unit testing on your code. The command is entered in the text box using the standard Windows Command Line commands. A sample script would contain a line to execute the testing tool of your choice, with the filename of the executable produced by the Build command as the option.

To execute this test select it from the project menu **Project | Build and Run | Test**

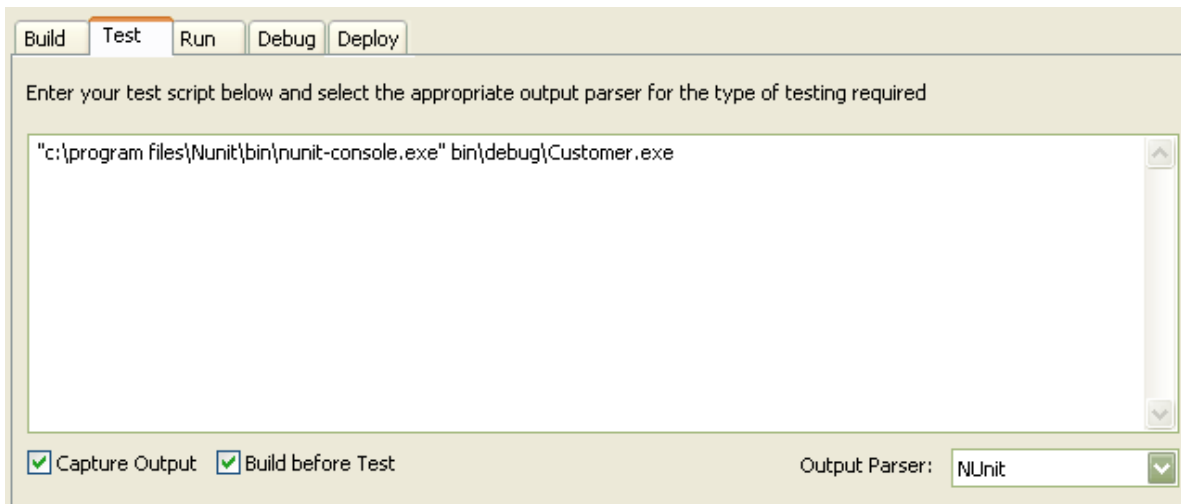
Testing could be integrated with any test tool using the command line provided - but in these examples we have shown how to integrate NUnit and JUnit testing with your source code. EA provides an inbuilt MDA Transform from source to Test Case, plus the ability to capture xUnit output and use it to go directly to a test failure. xUnit integration with your model is now a powerful means of delivering solid and well-tested code as part of the complete model-build-test-execute-deploy life-cycle.

Note: NUnit and JUnit must be downloaded and installed prior to their use - EA does not include these products in the base installer.

The **Capture Output** option allows EA to show the output of the program in the Output Window, while the Output Parser option specifies what format output is expected. When parsing is enabled, double-clicking on a result in the Output window will open the corresponding code segment in EA's code window.

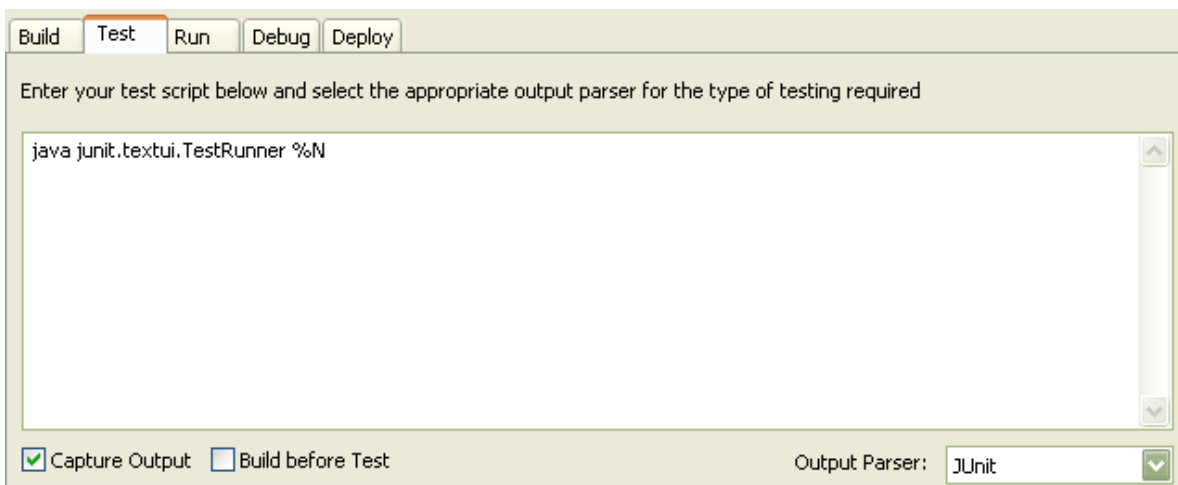
Checking the **Build before Test** box ensures that the package is recompiled each time you run the test.

Two example test scripts are included below. The first is an NUnit example and shows Build before Test checked. As a result every time the test command is given it will run the build script first.



Note: The command listed in this field is executed as if from the command prompt. As a result, if the executable path or any arguments contain spaces, they must be surrounded in quotes.

The second example is for JUnit. It doesn't have Build before Test checked, so the build script won't be executed before every test, but as a result it may test out of date code. This also shows the use of %N which will be replaced by the fully namespace qualified name of the currently selected class when the script is executed.



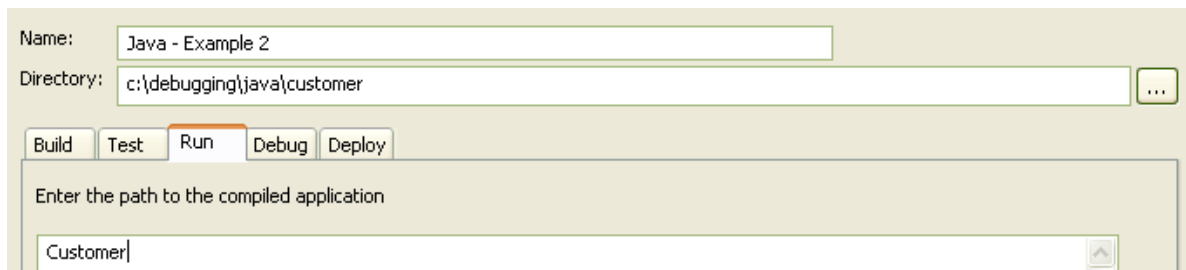
11.1.2.3 Run Command

Here you can enter a command for running your executable.

This is the command that will be executed when you select the *Project | Build and Run | Run* menu item. At its simplest, the script would contain the location and name of the file to be run.

Note that EA provides the ability to start your application normally OR with debugging from the same script. The Build & Run menu has separate options for starting a normal run or a debug run.

The following two examples show scripts configured to run a .Net and a Java application in EA.

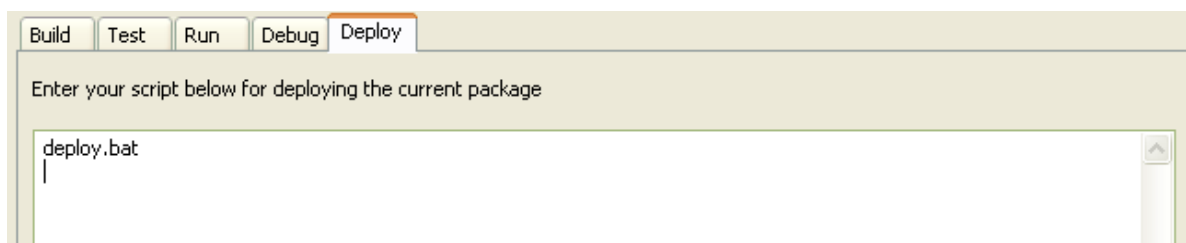


Note: The command listed in this field is executed as if from the command prompt. As a result, if the executable path or any arguments contain spaces, they must be surrounded in quotes.

11.1.2.4 Deploy Command

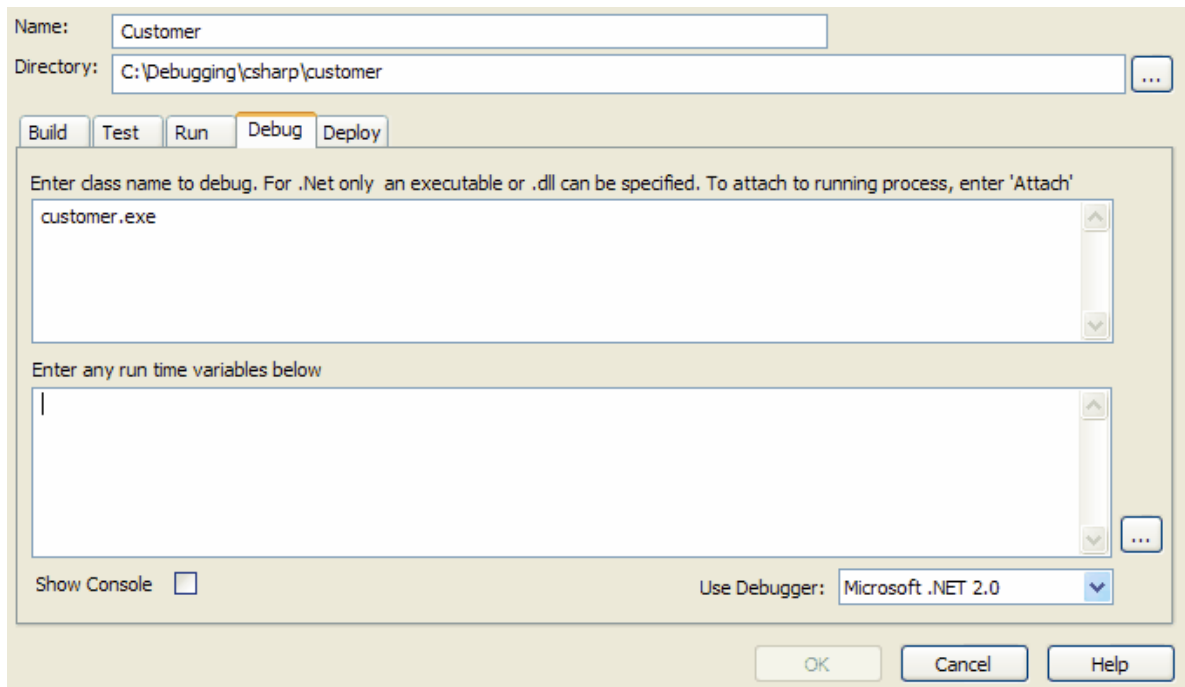
This section allows you to create a command for deploying the current package. These are the commands that will be executed when you select **Deploy** from the project menu, *Project | Build and Run*.

Write your script in the large text box using the standard Windows Command Line commands.



11.1.2.5 Debug Command

This dialog allows you configure how you wish to debug you Application within Enterprise Architect. There are three fields. Details on how to complete these fields for the variety of debugging scenarios is explained in the following sections.



11.1.2.5.1 Java

Enter class name to debug:

Enter the fully qualified class name you wish to debug followed by any arguments. This class must have a method declared with the following signature: `public static void main(String[]);` The debugger will attempt to call this method on the class you name. In the example below, three parameters 1 2 3 will be passed to this method.

Enter any runtime variables below:

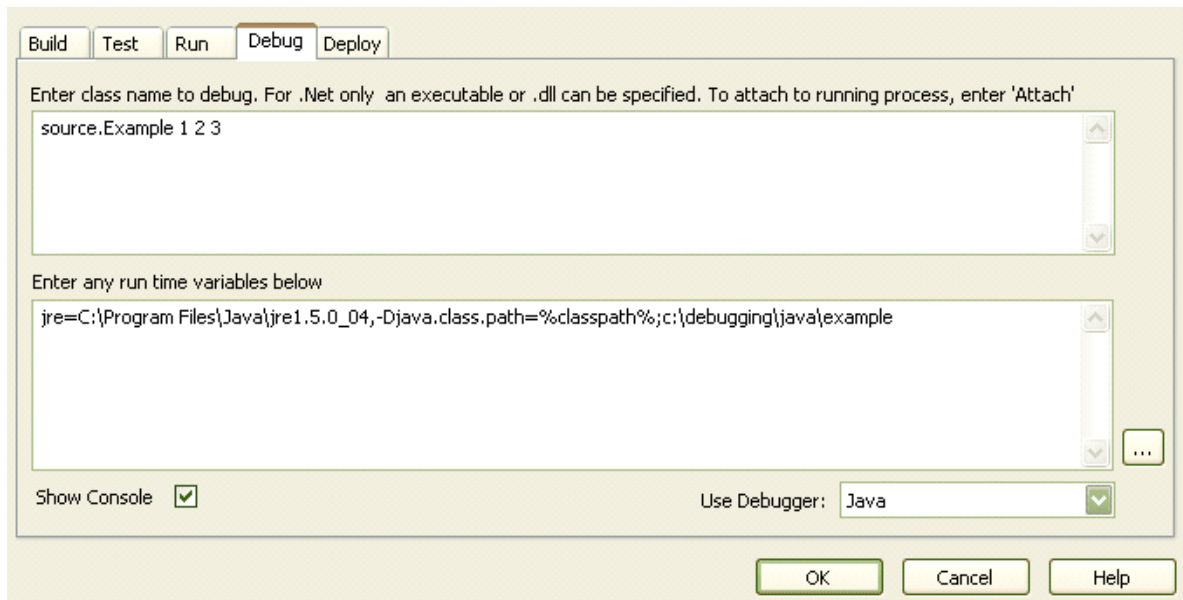
This field is where you supply any command line options to the Java Virtual Machine. You also must provide a parameter (jre) that is a path to be searched for the `jvm.dll`. This is the dll supplied as part of the Java runtime environment or Java JDK from Sun Microsystems™ (see [Profiling and Debugging](#)). In the example below, a VM will be created with a new class path property that comprises any paths named in the environment variable CLASSPATH plus the single path "c:\debugging\java\example". If no class path is specified, the debugger will always create the VM with a class path property equal to any path contained in the environment variable plus the path entered in the default working directory of this script.

Show Console:

Tick "Show Console" checkbox to create a console window for Java. Otherwise no console window is created.

Use Debugger:

Select **Java** as the debugger



11.1.2.5.1.1 Attach to VM

It is possible to debug a java application by attaching to an existing java process. The java process however requires a specific startup option specifying the Sparx Systems Java Agent. The format of the command line option is shown here.

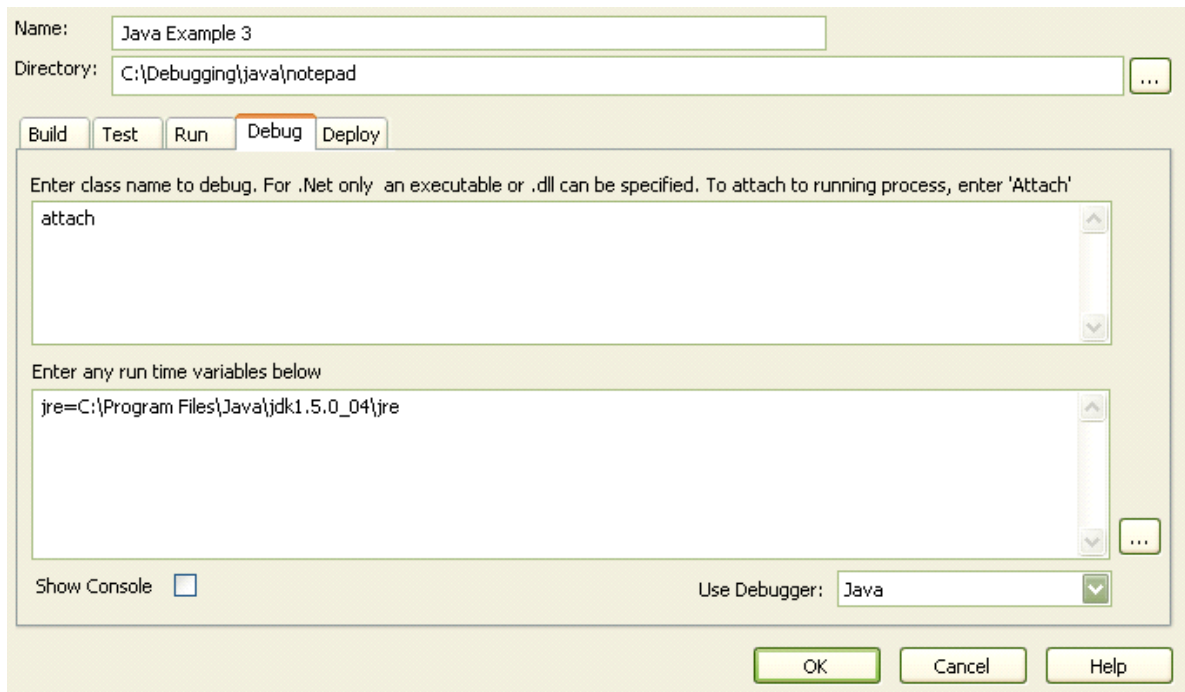
-agentlib:SSJavaProfiler65

you may have to include the path to the DLL in order for Java to locate it.

ie: **java -agentlib:c:\dll\SSJavaProfiler65**

The example below is for attaching to the Notepad example from the Java JDK. The keyword **Attach** is all that needs be entered. This keyword will cause the debugger to prompt you for a process at runtime.

Note: "Show Console" checkbox has no effect when attaching to an existing VM.



11.1.2.5.1.2 Debugging Java Web Servers

The procedure for debugging web servers is exactly the same as attaching to another VM in the previous section.

It is only possible however to debug a Web server that is hosting a VM locally on the same machine as Enterprise Architect.

Remote debugging is not available.

11.1.2.5.2 .NET

Enter class name to debug:

Enter either the full or relative path to the application executable, followed by any command line arguments

Enter any runtime variables below:

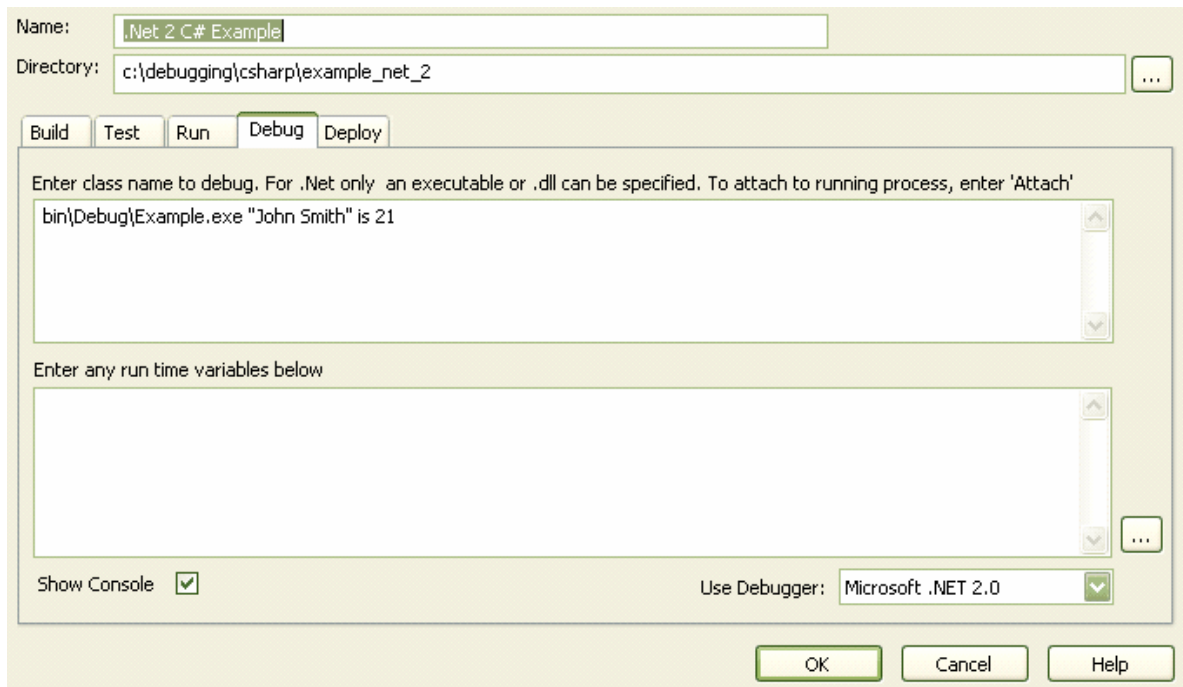
Applicable only where debugging a single .NET Assembly.

Show Console:

For console applications, tick checkbox to create a console window for debugger. Not applicable for attaching to a process.

Use Debugger:

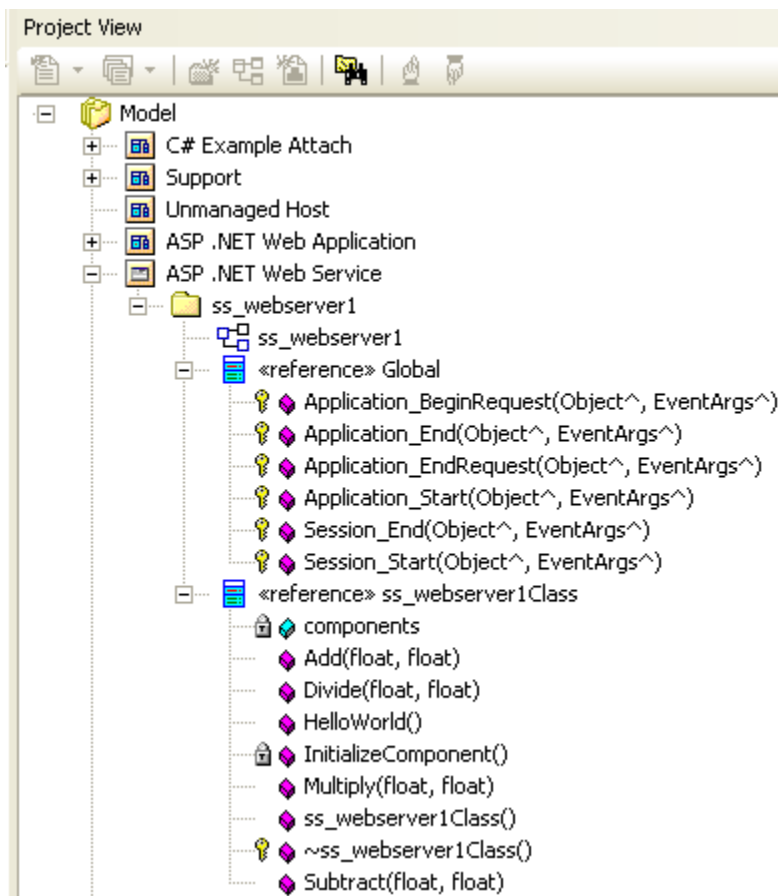
Select debugger to suit .NET Framework under which you application will run.



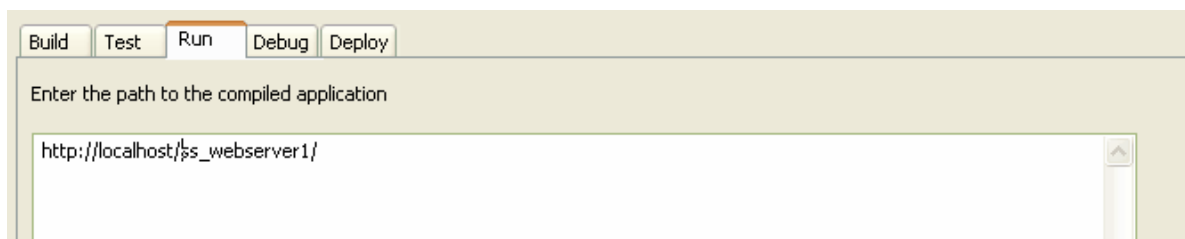
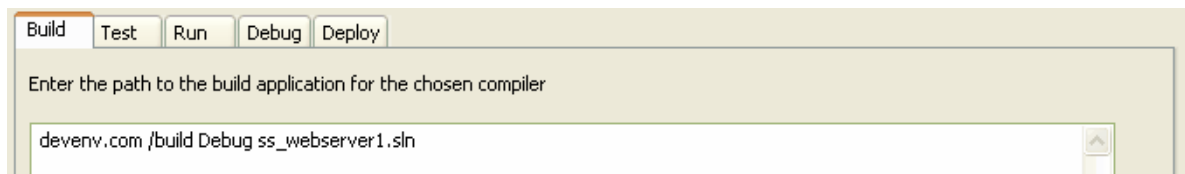
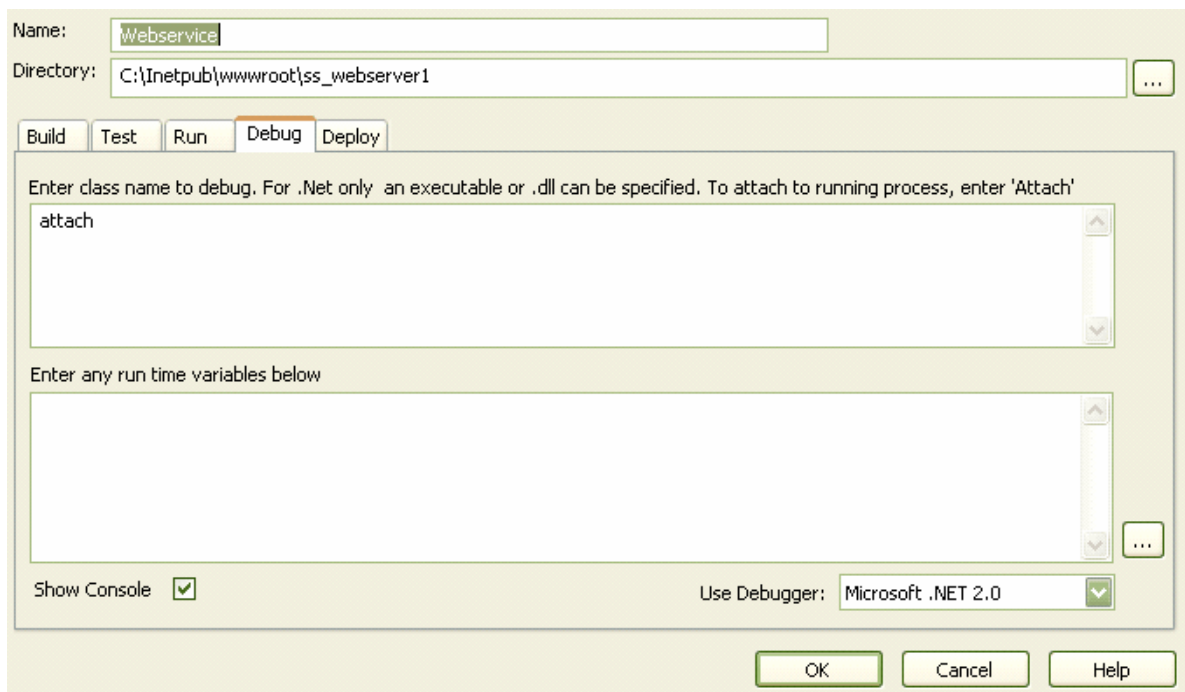
11.1.2.5.2.1 Debugging ASP .NET

Debugging for web services such as ASP requires that the Enterprise Architect debugger is able to attach to a running service. Begin by ensuring that the directory containing the ASP .NET service project has been imported into EA, and if required, the web folder containing the client web pages. If your web project directory resides under the website hosting directory, then you can import from the root and include both ASP code and web pages at the same time.

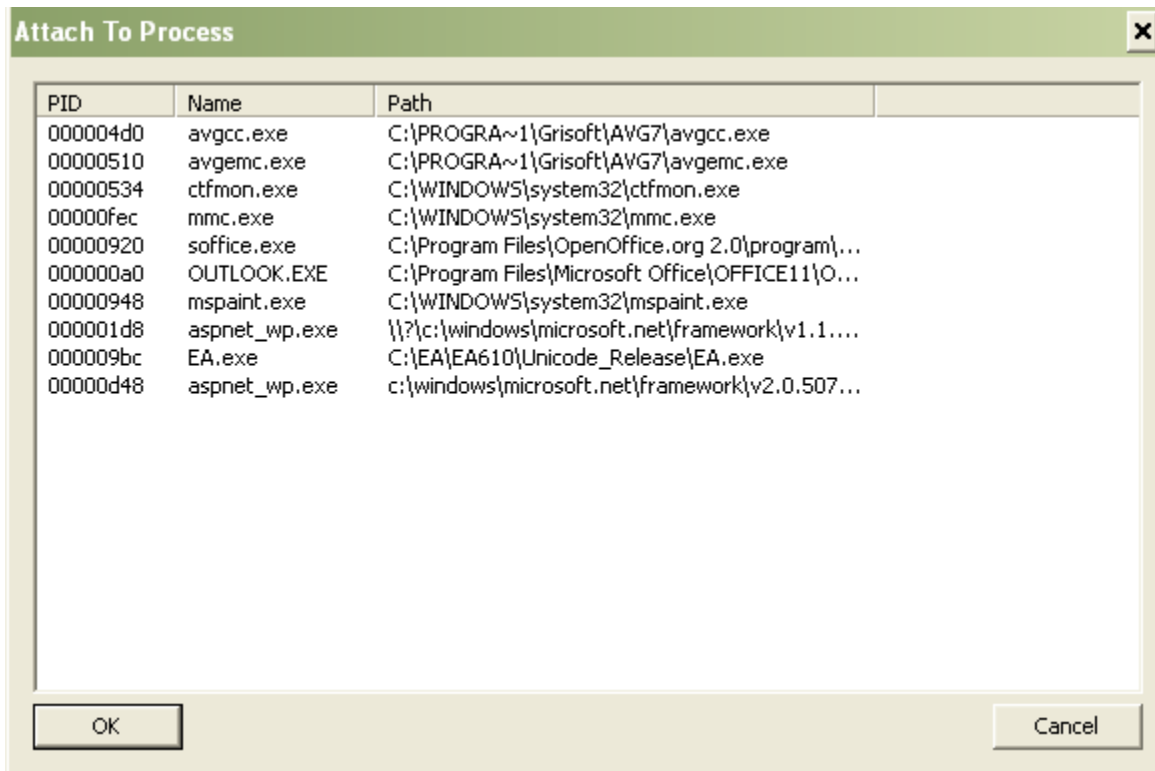
Below is an image showing the project tree of a web service imported into EA.



It is necessary to launch the client first, as the ASP .NET service process may not already be running. Load the client by using your browser. This ensures that the web server is running. The only difference to a debug script for ASP is that you specify the **attach** keyword in your script, like below.



Press the Debug Run key and you should be presented with a dialog similar to the one below.



Notice that two aspnet_wp.exe processes are running, one for version 2 of .NET and one for version 1.

This is because I have test projects that target both. Note also that the name of the process varies across Microsoft operating systems - check the ASP .NET SDK for more information. Select the aspnet_wp.exe process for the version configured (in the web.config file) for your web service, and press OK.

The Debugger Toolbar Stop button should be enabled and any breakpoints should be red, indicating they have been bound. (Note: not all breakpoints may have bound successfully, but if none at all are bound - indicated by being dark red with question marks - something has gone out of sync. Try rebuilding and reimporting source code)

You may set breakpoints at any time in the web server code. You can also set breakpoints in the ASP web page(s) if you imported them.

11.1.2.5.2 Debugging Assemblies

Enterprise Architect version 6.5 now permits debugging of individual assemblies. The assembly will be loaded and a specified method invoked. If the method takes a number of parameters, these can be passed.

Constraints:

Debugging of assemblies is only supported for .NET version 2.

The image below is of a Build Script configured for debugging a .NET assembly. Notice the **runtime variables** field.

This field is a **comma** delimited list of values that must present in the following order

`type_name, method_name, { method_argument_1, method_argument2,....}`

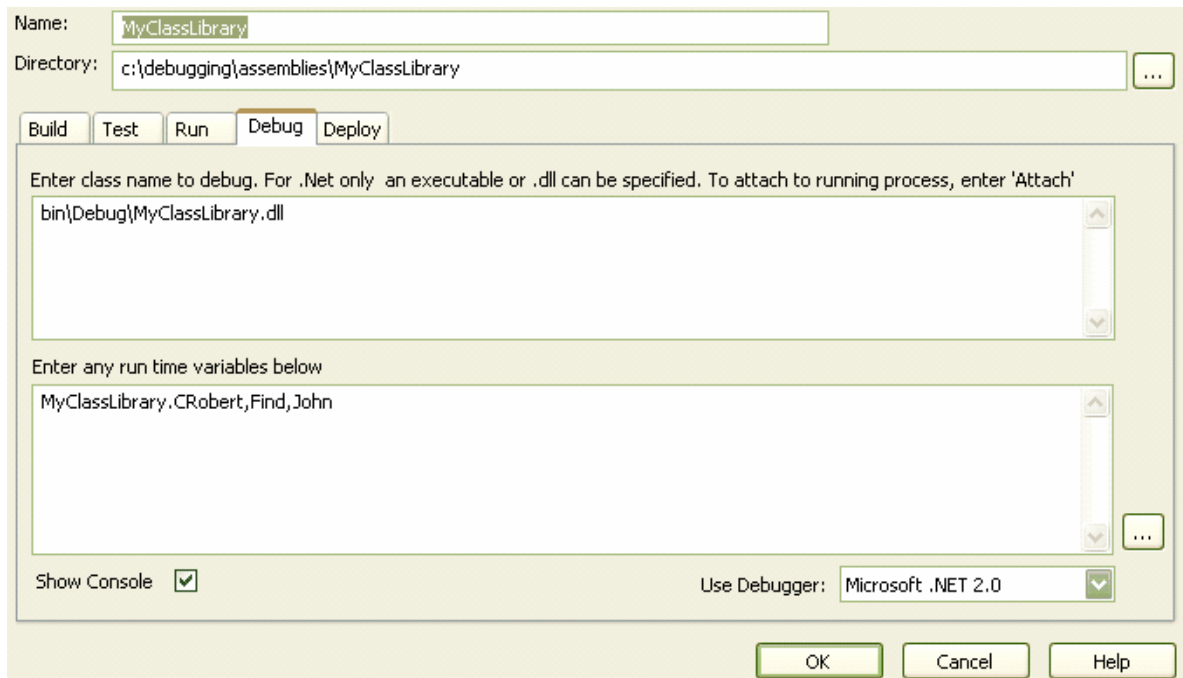
where

`type_name` is the qualified type to instantiate, and

`method_name` is the unqualified name of the method belonging to the type that will be invoked.

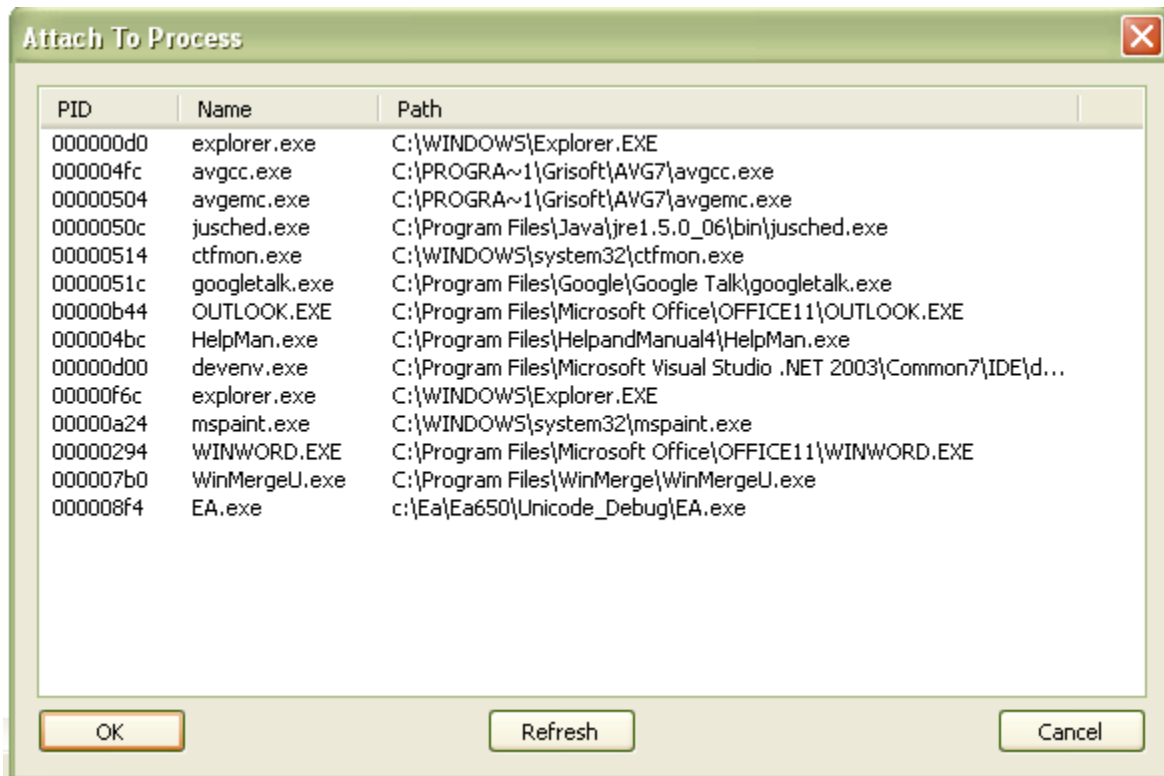
the argument list is optional depending on the method invoked.

The information in this field is passed to the debugger.



11.1.2.5.2.3 Debugging another process

If the Build Script Debug Command is set to the **Attach** keyword, pressing the **Debug Run** button [F6] will prompt you with a list of processes from which you can select.



Once attached to the process, any breakpoints encountered in will be detected by the debugger and the information will be available in the Debug Windows.
To detach from a process, press the **Debug Stop** button

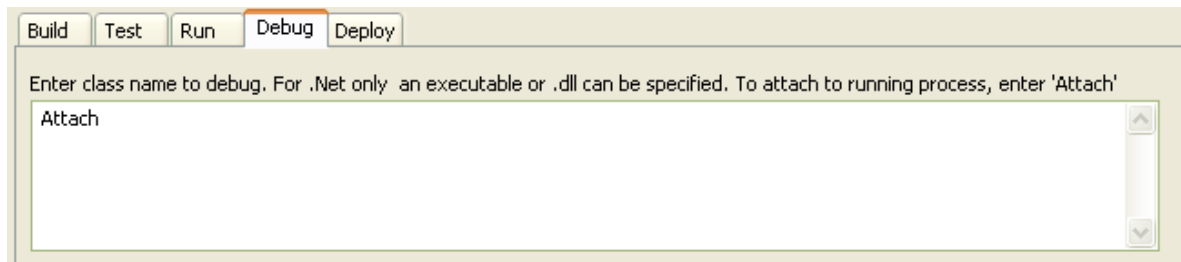
11.1.2.5.2.4 Debugging - COM interop

Enterprise Architect allows you to debug .NET managed code executed using COM either in a Local or In-Process server.

This feature is useful for debugging of Plugins and ActiveX components.

1. Create a package in EA and import the code you wish to debug. See [Code Engineering](#)
2. Ensure the COM component is built with debug information.
3. Create a Script for the Package.
 - a. In the **Debug** tab:

You can elect to either attach to an unmanaged process (specify **Attach** keyword) or specify the path to an unmanaged application that will call your managed code.



4. Add breakpoints in the source code you wish to debug.

Attaching to an unmanaged process:

1. If an In-Process COM server, attach to the client process or
2. If a Local COM Server, attach to the server process.

Pressing the Debug Run button [**F6**] will bring up a list of processes from which you can choose.

Important:

Detaching from a COM interop process you have been debugging will terminate the process. This is a well known issue for Microsoft .NET Framework, and information can be found on it many of the MSDN .NET blogs.

11.1.2.5.2.5 Debugging - CLR Versions

Please note that if you are debugging managed code using an unmanaged application, the debugger may fail to detect the correct version of the CLR to load. You should specify a **config** file if you don't already have one for the debug application specified in the Debug Command of your script. The **config** file should reside in the same directory as your application, and take the format

name.exe.config

where **name** = name of your application

The version of the CLR you should specify should match the version loaded by the managed code invoked by the debuggee.

Sample config file:

```
<configuration>
  <startup>
    <requiredRuntime version="version" />
  </startup>
</configuration>
```

where **version** = the version of the CLR targeted by you plugin or COM code

Reference <http://msdn2.microsoft.com/en-us/library/9w519wzk.aspx>

11.2 Profiling and Debugging

Enterprise Architect allows debugging of source code within EA, using a debug interface appropriate to the source language. Debugging is configured by creating a debug script for the package(s) you wish to test. One of the primary objectives of this feature is to allow you to perform a debug "walk" through executing code and capture your stack trace for direct conversion into a Sequence diagram. This is a great way to document and understand what your program is doing during its execution phase.

11.2.1 System Requirements

Important - please read

Supported Platforms

Enterprise Architect supports debugging on these platforms

.Net

Microsoft™ .NET Framework 1.1 and later
Language support, C#, C++, J#, Visual Basic

Java

Java 2 Platform Standard Edition (J2SE) version 5
J2EE JDK 1.4 and above
Requires previous installation of the Java Runtime Environment and Java Development Kit from Sun Microsystems™

Debugging is implemented through the Java Virtual Machine Tools Interface (JVMTI), which is part of the Java Platform Debugger Architecture (JPDA). The JPDA is included in the J2SE SDK 1.3 and later.

Prerequisites

Creation of Package Build Script and configuration of the Debug Command in that script.

11.2.2 Using the Debugger

The debugging components of EA comprise the Debug Toolbar and a Tabbed Debug Window. To display these, use the keyboard shortcut [*alt+8*] or select **Debug** from the View menu.

If a Debug script has been configured for the currently selected package, then the Run button will be enabled. If no script has been created or is incomplete, all buttons will be disabled and debugging will remain unavailable.

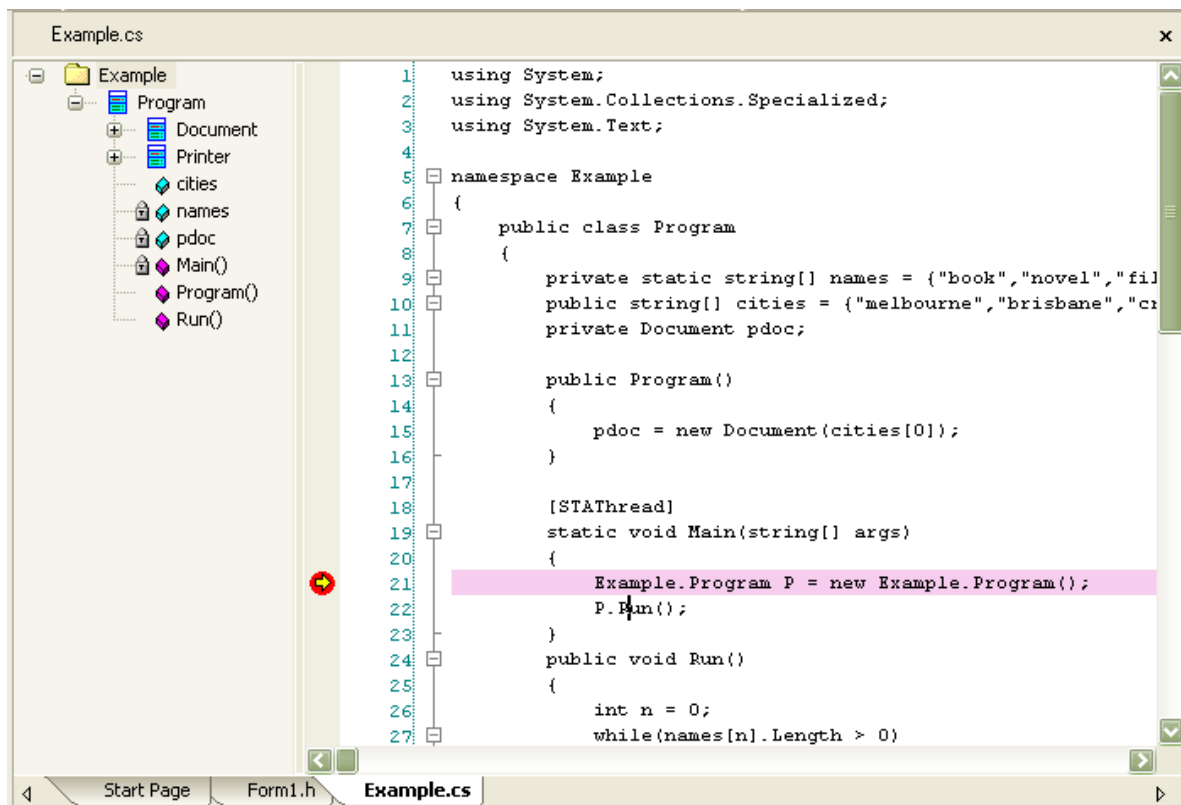
As Build Scripts are linked to a package, selecting a package in the project view will also change the active Build Script, and naturally the target to be debugged. Breakpoints are recorded against the package also, and these will update depending on which package is selected.

Starting The Debugger

To start debugging press the Run button on the toolbar [keyboard shortcut *F6*].

Note, the debugger will only stop if it encounters any breakpoints, so these should be set beforehand.

The figure below shows a view of EA where the debugger has been invoked for a .NET application written in C#.



The yellow arrow  and pink highlight indicate the line of execution.

When the debugger is at a breakpoint the other information will be presented in the various tabs of the Debug Window. The Stack tab will display the stack frames for any thread at a breakpoint, and the Locals tab will display any variables within the scope of the threads current stack frame.



Stopping the Debugger
















To stop debugging press the Stop button  [ctrl+alt+f6]

Note: Under most situations, the debugging will end when the debug process terminates or the java class thread exits, however due to the nature of the Java Virtual Machine, it will be necessary at times for Java developers to Stop the Debugger manually with this button.

11.2.2.1 The Debug Toolbar

The Debug toolbar hosts all the important commands you will need during a debug session. From left to right the buttons are as follows:

-  **Run Debugger (F6)** This button both initiates debugging, and continues execution of a thread that has been halted by a breakpoint.
-  **Stop Debugger (Ctrl+Alt+F6)** Terminates debugging. The debuggee process is immediately halted, and the debug platform closed. All debug output (local variables and stack) are cleared. The workbench is closed

-  **Step Over (Alt+F6)** Step Over causes the debugger to run until the next physical line of code after the current line, or if no further line exists in the method, at the next line in the caller if the method has a caller, ie the stack depth is greater than one.
-  **Step In (Shift+F6)** This command instructs the debugger to try and step over every line of physical code that the thread is executing. Saying that, the debugger will ignore any call to a namespace, function class or method that does not exist in the Model. In other words, only code either generated within Enterprise Architect or imported into the Model will be executed.
-  **Step Out (Ctrl+F6)** This command tells the debugger to run until the current method exits and if the method as a caller (stack depth > 1) stop at the next line of code in the calling method.
-  **Goto Execution Point** This command is only enabled when a thread has encountered a breakpoint. It presents the source code file in an editor window with the current line of code highlighted for the thread that has the current focus (if only one thread is suspended, that thread has focus, otherwise, a thread has focus if it is selected in the Stack tab window. If no thread is selected by the user, the first suspended thread has the focus)
-  **Show Break Points** Displays all current breakpoints. Breakpoints are linked to the project view tree, so changing view by clicking in the tree will change the breakpoints displayed, if any. The only time the breakpoints will not reflect selection changes in the project view is during a debug session.
-  **Delete Break Points** Will delete all breakpoints for the currently active view. If debugging, the breakpoints are removed.
-  **Disable All Break Points** This will disable all break points for the current view. They will not be deleted, so you may re-enable them.
-  **Enable All Break Points** This will re-enable any disabled break points.
-  **Show Local Variables** Will select the **locals** tab, and show all variables within the current scope. Note during a workbench session, this tab is not visible as all variables are displayed in the **Workbench** tab
-  **Show Call Stack** Will select the **Stack** tab and show all currently running threads. The call stack for the any suspended thread will be displayed.
-  **Record Stack Trace** Start recording the trace history. Recording will occur for the currently suspended thread.
The stack history will be cleared ready to accept the new trace history.
Note. this options is also available form shortcut menu in **Stack** Tab.
-  **Auto Record Stack Trace:** Automatically record a stack trace for selected thread. All other threads will be ignored, although entries will appear for all thread creations and terminations. The Step In command will be issued automatically until either a breakpoint is encountered, or the thread terminates. If a breakpoint is encountered, the debugger will halt. To continue automatic recording press any continue command {Run,StepIn,StepOut,StepOver} and Stack Trace recording will be resumed.
Note. this options is also available form shortcut menu in **Stack** Tab.
-  **Stop Recording** Stop recording into the stack trace history.
Note. this options is also available form shortcut menu in **Stack** Tab.
-  **Create Sequence Diagram** This will create a sequence diagram from a recorded Stack Trace history. Give your diagram a name and it will be place in the package for this debug session.
-  Save recorded history to HTML file for viewing in browser
-  Access a menu that allows quick access to relevant commands. The commands are:
- **Build** - run package build scripts
 - **Test** - run package test scripts
 - **Run** - run debug
 - **Create workbench instance**
 - **Package Build Scripts** - Configure package build scripts

Breakpoints

This tab lists any breakpoints placed in the package source code, along with their status (enabled/disabled),

line number, and the physical source file in which they are located.

To set a breakpoint in the code, open the Source Code window using *View | Other Windows | Source Code*, or press *Alt-7*. Find the line in the source where you wish to place the breakpoint and click in the column on the left of the line-numbers. A round circle appears in that column to indicate that a breakpoint has been placed there. (see [Breakpoints](#))

Locals

This display shows the local variables defined in the current code segment, their type and value. (see [Local Variables](#))

Stack

This view shows the position of the debugger in the code. Pressing the > button advances the stack through the code until the next breakpoint is reached. (see [Stack](#))

Output

Displays output from the debugger including any messages output by debugged process, such as writes to standard output.

Modules

This tab displays all the modules loaded during a .Net debug session.

Recording History

This tab is used to record any activity that takes place during a debug session. Once the activity has been logged, Enterprise Architect can use it to create a new Sequence diagram. For more information see [Recording a Debug session](#).

Workbench

This display is a place to create your own variables and invoke methods. (see Workbench)

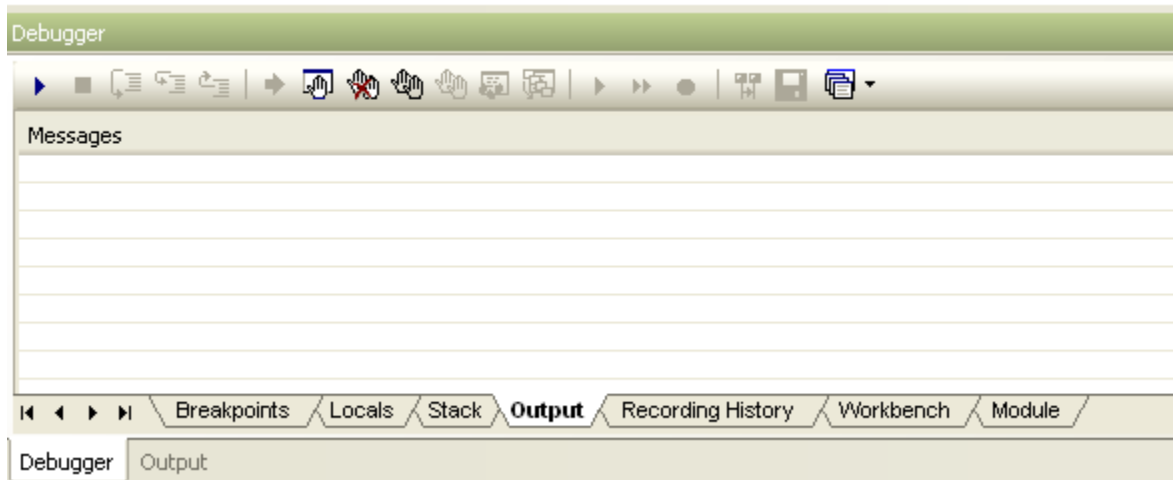
See Also

- [Setting up a Debug session](#)
- [Using the Debugger](#)
- [Recording a Debug session](#)

11.2.2.2 The Debug Window

The Debug window below is available through the Main Menu [*View->Debug*] or keyboard shortcut [*Alt+8*]. It contains seven displays

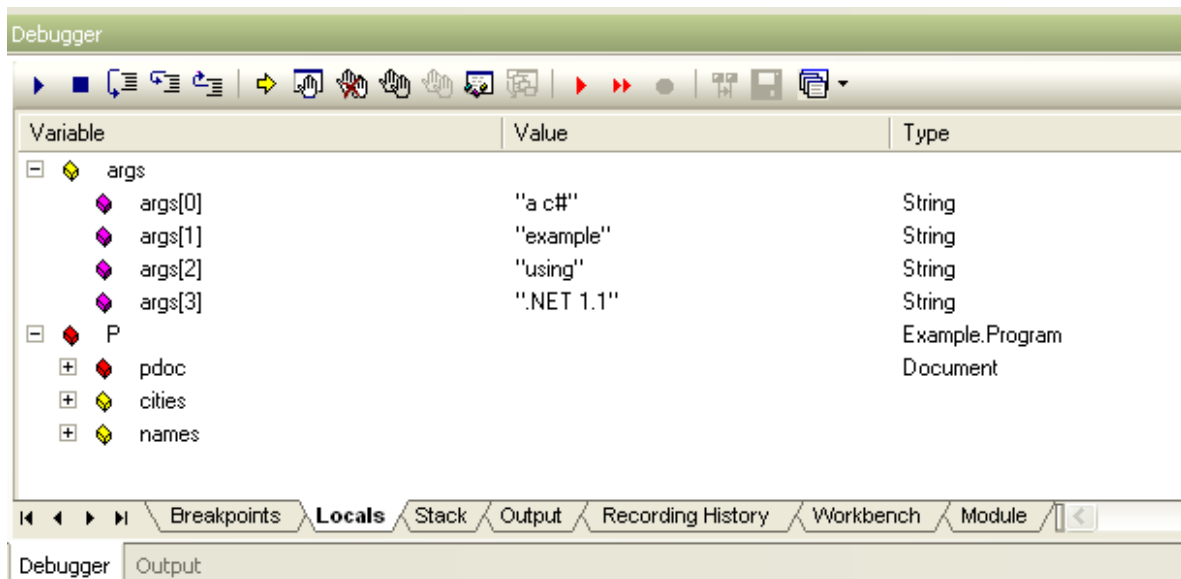
- Breakpoints
- Locals
- Stack
- Output
- Recording History
- Workbench
- Module



11.2.2.2.1 Local Variables

Any time a thread encounters a breakpoint this window will display all the local variables for the thread at its current stack frame.

The value and the type of any in-scope variables are displayed in a tree like the one below.



Local variables are now displayed with icons. The icons from left to right depict the following:-

| | |
|--------|---------------------|
| purple | Elemental types |
| blue | Workbench instances |
| green | Parameters |
| red | Object with members |
| yellow | Arrays |

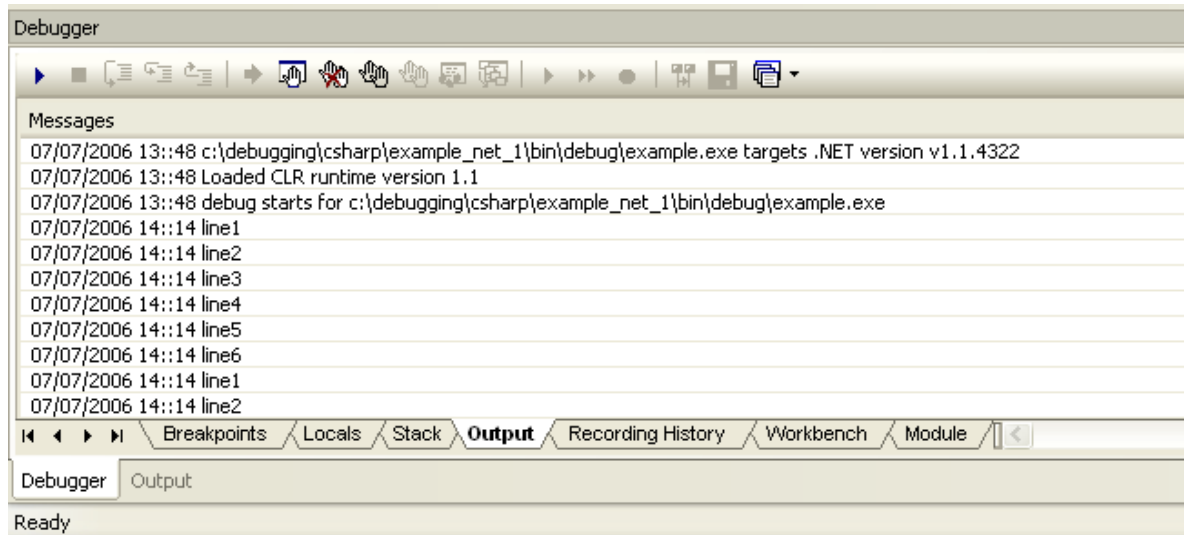
See Also

- [Breakpoints](#)

- [Output](#)
- [Stack](#)
- [Recording History](#)

11.2.2.2.2 Output

During a debug session the debugger will emit messages detailing both startup and termination of session. Details of exceptions and any errors are also output to this window. Any trace messages such as those output using Java *System.out* or .NET *System.Diagnostics.Debug* are also captured and appear here.




See Also

- [Breakpoints](#)
- [Local Variables](#)
- [Stack](#)
- [Recording History](#)

11.2.2.2.3 Stack

The stack view shows all currently running threads. Any thread that is at a breakpoint will display a stack trace.

Pressing the  button continues the thread until the another breakpoint is encountered, or the thread ends.

A yellow arrow is only present where a thread is suspended. It highlights the frame in the stack at which the thread's execution has suspended.

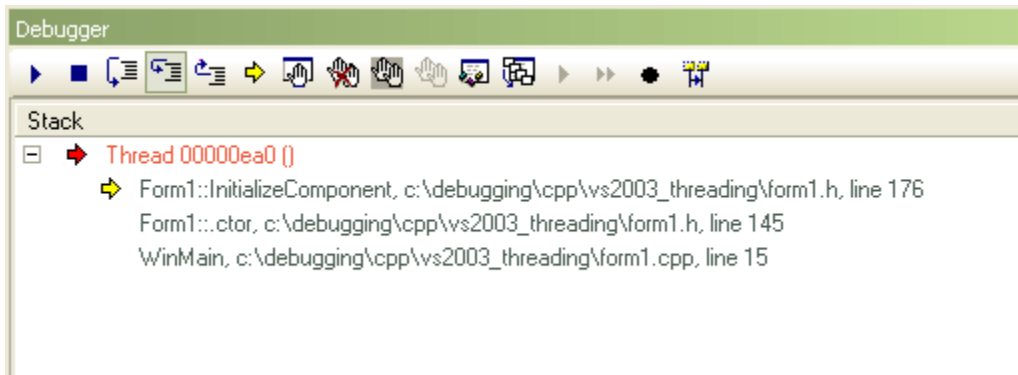
A blue arrows indicate a thread that is running.

A red arrow indicates a thread for which a stack trace history is being recorded.

If multiple threads are suspended, you can click the thread entry to select that thread.

Selecting a thread results in that thread being displayed orange. The Source Code Editor will also change to reflect the current line of code for that thread.

Double-clicking a frame will take you to that line of code in the Editor. Local Variables will also be refreshed for the selected frame.



See Also

- [Breakpoints](#)
- [Local Variables](#)
- [Output](#)
- [Recording History](#)

11.2.2.2.4 Recording History

This tab is used to record any activity that takes place during a debug session. Once the activity has been logged, Enterprise Architect can use it to create a new Sequence diagram. For more information see [Recording a Debug session](#).

Columns

Sequence: unique sequence number
Thread: operating system thread id.
Method: class and method names
Source: filename and line number
Direction: Stack Frame Movement , either Call, Return, Breakpoint, Escape (Escape is used internally when producing sequence diagram to mark end of an iteration)
Iteration: used internally during processing of sequence diagrams fragments.
Depth: stack depth at time of call. Used in generation of sequence diagrams.

Debugger

| Sequence | Thread | Method | Source | Direction | Method | Source |
|----------|------------|----------------|---|-----------|-----------------|------------------|
| 00000000 | 0x00000a0c | | | Call | Program::Main | c:\Debugging\csh |
| 00000001 | 0x00000a0c | Program::Main | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Program::ctor | c:\Debugging\csh |
| 00000002 | 0x00000a0c | Program::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Document::ctor | c:\Debugging\csh |
| 00000003 | 0x00000a0c | Document::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Printer::ctor | c:\Debugging\csh |
| 00000004 | 0x00000a0c | Printer::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Return | Document::ctor | c:\Debugging\csh |
| 00000005 | 0x00000a0c | Document::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Return | Program::ctor | c:\Debugging\csh |
| 00000006 | 0x00000a0c | Program::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Return | Program::Main | c:\Debugging\csh |
| 00000007 | 0x00000a0c | Program::Main | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Program::Run | c:\Debugging\csh |
| 00000008 | 0x00000a0c | Program::Run | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Document::ctor | c:\Debugging\csh |
| 00000009 | 0x00000a0c | Document::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Printer::ctor | c:\Debugging\csh |
| 00000010 | 0x00000a0c | Printer::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Return | Document::ctor | c:\Debugging\csh |
| 00000011 | 0x00000a0c | Document::ctor | c:\Debugging\csharp\example_net_1\Example.cs... | Return | Program::Run | c:\Debugging\csh |
| 00000012 | 0x00000a0c | Program::Run | c:\Debugging\csharp\example_net_1\Example.cs... | Call | Document::Print | c:\Debugging\csh |

Debugger Output

See Also

- [Breakpoints](#)
- [Local Variables](#)
- [Output](#)

- [Stack](#)

11.2.2.2.5 Breakpoints

Breakpoints work in Enterprise Architect much like in any other debugger. Setting a breakpoint notifies the debugger to trap code execution at the point you have specified.

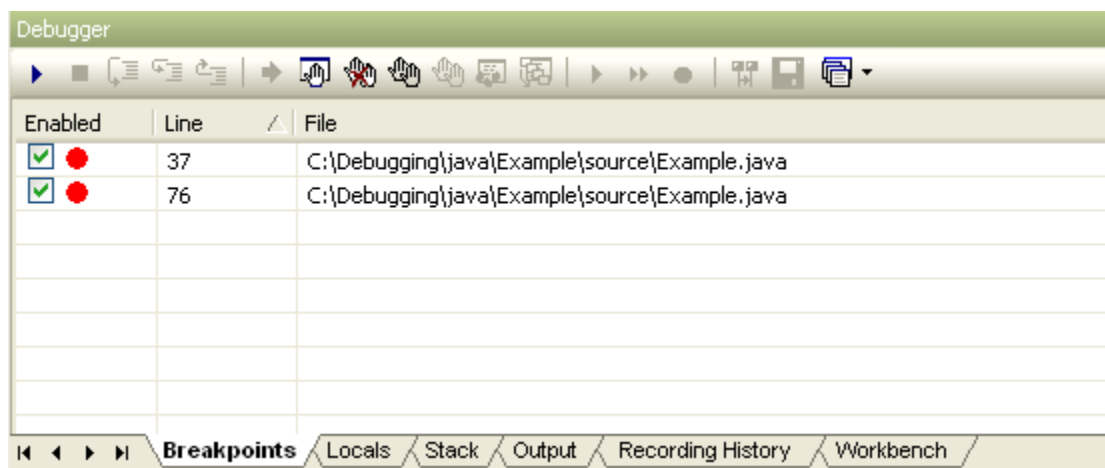
When a breakpoint is encountered by a thread of the application being debugged, the source code will be displayed in an editor window, and the line of code where the breakpoint occurred will be highlighted.

An Enterprise Architect model maintains breakpoints for every package having a Build Script - Debug Command.

Breakpoints are displayed in a tab of the Debug Window **[alt+8]**

Selecting a different package in the tree will update the breakpoints displayed, depending on whether the node selected, or it's parent, has a script attached.

It should be noted that the debugger will not stop automatically. The debugger will run to completion unless it encounters a breakpoint.






Breakpoints are maintained in a file according to the format:

path\prefix_username.brkpt, where:

- path = The default working directory specified in your Build Script.
- prefix = Tree View Node Name / Package name
- username = Host system username of EA user.

11.2.2.2.5.1 States

DEBUGGER STATE

| | <i>Running</i> | <i>Not running</i> |
|---|---------------------------|---------------------------|
|  | <i>active breakpoint</i> | <i>enabled breakpoint</i> |
|  | <i>unbound breakpoint</i> | N/a |
|  | <i>failed breakpoint</i> | N/a |

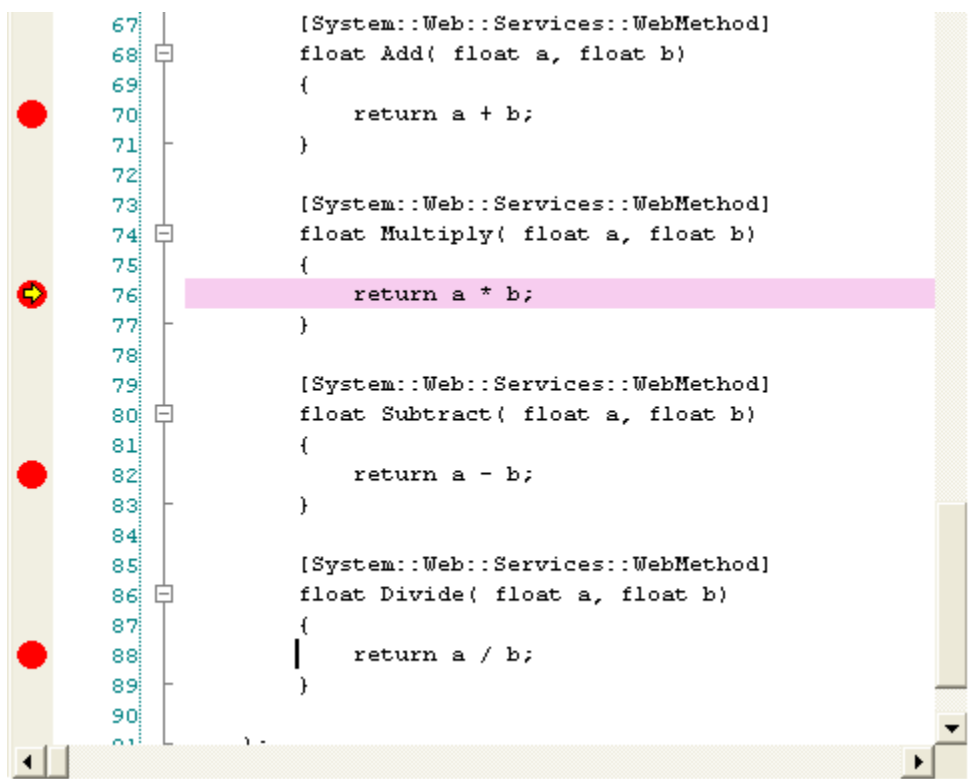


11.2.2.2.5.2 Adding

To set a breakpoint for a code segment, simply open the model code that you wish to debug, find the appropriate line and click in the left margin column. A solid red circle in the margin indicates that a breakpoint has been set at that position. You can also disable a breakpoint, in which case it will be shown as an empty grey circle.

If the code is currently halted at a breakpoint, that point will be indicated by a yellow arrow within the red circle.

To disable a breakpoint, uncheck the box next to it in the *Breakpoints* tab. To delete a breakpoint, either click on the red circle in the left margin, or, select the breakpoint from the *breakpoints* tab and press the delete key.



11.2.2.2.5.3 Deleting

To delete a breakpoint, either click the breakpoint in the Source Code Editor if the breakpoint is enabled, or select the breakpoint in the Breakpoints Tab and press the Delete Key.

You can also delete all breakpoints by using the appropriate button on the debug toolbar.

11.2.2.2.5.4 Disabling / Enabling

To disable a breakpoint, uncheck its checkbox in the Breakpoints Tab. Check it again to enable the breakpoint.

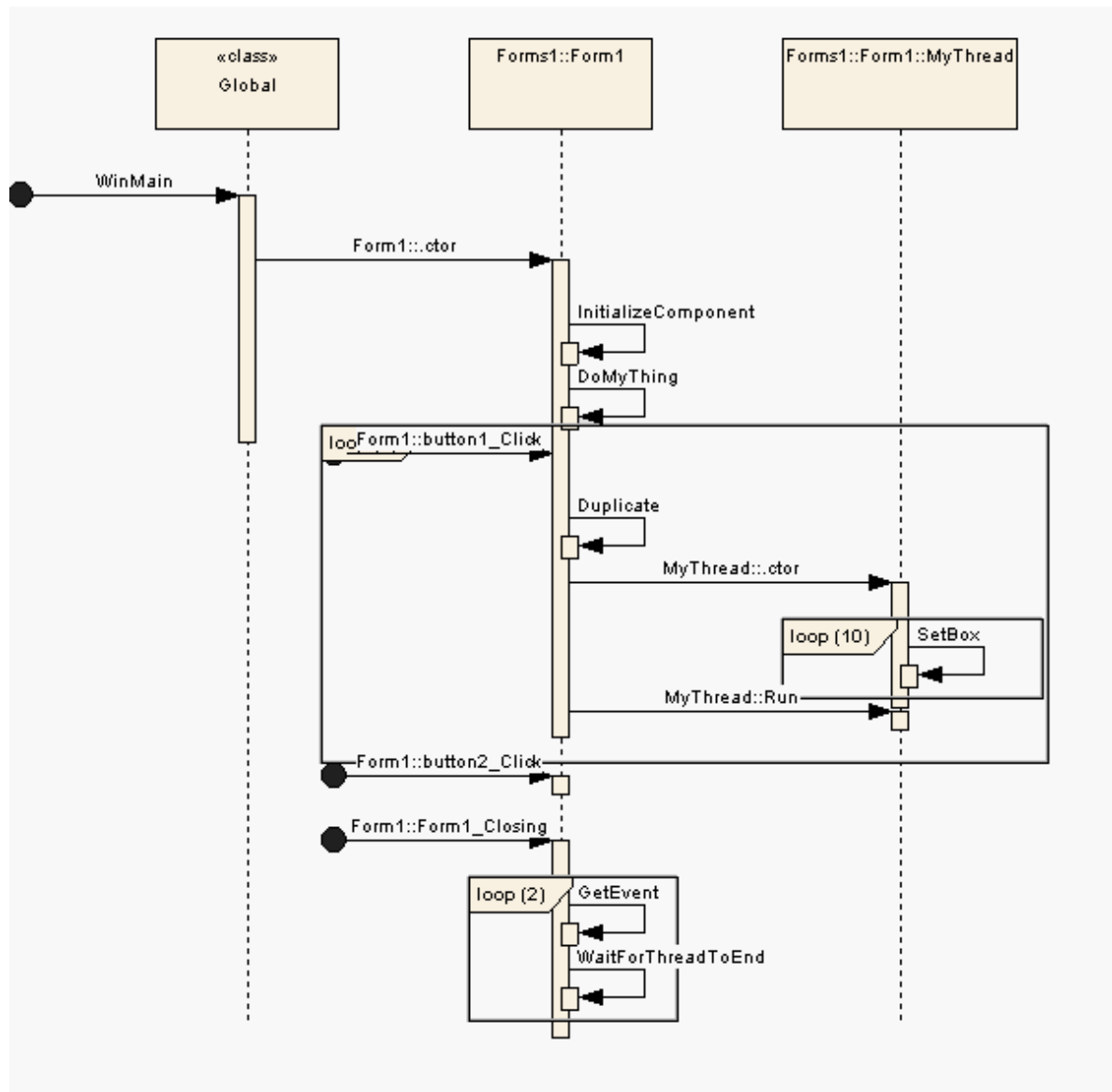
11.2.3 Generating Sequence Diagrams

With Enterprise Architect you can easily create detailed and comprehensive Sequence diagrams from your recorded Debug sessions. EA makes it simple to either manually step through your executing code and record specific execution traces, or let the debugger automatically step through.

Enterprise Architect can take the recorded stack history captured during one of these runs and automatically build the Sequence diagram, including compacting looping sessions for easy reading.

Once you have set up your debugger and can run and debug your application successfully, you can move on to recording the execution.

The following diagram illustrates the kind of diagram you can produce by carefully stepping through your running program or letting EA profile it automatically.



11.2.3.1 Recording and Diagramming a Debug Session

The [debugger](#) gives you the ability to record your debug session and you will be able to create a sequence diagrams from the Stack Trace History.

Recording can only occur for a given thread. The *red record arrows* (record and autorecord) will become enabled whenever a thread is available for recording.

This occurs when a thread encounters a breakpoint, and becomes suspended.

For .Net applications, the first thread to begin execution will automatically stop at the first method entered that is recognised as being part of the current EA model.
For most applications it is typical that you would set breakpoints at positions in the code that you wish to trace.

You may begin recording in either one of two ways.

A. Manual Recording For A Thread

Either 1, click the record button, or 2, select **Stack** Tab, right-click for shortcut menu, (see figure A below), and choose **record**.

Thereafter you must issue debug commands {StepIn, StepOver, StepOut, Stop} manually.
Each time you issue a step command and the thread stack changes, the sequence of execution will be logged.

When you have finished tracing, click the stop button.
The "Generate Sequence Diagram" should be enabled.
Click it to generate.

B. Automatic Recording For A Thread

Either 1, click the **autorecord** button, or 2, select **Stack** Tab, right-click for shortcut menu, (see figure A below), and choose **Auto Record**

After choosing autorecord you will notice that the Stack Trace History, Stack tab and Source Code Editors will dynamically update
to reflect the current execution sequence for the thread.

Note, no other threads are recorded, with the exception of entries for thread creation and termination.

Stack Trace Recording will end when the thread ends, or when you click the Stop button.
The "Generate Sequence Diagram" should be enabled.
Click it to generate.

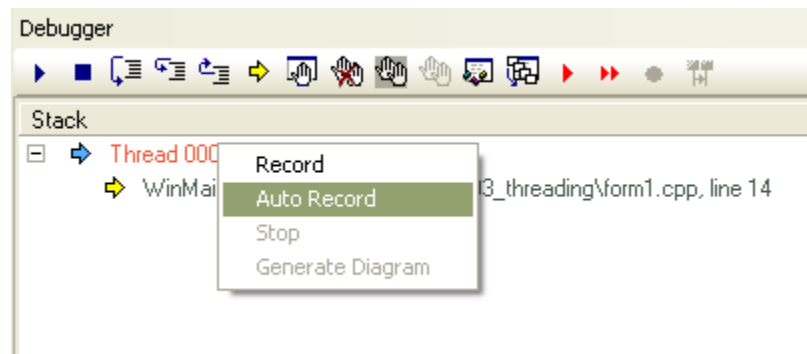
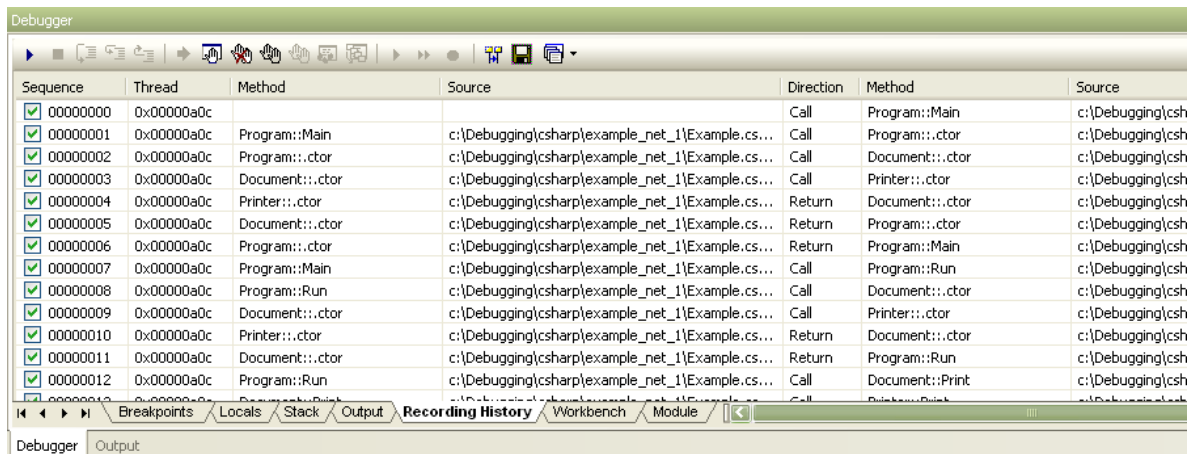


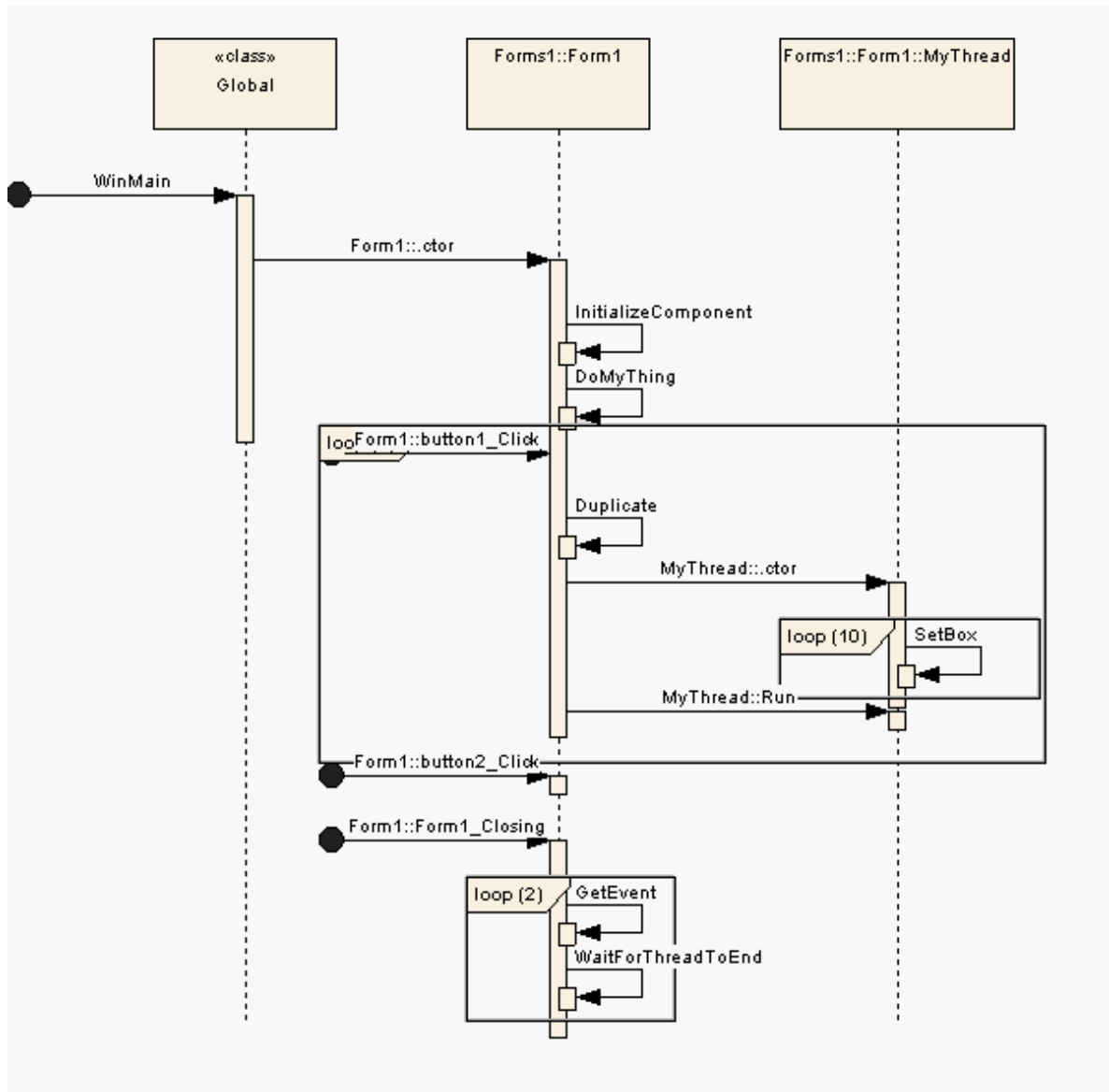
figure A - Stack Tab

This is an example for a debug sequence recorded for a thread executing managed C++ code under Microsoft .Net 1.1







Creating the Sequence Diagram

Once you have built up a stack trace history, you are then able to create a sequence diagram from your results. To do this, press the *Create Sequence Diagram* button on the toolbar and give your diagram a name. Your diagram will be placed under the package that is running your debug session.



Commands

-  Record Stack Trace for Thread
-  Auto Record Stack Trace For Thread
-  Stop Recording
-  Create Sequence Diagram
-  Save recorded history to HTML file for viewing in browser

11.2.4 The Debug Workbench

The Debug Workbench is a tool in Enterprise Architect Debugging enabling users to create their own variables and invoke methods on them. Stack trace can be recorded and Sequence Diagrams produced from the invocation of such methods. It provides a quick and simple way to debug your code.

Mode

The debug workbench operates in two modes.

Idle mode

When the workbench is in idle mode, instances can be created and viewed and their members inspected.

Active mode

When methods are invoked on an instance, the workbench enters Active mode, and the variables displayed will change if the debugger encounters any breakpoints. If no breakpoints are set, then no change in variables will be noticed. The workbench will immediately return to Idle mode.

Logging

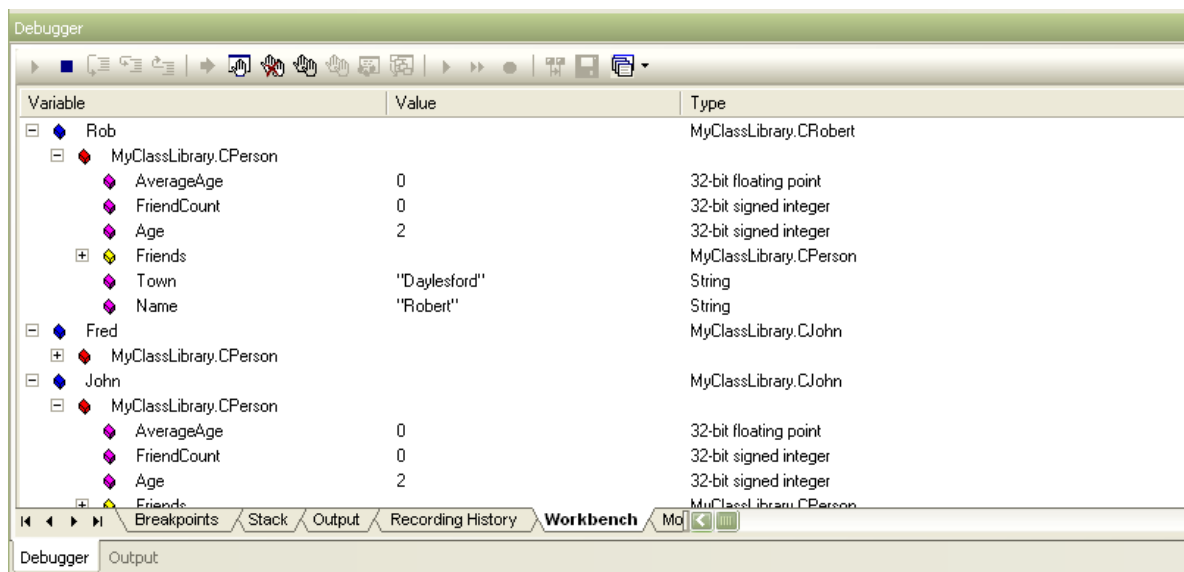
The result of creating variables and the result of calls on their methods is displayed in the Output Tab.

See Also

- [Workbench Variables](#)
- [Invoking Methods](#)

11.2.4.1 Workbench Variables

You can create workbench variables from any class in your model. When you do so, you are asked to name the variable. It will then appear in the Workbench Tab of the Debug window. This window is just like the Local Variables Tab in normal debugging, [hidden during workbench mode]. It shows the variable in a tree control, displaying its type and value and that of any members.



Workbench Requirements

- NET framework version 2 is required to workbench any .NET model.
- The Package from which the variable is created must have a debugger configured. (refer: [Debug Command](#))

Constraints (.NET)

- Members defined as **struct** in managed code are not supported
- Classes defined as **internal** are not supported.

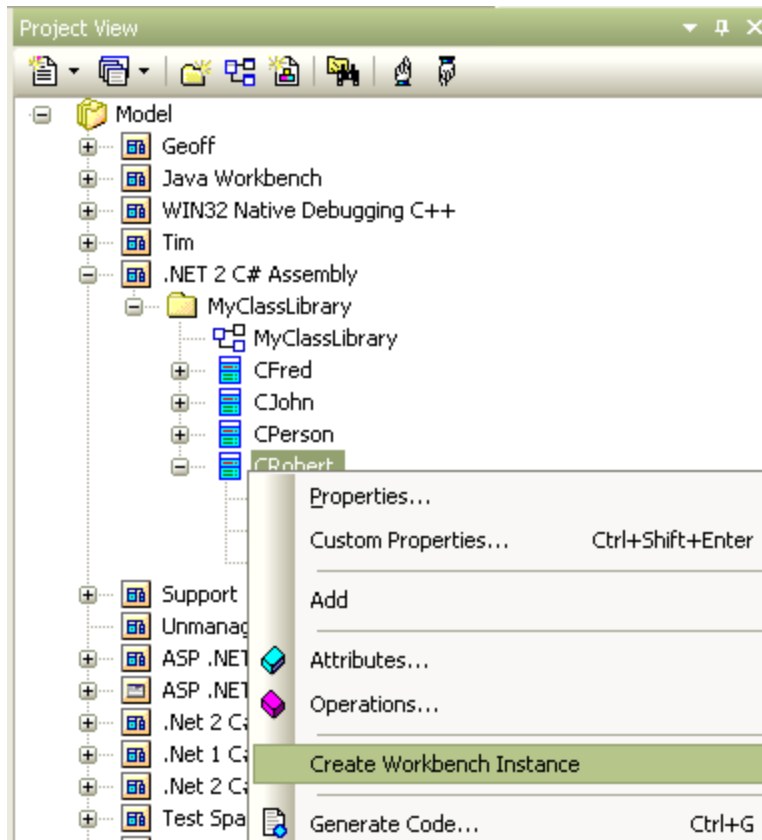
See Also

- [Creating Workbench Variables](#)
- [Deleting Workbench Variables](#)

11.2.4.1.1 *Creating Workbench Variables*

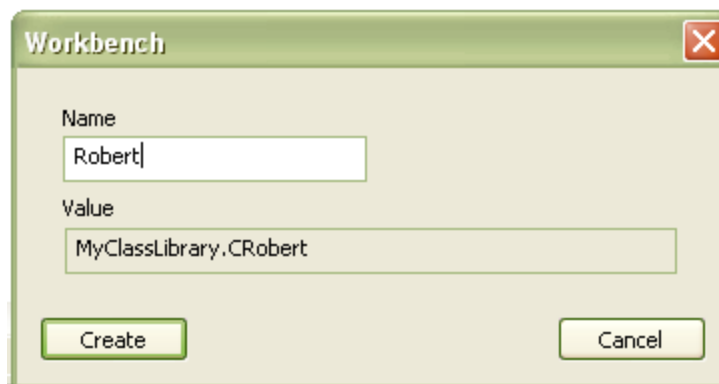
Creating a Workbench Variable

Right click in the Project View on a Class node and select the **Create workbench instance** shortcut menu. [keyboard shortcut **ctrl+shift+j**]
This menu item is also available from within a class diagram.



Naming the Workbench

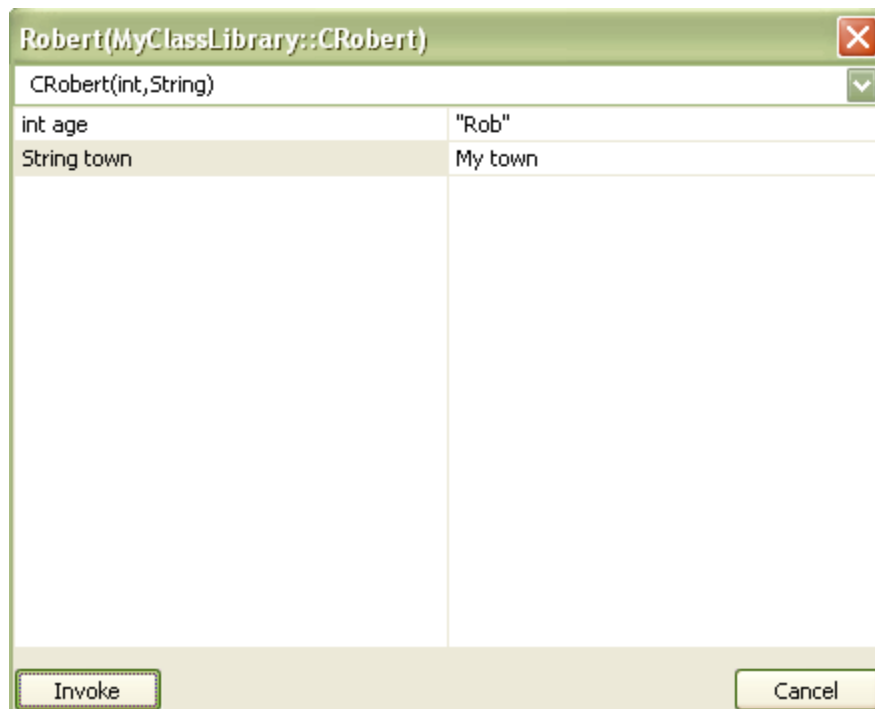
When you elect to create an instance of a type then you will be prompted with the following dialog and requested to name the variable. Each instance name must be unique for the workbench.



Choosing a Constructor

Having given the variable a name, you must now choose which constructor to use. If no constructor is defined or a single constructor only, taking no arguments, it will automatically be invoked.

Otherwise you will be presented with the dialog below. Select the constructor from the drop down list and fill in any parameters.



Arguments

In the dialog above, any parameters required by the constructor can be typed.

- **Literals as arguments**

Text: abc or "abc" or "a b c"
 Numbers: 1 or 1.5

- **Objects as arguments**

If an argument is not a literal then you can supply it in the list only if you have already created an instance of that type in the workbench. You do this by typing the name of the instance as the argument. The debugger will check any name entered in an argument against its list of workbench instances, and will substitute that instance in the actual call to the method.

- **Strings as arguments**

Surrounding strings with quotes is unnecessary as anything you type for a string argument becomes the value of the string eg:

The only time you need to surround strings with quotes is in supplying elements of a string array, or where the string is equal to the name of an existing workbench instance.

```
"A b c"
"a b $ % 6 4 "
A b c d
As 5 7 ) 2 === 4
```

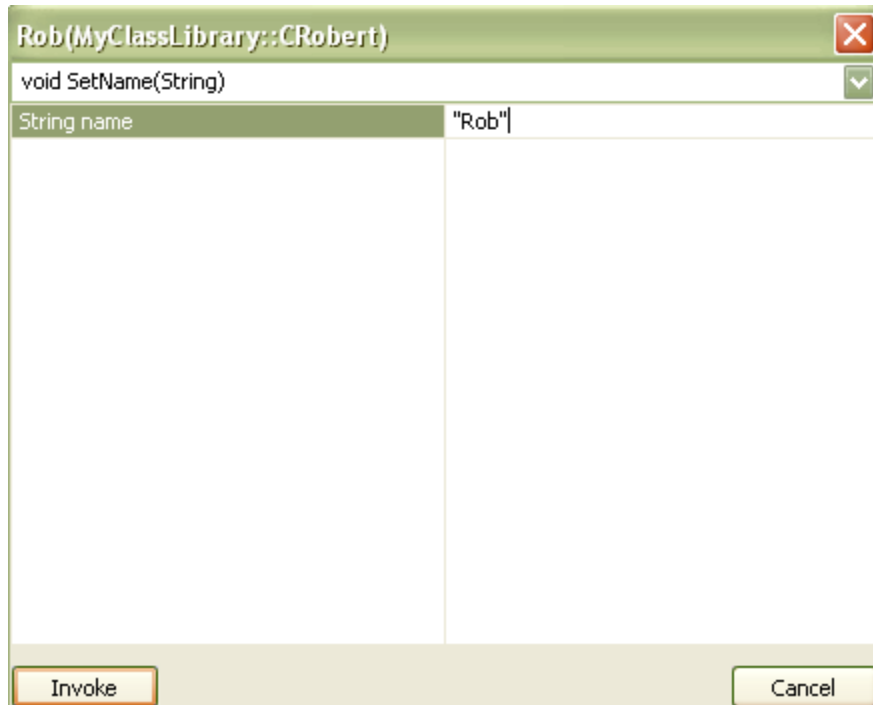
- **Arrays as arguments**

Enter the elements that compose the array separated by commas.

| Type | Arguments |
|----------|--|
| String[] | one,two,three,"a book","a bigger book" |

CPerson[] Tom,Dick,Harry

Note: If you enter text that matches the name of an existing instance, surround it in quotes to avoid the debugger passing the instance rather than a string.



Invoke

Having chosen the constructor and supplied any arguments, pressing the Invoke button will cause the variable to be created.

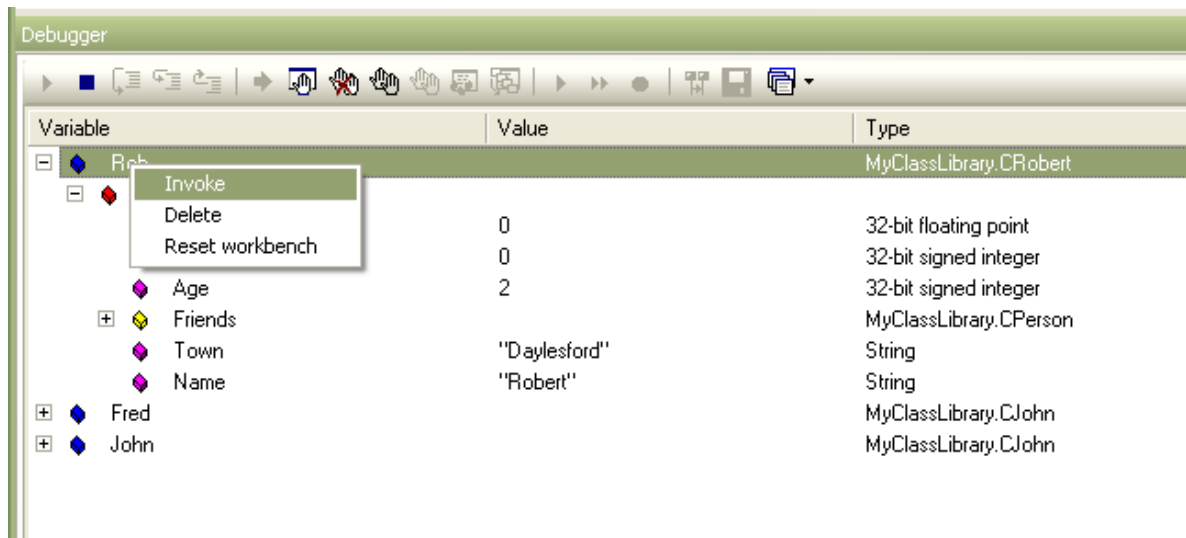
Output confirming this action will be displayed in the Output Tab

11.2.4.1.2 Deleting Workbench Variables

You can delete variables by using the Delete shortcut menu on any instance on the workbench. If all instances are deleted the debugger will be shutdown, and the workbench is closed.

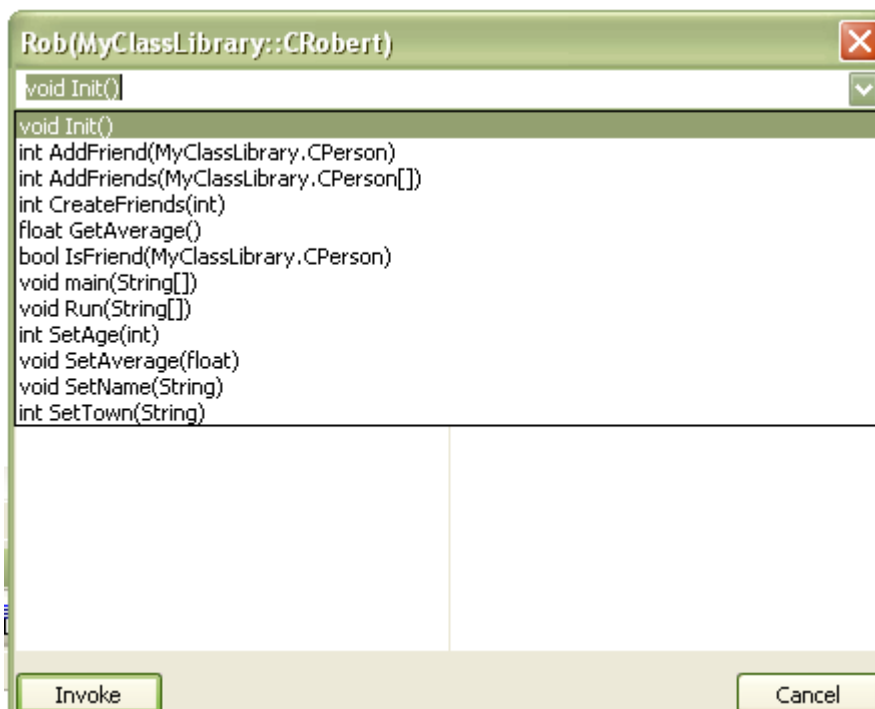
11.2.4.2 Invoking Methods

In the workbench tab, right click the instance on which you wish to execute a method.



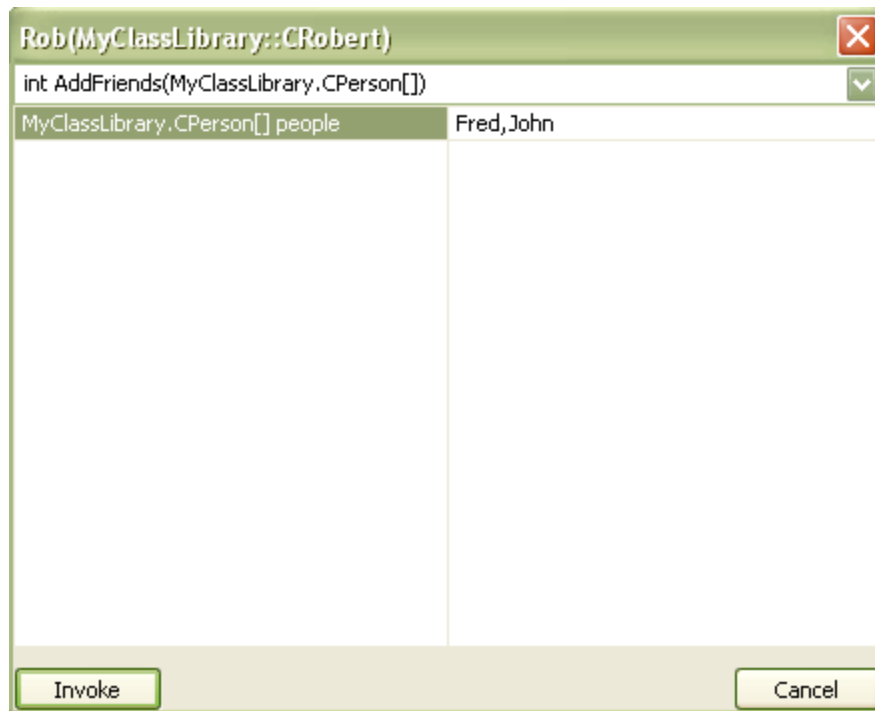
Choose method

A list of methods for the type are presented in a dialog. Select a method from the list. Note, private methods are not available.



Supply arguments

In this example, we have created an instance or variable named Rob of type MyClassLibrary.CRobert. We will invoke a method named AddFriends which takes an array of CPerson objects as its only argument. What we supply to it are the two other Workbench instances Fred and Tom



11.3 Unit Testing

EA supports integration with unit testing tools in order to make it easier to develop good quality software.

Firstly, EA helps you to create test classes with the [JUnit](#) and [NUnit](#) transformations. Then you can [set up](#) a [test script](#) against any package and run it. Finally, all tests results are automatically recorded inside EA.

The following pages give information on how to do this.

See Also

- [JUnit Transformation](#)
- [NUnit Transformation](#)
- [Setting Up Unit Testing](#)
- [Running Unit Tests](#)
- [Recording Test Results](#)

11.3.1 Setting Up Unit Testing

In order to use unit testing in EA, you will first need to set up your unit testing. This happens in two parts, firstly the appropriate tests need to be defined. EA is able to help with this. By using the [JUnit](#) or [NUnit](#) transformations and [code generation](#) you can create test method stubs for all of the public methods in each of your classes.

The following is a NUnit example in C# that will be followed through the rest of this topic, although it could also be any other .Net language or Java and JUnit.

```
[TestFixture]
public class CalculatorTest
{
    [Test]
    public void testAdd(){
        Assert.AreEqual(1+1,2);
    }
}
```

```

    }

    [Test]
    public void testDivide(){
        Assert.AreEqual(2/2,1);
    }

    [Test]
    public void testMultiply(){
        Assert.AreEqual(1*1,1);
    }

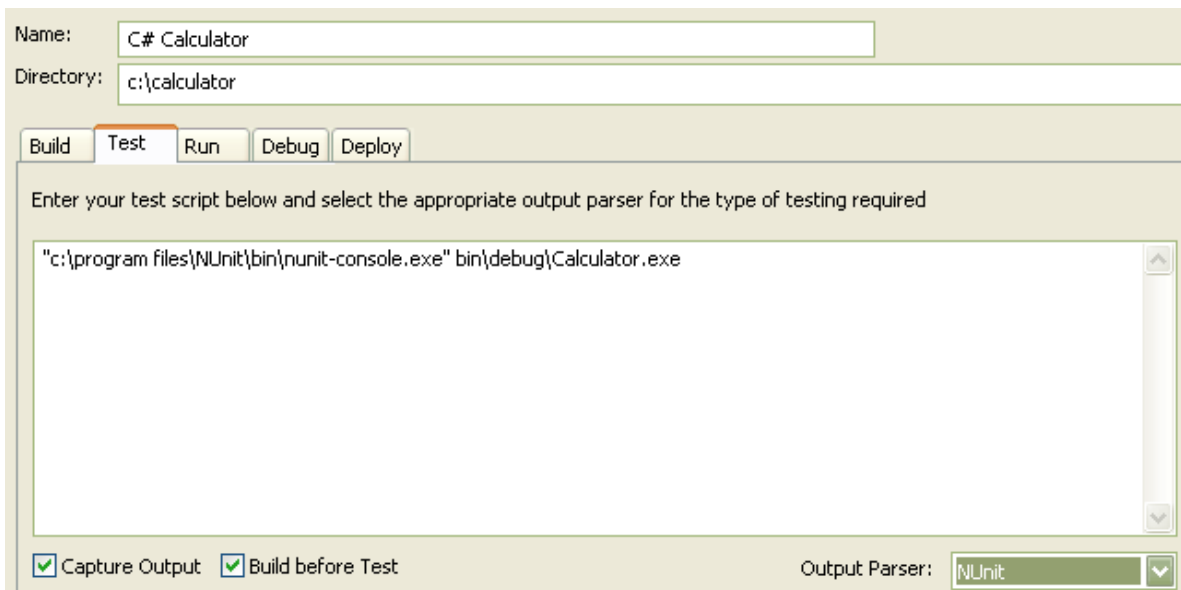
    [Test]
    public void testSubtract(){
        Assert.AreEqual(1-1,1);
    }
}

```

This code can be reverse engineered into EA so that EA can record all test results against this class.

Once the unit tests are set up, you can then set up the Build and Test scripts needed to run the tests. These scripts need to be set up against a package.

The sample above can be called by setting up the [Package Build Scripts](#) dialog as follows.



It is important if you want EA to handle unit testing, that you check Capture Output and select the appropriate Output Parser for the testing. Without doing this you won't see the program output and be able to open the source at the appropriate location.

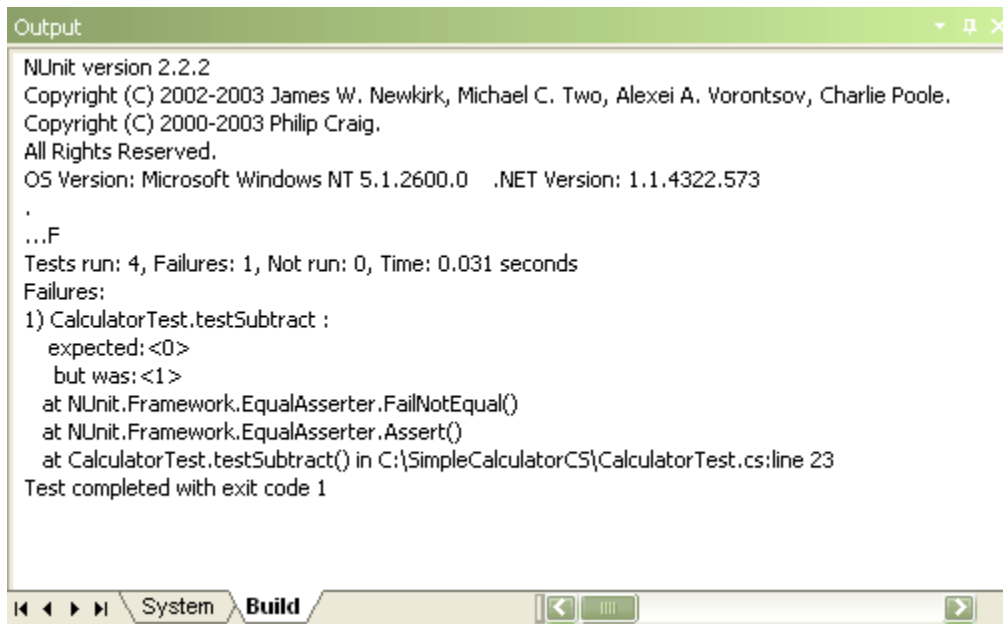
See Also

- [JUnit Transformation](#)
- [NUnit Transformation](#)
- [Testing Scripts](#)

11.3.2 Running Unit Tests

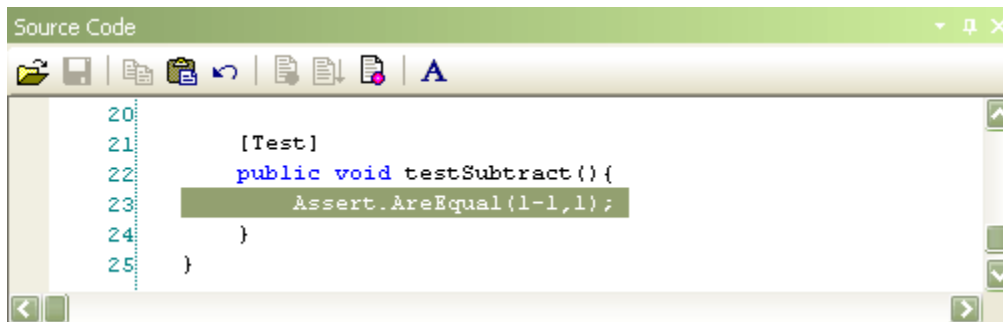
The test script that we setup previously can now be run by selecting **Build and Run | Build** from the **Project** menu.

The following output is given.



```
Output
-----
NUnit version 2.2.2
Copyright (C) 2002-2003 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov, Charlie Poole.
Copyright (C) 2000-2003 Philip Craig.
All Rights Reserved.
OS Version: Microsoft Windows NT 5.1.2600.0 .NET Version: 1.1.4322.573
.
...F
Tests run: 4, Failures: 1, Not run: 0, Time: 0.031 seconds
Failures:
1) CalculatorTest.testSubtract :
   expected: <0>
   but was: <1>
   at NUnit.Framework.EqualAsserter.FailNotEqual()
   at NUnit.Framework.EqualAsserter.Assert()
   at CalculatorTest.testSubtract() in C:\SimpleCalculatorCS\CalculatorTest.cs:line 23
Test completed with exit code 1
```

Notice how NUnit reports that four tests have run, including one failure. It also reports what method failed and the file and line number the failure occurred at. We can now double click on that error and EA opens the editor to that line of code.



```
Source Code
-----
20
21     [Test]
22     public void testSubtract() {
23         Assert.AreEqual(1-1,1);
24     }
25 }
```

This allows us to quickly find and fix the error.

EA will have also recorded the run status of each test as described in [Recording Test Results](#).

11.3.3 Recording Test Results

EA is able to automatically record all results from tests by a testing script in EA. In order to use this feature, you need to do is reverse engineer the test class into the package containing your test script.

Once your model contains your test class, on the next run of the test script EA will add test cases to the class for each test method found. On this and all subsequent test runs all test cases will be update with the current run time and if the passed or failed as shown below.

| CalculatorTest |
|---|
| + testAdd() : void + testDivide() : void + testMultiply() : void + testSubtract() : void |
| test scripts Unit : (Pass) testAdd Unit : (Pass) testDivide Unit : (Pass) testMultiply Unit : (Fail) testSubtract |

The error description for each failed test will be added along with the current date and time to any existing results for that test case. Over time this will give a log of all test runs where each test case has failed. This can then be included in generated documentation and could look like the following.

```
Failed at 05-Jul-2006 1:02:08 PM  
expected: <0>  
but was: <1>
```

```
Failed at 28-Jun-2006 8:45:36 AM  
expected: <0>  
but was: <2>
```

See Also

- [Reverse Engineering](#)
- [Setting Up Unit Testing](#)
- [Running Unit Tests](#)

Part

12

12 XML Technologies

Enterprise Architect enables the rapid modeling, forward and reverse engineering of two key W3C XML technologies : [XML Schema](#) (XSD) and [Web Service Definition Language](#) (WSDL). XSD and WSDL support is critical for the development of a complete Service Oriented Architecture (SOA), and the coupling of UML 2.0 and XML provides the natural mechanism for specifying, constructing and deploying XML based SOA artifacts within an organization.

The following topics explain how to work with these technologies using Enterprise Architect.

- [XML Schema \(XSD\)](#)
- [Web Services \(WSDL\)](#)

12.1 XML Schema (XSD)

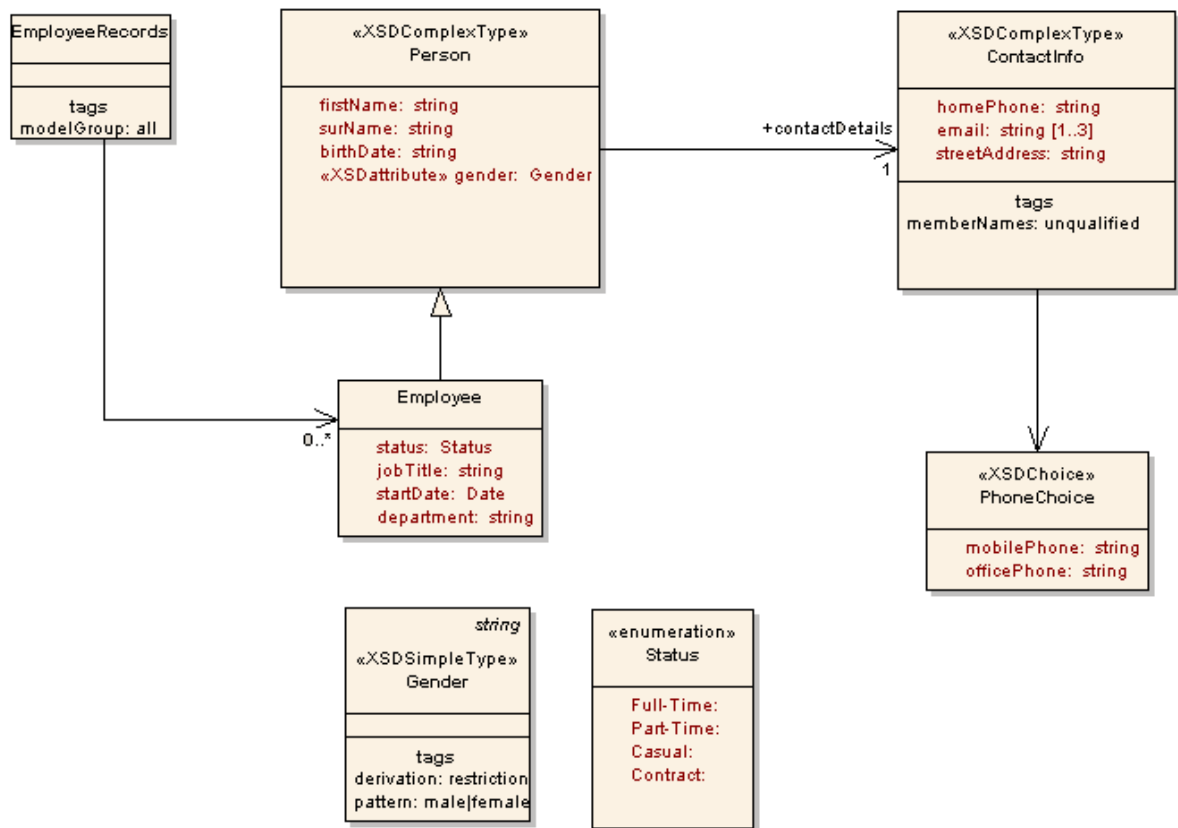
EA supports Forward and Reverse engineering of W3C XML schemas (XSD). The following topics explain how to use EA to model, generate and import XML schemas :

- [Model XSD](#)
- [Import XSD](#)
- [Generate XSD](#)

12.1.1 Model XSD

XML schemas are modeled using UML class diagrams and the [XML Schema toolbox](#). EA's XML Schema toolbox provides in-built support for the [UML profile for XSD](#). This allows an abstract UML class model to be automatically generated as a [W3C XML Schema](#) (XSD) file.

The Class diagram below models simple schema for an example "Employee Details" system, intended to store a company's employee contact information. The classes shown form the "EmployeeDetails" package. The UML attributes of the classes map directly to XML elements or attributes. Note that the classes have no methods, since there is no meaningful correspondence between class methods and XSD constructs.



The following figure shows the schema which is generated for the Employee Details package by default. Notice how each UML class corresponds to a complexType definition in the schema. The class attributes are generated as schema elements contained in a "sequence" model group within the definition. The enumeration class is the exception here- it maps directly to an XSD enumeration, contained within a simpleType definition.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="Gender">
    <xs:restrictionbase="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Person" type="Person"/>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="Person.surName" type="xs:string"/>
      <xs:element name="Person.firstName" type="xs:string"/>
      <xs:element name="Person.birthDate" type="xs:string"/>
      <xs:element name="Person.contactDetails">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ContactInfo"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attributeGroup ref="Person.att"/>
  </xs:complexType>
  <xs:attributeGroup name="Person.att">
    <xs:attribute name="Person.gender" type="Gender"/>
  </xs:attributeGroup>
  <xs:element name="Employee" type="Employee"/>
  <xs:complexType name="Employee">
  
```

```

        <xs:complexContent>
          <xs:extension base="Person">
            <xs:sequence>
              <xs:element name="Employee.status" type="Status"/>
              <xs:element name="Employee.jobTitle" type="xs:string"/>
              <xs:element name="Employee.startDate" type="xs:date"/>
              <xs:element name="Employee.department" type="xs:string"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    <xs:element name="ContactInfo" type="ContactInfo"/>
    <xs:complexType name="ContactInfo">
      <xs:sequence>
        <xs:element name="homePhone" type="xs:string"/>
        <xs:element name="email" type="xs:string" maxOccurs="3"/>
        <xs:element name="streetAddress" type="xs:string"/>
        <xs:choice>
          <xs:element name="mobilePhone" type="xs:string"/>
          <xs:element name="officePhone" type="xs:string"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="Status">
      <xs:restriction base="xs:string">
        <xs:enumeration value="Full-Time"/>
        <xs:enumeration value="Part-Time"/>
        <xs:enumeration value="Casual"/>
        <xs:enumeration value="Contract"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:element name="EmployeeRecords" type="EmployeeRecords"/>
    <xs:complexType name="EmployeeRecords">
      <xs:all>
        <xs:element name="EmployeeRecords.Employee">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="Employee" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:schema>

```

The following topics explain further

12.1.1.1 UML Profile for XSD

The UML Profile for XSD specifies a set of **stereotypes**, **tagged values** and **constraints** which may be applied to the UML model in order to change particular aspects of the resulting schema. For example, we may wish to have certain UML class attributes converted to XSD attributes or, we may need to use a different model group than the default "sequence".

EA provides native support for the UML Profile for XSD via the [XML schema Toolbox](#). Alternatively, the profile can be used via EA's generic profile mechanism by downloading the [UML Profile for XSD](#). See the topic [Using Profiles](#) for details on importing UML profiles into EA. The XSD profile used by EA, is an adaptation of the profile defined by David Carlson in his book "Modeling XML Applications with UML".

The XSD stereotypes provide an explicit mapping from XSD to UML constructs. The tagged values further define aspects of the mapping, such as whether the elements should be qualified. The constraints define any conditions that must be satisfied for the stereotype to apply.

The following table lists the features of the UML Profile for XSD. Tagged value names are shown in bold followed by the allowed values. If there is a default value used by EA's schema generator it is underlined.

<<XSDschema>>

| | |
|---|---|
| UML Construct | Package |
| Description | All classes in a package are defined within one schema. This stereotype can be used to specify schema-wide settings. |
| Tagged Values | |
| anonymousRole: (true false) | Specifies if the role name is included in the element declaration for the UML attribute. |
| anonymousType: (true false) | Specifies whether the class type is anonymous for attributes. |
| attributeFormDefault: (qualified unqualified) | Determines whether attribute instances must be qualified. |
| defaultNamespace: | The default namespace used in this schema. This value is used to specify the default namespace attribute (xmlns=), in the schema element. |
| elementDerivation: (true false) | Determines whether inheritances are generated using XSD extension or copy-down inheritance. |
| elementFormDefault: (qualified unqualified) | Determines whether element instances must be qualified. |
| memberNames: (qualified unqualified) | Determines whether elements generated from Class attributes have their name qualified by the corresponding class name. |
| modelGroup: (all sequence choice) | Specifies the default XSD model group used to generate complexType definitions. |
| schemaLocation: | The URI which identifies the location of the schema. This value is used in the import and include elements. |
| targetNamespace: | The URI which uniquely identifies this schema's namespace. |
| targetNamespacePrefix: | The prefix which abbreviates the targetNamespace. |
| version: | The version of this schema. |
| Constraints | None. |

<<XSDcomplexType>>

| | |
|--|---|
| UML Construct | Class |
| Description | complexType definitions are created for generic UML classes. This stereotypes helps tailor the generation of a complexType definition |
| Tagged Values | |
| memberNames: (qualified unqualified) | Determines whether elements generated from the UML class attributes and associations have their name qualified by the corresponding class name for this complexType definition. |
| mixed: (true false) | Determines whether this element may contain mixed element and character content. Refer to the W3C XML Schema recommendation. |
| modelGroup: (all sequence choice) | Overrides the default XSD model for generating this complexType definition. |
| Constraints | None. |

<<XSDsimpleType>>

| | |
|--|---|
| UML Construct | Class |
| Description | An XSD simpleType is generated for classes with this stereotype. |
| Tagged Values | |
| derivation: (restriction list) | Specifies the derivation of the simpleType. Refer to the W3C XML Schema recommendation. |
| length: | Refer to the W3C XML Schema recommendation. |
| minLength: | Refer to the W3C XML Schema recommendation. |
| maxLength: | Refer to the W3C XML Schema recommendation. |
| minInclusive: | Refer to the W3C XML Schema recommendation. |
| minExclusive: | Refer to the W3C XML Schema recommendation. |
| maxInclusive: | Refer to the W3C XML Schema recommendation. |
| maxExclusive: | Refer to the W3C XML Schema recommendation. |
| totalDigits: | Refer to the W3C XML Schema recommendation. |
| fractionDigits: | Refer to the W3C XML Schema recommendation. |
| whiteSpace: | Refer to the W3C XML Schema recommendation. |
| pattern: | Refer to the W3C XML Schema recommendation. |
| Constraints | This class can only participate in an inheritance relation with another simpleType. It cannot have any attributes or own any associations. They will be ignored if present. |

<<XSDsequence>>

| | |
|----------------------|--|
| UML Construct | Class |
| Description | <p>The schema generator creates a sequence model group as the container for the attributes and associations owned by this class. The model group is in turn added to the model groups of this class' respective owners.</p> <p>Note: Tagged values specified by owners of this class persist through to the child elements of this model group. Thus if memberNames are unqualified for a complexType, so will be the children of this model group when added to that complexType.</p> |
| Tagged Values | None. |
| Constraints | This class must be the destination of unidirectional associations. If it is not, this class and its connectors are ignored, possibly invalidating other model group classes. Inheritance relations are ignored for this class. |

<<XSDchoice>>

| | |
|----------------------|---|
| UML Construct | Class |
| Description | Creates an XSD choice element. Refer to XSDsequence for more details. |
| Tagged Values | None. |
| Constraints | As for XSDsequence. |

<<XSDelement>>

| | |
|---|---|
| UML Construct | Attribute; AssociationEnd |
| Description | By applying this stereotype to a UML class attribute or AssociationEnd, the corresponding UML entity is generated as an element within the parent complexType and not as an XSD attribute. |
| Tagged Values | |
| form: (qualified unqualified) | Overrides the schema's elementFormDefault value. |
| position: | Causes the elements to be ordered within a sequence model group of the containing complexType. Duplicated and invalid position tagged values are ignored and result in undefined ordering of the UML attributes. Missing position values cause the defined positions to be allocated as specified, with the remaining elements filling the missing positions in an undefined order. |
| anonymousRole: (true false) | Specifies if the role name is included in the element declaration for the UML attribute |
| anonymousType: (true false) | Specifies whether the class type is anonymous for attributes. |
| Constraints | None. |

<<XSDatatribute>>

| | |
|---|--|
| UML Construct | Attribute; AssociationEnd |
| Description | By applying this stereotype to a UML class attribute or AssociationEnd, the corresponding UML entity is generated as an XSD attribute within the parent complexType and not as an XSD element. |
| Tagged Values | |
| form: (qualified unqualified) | Overrides the schema's attributeFormDefault value. |
| use: (prohibited <u>optional</u> required) | Refer to the W3C XML Schema recommendation. |
| default: | Refer to the W3C XML Schema recommendation. |
| fixed: | Refer to the W3C XML Schema recommendation. |
| Constraints | The attribute datatype should not refer to a class specification, it will be ignored otherwise. |

<<XSDany>>

| | |
|--|--|
| UML Construct | Class; Attribute |
| Description | If applied to a UML attribute, an XSD anyAttribute element is generated. If applied to a UML class, an XSD any element is generated. |
| Tagged Values | |
| namespace: | Refer to the W3C XML Schema recommendation. |
| processContents: (skip lax strict) | Refer to the W3C XML Schema recommendation. |
| Constraints | None. |

<<XSDrestriction>>

| | |
|----------------------|---|
| UML Construct | Generalization |
| Description | Overrides the default use of XSD extension for inheritance and generates the child as a complexType with a restriction element instead. |
| Tagged Values | None. |
| Constraints | Applies only to UML class parent-child relations. |

<<XSDgroup>>

| | |
|---|--|
| UML Construct | Class |
| Description | An XSDgroup is generated for classes with this stereotype. |
| Tagged Values | |
| modelGroup: (sequence choice all) | Overrides the default XSD model for generating this group definition. |
| Constraints | A group class can only associate itself to other group classes. A group class can be associated by another group class or a complexType class. The association should be via an association link. A group class cannot be inherited/aggregated. |

<<XSDtopLevelElement>>

| | |
|----------------------|--|
| UML Construct | Class |
| Description | Creates an <xs:element> construct which acts as a container for XSDcomplexType and XSDsimpleType class. |
| Tagged Values | None |
| Constraints | An XSDtopLevelElement class can contain either a XSDsimpleType or a XSDcomplexType as its child class. When such a class is present as its child, all its inheritance will be ignored. This class cannot be inherited. |

<<XSDtopLevelAttribute>>

| | |
|---|--|
| UML Construct | Class |
| Description | Creates an <xs:attributr> construct which acts as a container for XSDsimpleType class |
| Tagged Values | |
| use: (optional required prohibited) | Refer to the W3C XML Schema recommendation. |
| Constraints | An XSDtopLevelAttribute class can contain only a XSDsimpleType class as its child class. When such a class is present as its child, all its inheritance will be ignored. This class can inherit from only one XSDsimpleType class. |

<<XSDunion>>

| | |
|----------------------|--|
| UML Construct | Class |
| Description | Creates an <xs:union> construct which can act as a container for XSDsimpleType class. |
| Tagged Values | None |
| Constraints | An XSDunion class can contain only XSDsimpleType as its child class and can generalize from other XSDsimpleType classes only. All the classes that this class generalizes becomes the members of the attribute "memberTypes". This class cannot have any attributes or associations. |

<<XSDattributeGroup>>

| | |
|----------------------|---|
| UML Construct | Class |
| Description | Creates an <xsattributeGroup> construct which can act as a container for a set of elements fo stereoType XSDattribute. |
| Tagged Values | None |
| Constraints | An XSDattributeGroup class can contain only elements of stereoType XSDattribute and can be associated to other XSDattributeGroup classes only.Only XSDcomplexType classes can associate itself with this class. This class cannot be inherited. |

12.1.1.2 XSD Datatypes Package

When modeling XSD constructs, it is often useful to have the XSD primitive types represented as UML elements. In this way, user-defined types for example, can reference the datatype elements as part of inheritance or association relationships.

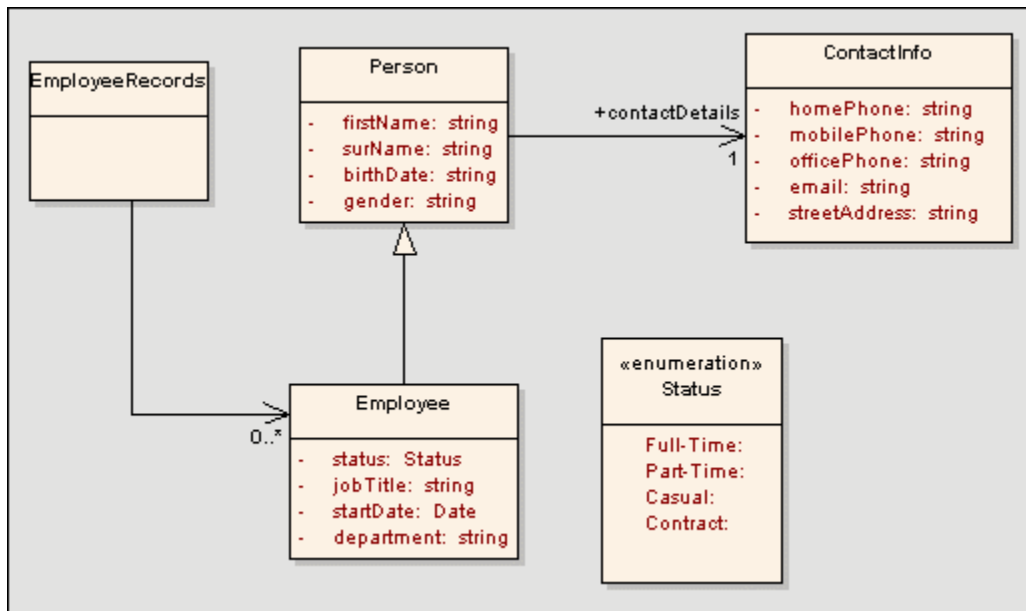
SparxSystems provides the set of primitive XSD data types as a UML package in the form of an XMI file. Each of the XSD primitive types are represented by an UML class in a package named XSDDatatypes. To import the XSDDatatypes package into your model, use the following steps:

1. Download the XSDDatatypes package using the following link : [XSDDatatypes Package](#). The file XSDDataTypes.xml, is an XMI file
2. Use EA's [XML import](#) facility which is available from the menu item : *Project | Import/Export | Import Package from XMI*
3. Once the XMI import is complete, you will have the UML package named XSDDatatypes in your model, from which you can drag and drop the relevant types as required

12.1.1.3 Abstract XSD models

XML schemas may be modeled using simple, abstract class models. This can be useful for allowing an architect to start work a higher level of abstraction, without concern for the implementation details of a schema. Such an abstract model may be refined further using EA's XML Schema Toolbox or it may be generated directly, by EA's [schema generator](#). In this case, a set of [default mappings](#) is assumed by the schema generator to convert the abstract model to an XSD file.

Following is a simplified version of our Employee Details example model, which does not use XSD-specific stereotypes or tagged values.



The following schema fragment would be generated by EA, given the above model.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Person" type="Person"/>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="Person.firstName" type="xs:string"/>
      <xs:element name="Person.surName" type="xs:string"/>
      <xs:element name="Person.birthDate" type="xs:string"/>
      <xs:element name="Person.gender" type="xs:string"/>
      <xs:element name="Person.contactDetails">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ContactInfo"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Employee" type="Employee"/>
  <xs:complexType name="Employee">
    <xs:complexContent>
      <xs:extension base="Person">
        <xs:sequence>
          <xs:element name="Employee.status" type="Status"/>
          <xs:element name="Employee.jobTitle" type="xs:string"/>
          <xs:element name="Employee.startDate" type="xs:date"/>
          <xs:element name="Employee.department" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ContactInfo" type="ContactInfo"/>
  <xs:complexType name="ContactInfo">
    <xs:sequence>
      <xs:element name="ContactInfo.homePhone" type="xs:string"/>
      <xs:element name="ContactInfo.mobilePhone" type="xs:string"/>
      <xs:element name="ContactInfo.officePhone" type="xs:string"/>
      <xs:element name="ContactInfo.email" type="xs:string"/>
      <xs:element name="ContactInfo.streetAddress" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

```

```

<xs:simpleType name="Status">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Full-Time"/>
    <xs:enumeration value="Part-Time"/>
    <xs:enumeration value="Casual"/>
    <xs:enumeration value="Contract"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="EmployeeRecords" type="EmployeeRecords" />
<xs:complexType name="EmployeeRecords">
  <xs:sequence>
    <xs:element name="EmployeeRecords.Employee">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Employee" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

12.1.1.3.1 Default UML to XSD mappings

The following table describes the default mapping of UML to XSD constructs. This set of mappings is useful when defining simple schemas from abstract class models. The defaults are also assumed by the the schema generator when generating unsteretyped elements in an abstract model. The XML schema toolbox (and UML Profile for XSD) override these default mappings through the use of stereotypes and tagged values.

| UML Construct | Default XSD Production Rules |
|---------------|--|
| Package | <p>A schema element is generated for the target package. If the target package includes classes from another package, which has the tagged values "targetNamespace" and "targetNamespacePrefix" set, these are included as attributes of the schema element.</p> <p>In addition, an import or include element is created for each referenced package. (An include element is used if the external package shares the same targetNamespace tagged value as the target package. An import element is used where the targetNamespaces differ).</p> |
| Class | <p>A root-level element declaration and complexType definition are generated. The element name and type are the same as the class name. An XSD sequence model group is generated to contain UML attributes generated as elements.</p> |
| Attribute | <p>An element is declared for each class attribute. The element name is set to that of the UML attribute name. This is prepended with the class name to make the element unique. The minOccurs and maxOccurs attributes are set to reflect the attribute cardinality. (Note: If left unspecified, minOccurs and maxOccurs default to 1.) If the attribute refers to another class, the element declaration is followed a complexType definition, which contains a reference to the appropriate complexType.</p> |

| | |
|------------------------------|--|
| Association | An element is declared for each association owned by a class. The element name is set to that of the association role. The minOccurs and maxOccurs reflect the cardinality of the association. Note: if the direction of the association is unspecified, the owner is assumed to be the source. |
| Generalization (Inheritance) | For single inheritances, an extension element is generated with the base attribute set to the base class name. The UML attributes of the child class are then appended to an all model group within the extension element. |
| <<enumeration>> (stereotype) | A simpleType element is declared for the enumeration class with the name attribute set to the class name. A restriction element is generated with base set to string . Each of the class attributes are appended to the restriction element as XSD enumeration elements with value set to the UML attribute name. Any type specification for the UML attributes will be ignored by the schema generator. |

12.1.2 Generate XSD

The Generate XML Schema feature Forward Engineers a UML class model to a W3C XML Schema (XSD) file. An XML schema corresponds to a UML package in EA, therefore XML schema generation is a package-level operation. Use the following steps to generate an XML schema from a package:

1. In the Project Browser, right click on the package to be converted to XSD to open the context menu.
2. Select *Generate XML Schema* from the *Code Engineering* submenu.
3. Set the desired output file using the *Filename* field.
4. Set the desired XML encoding using the *Encoding* field.
5. Press *Generate* to generate the schema.
6. The progress of the schema generator will be shown in the *Progress* edit box.

*Tip: The *Generate XML Schema* dialog may also be accessed from the active diagram by selecting *Generate XML Schema* from the *Project* menu.*

12.1.3 Import XSD

The XML Schema Import facility is used to Reverse Engineer a W3C XML Schema (XSD) file as a UML class model. XSD files are imported into EA as a UML package. Use the following steps to import an XSD file:

1. In the Project View, right click on a package to open the context menu. This package will contain the imported XSD package.
2. Select *Import XML Schema* from the *Code Engineering* submenu.
3. Select the input file using the *Filename* field.
4. The *Target Package* field will be automatically set to the name of the selected input file by default. If required, this name can be changed.

5. Check the *Create Logical Diagram for Package* checkbox if the imported elements are to be displayed on the diagram (selected by default).
6. Press *Import* to Import the schema.
7. The progress of the schema import will be shown in the Progress status bar.

Tip: The Import XML Schema dialog may also be accessed for the active diagram by selecting Import XML Schema from the Project menu.

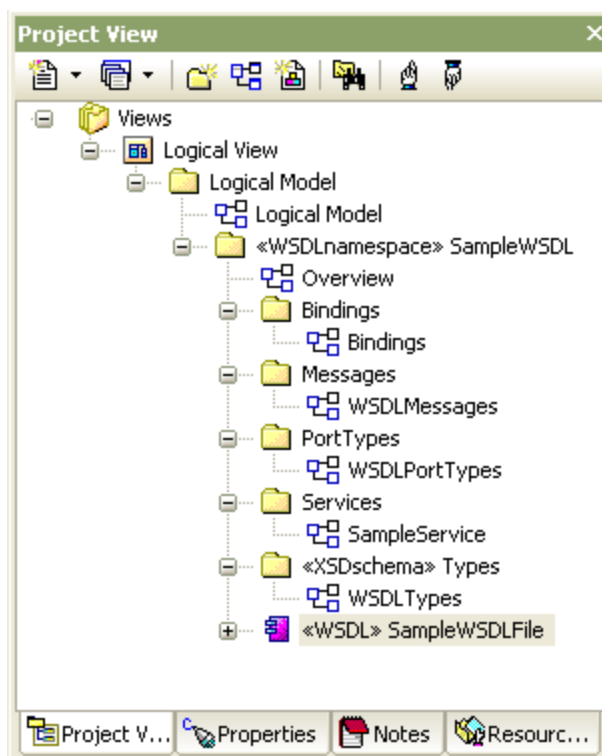
12.2 Web Services (WSDL)

EA supports Forward and Reverse Engineering of the W3C Web Service Definition Language (WSDL). The following topics explain how to use EA to model, generate and import WSDL files.

- [Model WSDL](#)
- [Import WSDL](#)
- [Generate WSDL](#)

12.2.1 Model WSDL

EA's WSDL toolbox can be used to conveniently model WSDL documents. WSDL documents are represented as components marked with the stereotype "WSDL". WSDL documents are contained in a package hierarchy representing the target WSDL namespace and its constituent XSD Types, Messages, PortTypes, Bindings and Services. The top-level package is stereotyped as a WSDLnamespace. The figure below shows a skeletal WSDL namespace package structure:

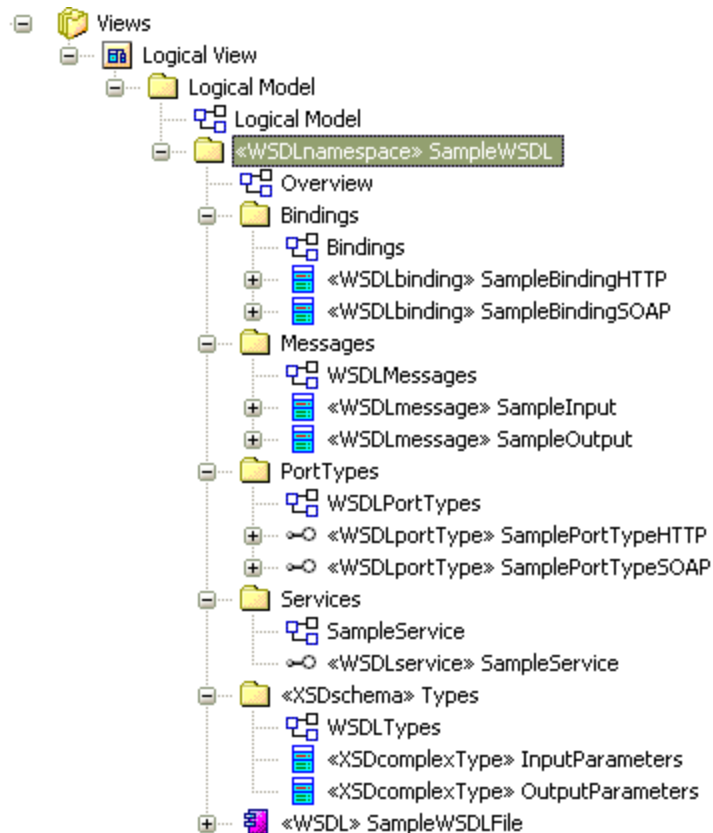


A WSDLnamespace package may contain one or more WSDL components. Each WSDL component may be automatically generated to a WSDL file using EA's built in [WSDL generator](#). The following sections describe the various WSDL elements supported by EA.

12.2.1.1 WSDL Namespace

The WSDL namespace in EA represents the top-level container for the WSDL elements, including WSDL documents. Conceptually it maps to the targetNamespace in a WSDL definition element. A given WSDL namespace may reuse its schema Types, Messages, Port Types, Bindings and Service across multiple physical WSDL documents

The figure below shows an example WSDL namespace, including a single WSDL document:



Use the following steps to create a new WSDL namespace in your model.

1. Select the WSDL item from the UML Toolbox
2. Drag the *Namespace* item from the toolbox onto a diagram. This will invoke the *WSDL Namespace Properties* dialog:

WSDL Package Name :

Target Namespace :

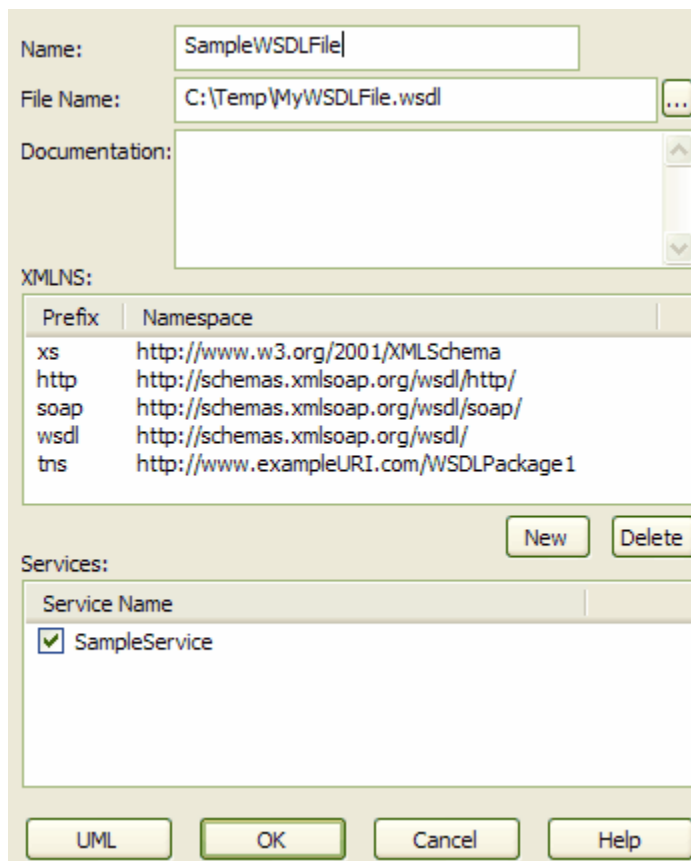
3. Enter a name and target namespace. These values may be changed later.
4. Click the OK button. This will create a package stereotyped as "WSDLnamespace". It will contain the following sub-packages and an overview diagram to navigate between the sub-packages:
 - Types : Contains the XSD types used by the WSDL Message elements. This package is modeled as an [XML Schema](#)
 - Messages : Contains the WSDL Messages. Messages are modeled as UML classes marked with the stereotype WSDLmessage.
 - PortTypes : Contains the WSDL Port Types. PortTypes are modeled as UML interfaces marked with the stereotype WSDLportType.
 - Bindings : Contains the WSDL Bindings. Bindings are modeled as UML classes which realize the PortTypes.
 - Services : Contains the WSDI Services. Services are modeled as UML interfaces with associations to each exposed Binding.
5. Use the "Overview" diagram to navigate between the subpackages, by double-clicking the relevant packages. You may edit the sample WSDL elements created in the previous step, or drag new items from the WSDL tool box onto the relevant diagrams.
6. You may edit the WSDL-specific properties of the namespace again by double-clicking the package in the Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for a package.

12.2.1.2 WSDL Document

WSDL documents are represented in EA by UML components stereotyped as "WSDLdocument". These components are modeled as direct child elements of the top-level WSDL namespace package. Multiple WSDL documents may be created for a single namespace, thus allowing the services for that namespace to be reused and exposed as required across multiple WSDLs.

You may use the following steps to define new WSDLdocument components for your namespace:

1. Open the Overview diagram defined for your WSDL namespace package. This will open the WSDL item in the UML toolbox
2. Drag the *Document* item from the toolbox onto the Overview diagram. This will invoke the *WSDL Document Properties* dialog:



3. Enter the *Name* and *File Name* for the document
4. You may optionally use the *XMLNS* field to specify the XML namespaces used by the document
5. Select one or more services that should be exposed by this document. The list of available services is populated from the Services package
6. Click the OK button
7. You may edit the WSDL-specific properties of the document later by double-clicking the component in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for a package

12.2.1.3 WSDL Service

WSDL services are represented in EA by UML interfaces, stereotyped as "WSDLservice". Services should be defined under the Services packages in the WSDL namespace structure.

You may use the following steps to define new WSDLservice elements for your namespace:

1. Open the Overview diagram defined for your WSDL namespace package. This will open the WSDL item in the UML toolbox
2. Drag the *WSDL Service* item from the toolbox onto a diagram. This will invoke the *WSDL Service Properties* dialog:

Name:

Documentation:

Ports

| Port Name | Binding | Location |
|----------------|-------------------|------------------------|
| SamplePortHTTP | SampleBindingHTTP | http://www.exampleL... |
| SamplePortSOAP | SampleBindingSOAP | http://www.exampleL... |

3. Enter the *Name* for the document
4. Click the *New* button to add Service Ports. This will invoke the *WSDL Port* dialog shown below :

Port Name:

Binding:

Location:

Documentation:

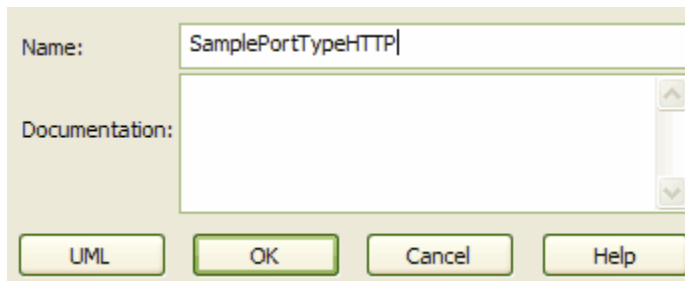
5. Enter the *Port Name*, *Location* and select a *Binding*. The list of Bindings is taken from those defined in the Bindings package
6. Click the *OK* button to close the *WSDL Port* dialog. For each Port defined in this way, EA will create an Association relationship between the Service and corresponding Binding element
7. Click the OK button to close the *WSDL Service Properties* dialog
8. You may edit the WSDL-specific properties of the service later by double-clicking the Service interface in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for an interface

12.2.1.4 WSDL Port Type

WSDL portTypes are represented in EA by UML interfaces stereotyped as "WSDLportType". PortTypes should be defined under the PortTypes packages in the WSDL namespace structure.

You may use the following steps to define new WSDLportType elements for your namespace:

1. Open the PortTypes diagram defined for your WSDL namespace package
2. From the WSDL toolbox, drag the *Port Type* item onto the diagram. This will invoke the *WSDL PortType Properties* dialog:



The screenshot shows a dialog box titled "WSDL PortType Properties". It has a "Name:" label followed by a text input field containing "SamplePortTypeHTTP". Below that is a "Documentation:" label followed by a large empty text area with a vertical scrollbar. At the bottom, there are four buttons: "UML", "OK", "Cancel", and "Help".

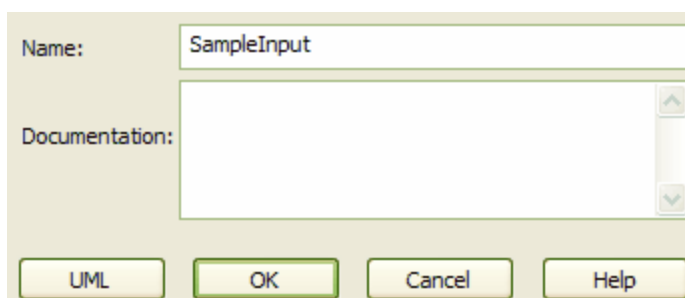
3. Enter the *Name* for the portType
4. Click the OK button to close the *WSDL PortType Properties* dialog
5. You can define operations for the portType by dragging the [Port Type Operation](#) item from the WSDL toolbox onto the portType interface
6. You may edit the WSDL-specific properties of the portType later by double-clicking the interface in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for an interface

12.2.1.5 WSDL Message

WSDL messages are represented in EA by UML classes stereotyped as "WSDLmessage". Messages should be defined under the Messages package in the WSDL namespace structure.

You may use the following steps to define new WSDLmessage elements for your namespace:

1. Open the Messages diagram defined for your WSDL namespace package
2. From the WSDL toolbox, drag the *Message* item onto the diagram. This will invoke the *WSDL Message Properties* dialog:



The screenshot shows a dialog box titled "WSDL Message Properties". It has a "Name:" label followed by a text input field containing "SampleInput". Below that is a "Documentation:" label followed by a large empty text area with a vertical scrollbar. At the bottom, there are four buttons: "UML", "OK", "Cancel", and "Help".

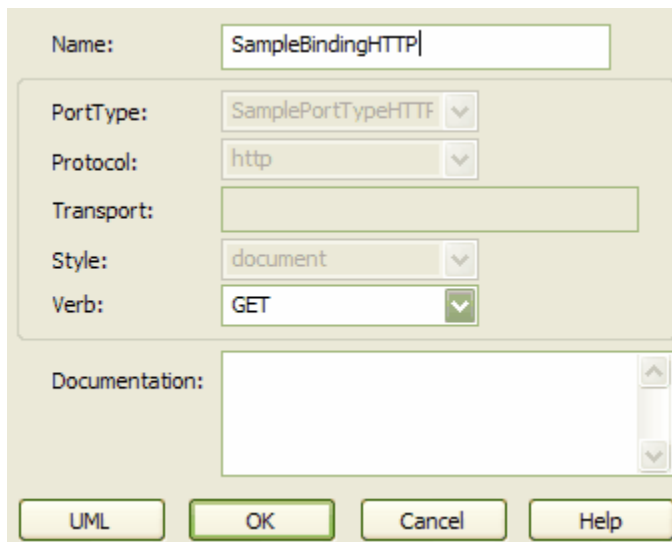
3. Enter the *Name* for the message
4. Click the OK button to close the *WSDL Message Properties* dialog
5. You can define parts for the message by dragging the *Message Part* item from the WSDL toolbox onto the message class
6. You may edit the WSDL-specific properties of the message later by double-clicking the message class in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for a class

12.2.1.6 WSDL Binding

WSDL bindings are represented in EA by UML classes stereotyped as "WSDLbinding". Bindings should be defined under the Bindings package in the WSDL namespace structure. Each WSDLbinding classes implements the operations specified by a particular WSDLportType interface. Therefore, WSDLportTypes should be defined before creating WSDLbindings.

You may use the following steps to define new WSDLbinding elements for your namespace:

1. Open the Bindings diagram defined for your WSDL namespace package
2. From the WSDL toolbox, drag the *Binding* item onto the diagram. This will invoke the *WSDL Binding Properties* dialog:



3. Enter a *Name* for the Binding
4. Select the *PortType* for the Binding. The list of PortTypes is taken from those defined in the PortTypes package
5. Select the *Protocol* for the Binding - either http or soap
6. For soap Bindings, enter the *Transport* URL and select the *Style*. For http Bindings, select the *Verb*
7. To specify the Binding operations, double click on an *Operation*. This will invoke the *WSDL Binding Operations* dialog:

Name: GetSampleSOAP

Documentation:

Operation Type: Request-Response

Input

Name: Request

Message: SampleInput

Documentation:

Output

Name: Response

Message: SampleOutput

Documentation:

Faults

New Delete

| Fault Name | Type |
|------------|------|
| | |

UML OK Cancel Help

8. Enter the Binding Operation details for each operation in the list. Click the Parameters button to invoke the *WSDL Binding Operation Details* dialog and enter the input, output and fault details.
9. Click the OK button to close the *WSDL Binding Operations* dialog
10. Click the OK button on the *WSDL Binding Properties* dialog to close the dialog and create the binding. A realization connector will be created between the binding and the corresponding portType interface
11. You may edit the WSDL-specific properties of the binding later by double-clicking the binding class in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for a class

12.2.1.7 WSDL Port Type Operation

WSDL portType operations are represented in EA by operations defined as part of a WSDLportType interface (see the [WSDL Port Type](#) topic).

You may use the following steps to add portType operations to your WSDLportType interfaces:

1. Open the PortTypes diagram defined for your WSDL namespace package
2. From the WSDL toolbox, drag the *PortType Operation* item onto a WSDLPortType stereotyped interface. This will invoke the *WSDL PortType Operation* dialog:

The dialog box is titled "WSDL PortType Operation" and is used to configure the properties of a WSDL operation. It is organized into several sections:

- Name:** A text field containing "GetSampleSOAP".
- Documentation:** A text area for providing documentation for the operation.
- Operation Type:** A dropdown menu set to "Request-Response".
- Input:** A section for defining the input message, including:
 - Name:** "Request"
 - Message:** A dropdown menu set to "SampleInput".
 - Documentation:** A text area for input message documentation.
- Output:** A section for defining the output message, including:
 - Name:** "Response"
 - Message:** A dropdown menu set to "SampleOutput".
 - Documentation:** A text area for output message documentation.
- Faults:** A table for defining fault types, with "New" and "Delete" buttons above it. The table has two columns: "Fault Name" and "Type".

At the bottom of the dialog are four buttons: "UML", "OK", "Cancel", and "Help".

3. Enter the *Name* for the operation
4. Select the *Operation Type*
5. Enter the Input, Output and Fault details for the operation. The Message drop-list is taken from the WSDLmessage elements defined under the Messages package
6. Click the *OK* button to close the *WSDL PortType Operation* dialog and create the operation
7. You may edit the WSDL-specific properties of the portType operation later by double-clicking the operation in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for an operation

12.2.1.8 WSDL Message Part

WSDL message parts are represented in EA by UML attributes defined as part of a WSDLmessage class (see the [WSDL Message](#) topic).

You may use the following steps to add message parts to your WSDLmessage classes:

1. Open the PortTypes diagram defined for your WSDL namespace package
2. From the WSDL toolbox, drag the *Message Part* item onto a WSDLmessage stereotyped class. This will invoke the *WSDL Message Part* dialog:

3. Enter a Name and Type for the message part. The type should be selected from the drop-list of primitive XSD types or selected from the types defined under the Types package.
4. You may edit the WSDL-specific properties of the message part later by double-clicking the attribute in the diagram or Project View. Alternatively, click the *UML Properties* button to invoke the standard properties dialog for an attribute

12.2.2 Generate WSDL

The Generate WSDL feature Forward Engineers a UML model to a Web Service Definition Language (WSDL) file. The Generate WSDL feature acts on a package stereotyped with WSDLnamespace. It is used to Generate any or all of the WSDL stereotyped components owned by the target WSDLnamespace structure. Use the following steps to generate one or more WSDL files from a WSDLnamespace:

1. In the Project Browser, right click on the target WSDLnamespace package to open the context menu.
2. Select *Generate WSDL* from the *Code Engineering* submenu.
3. For each WSDL component, set the desired output file using the *Target File* column.
4. Set the desired XML encoding using the *Encoding* field.
5. Press *Generate* to generate the WSDL file(s).
6. The progress of the WSDL generator will be shown in the *Progress* edit box.

Tip: The *Generate WSDL* dialog may also be accessed from the active diagram by selecting *Generate WSDL* from the *Project* menu.

12.2.3 Import WSDL

The WSDL Import facility is used to Reverse Engineer WSDL files as UML class models. Use the following steps to import a WSDL file:

1. In the Project View, right click on a package to open the context menu. This package will contain the imported WSDL package.
2. Select *Import WSDL* from the *Code Engineering* submenu.
3. Select the input file using the *Filename* field.

4. The *Target Package* field will be automatically set to the name of the selected input file by default. If required, this name can be changed.
5. Press *Import* to Import the schema.
6. The progress of the WSDL import will be shown in the Progress status bar.

Part

13

13 Data Modeling

Database Modeling

Database modeling and database design are not explicitly covered by the UML Specification, but they may be achieved in Enterprise Architect using the **UML Data Modeling Profile**. This profile provides easy to use and easy to understand extensions to the UML standard, mapping the database concepts of tables and relationships onto the UML concepts of classes and associations. These extensions also allow you to model database keys, triggers, constraints, RI and other relational database features.

Supported Databases

EA supports import of database schema from these databases...

- DB2
- InterBase
- MS Access
- MySQL
- Oracle 9i and 10g
- PostgreSQL
- MS SQL Server
- Sybase Adaptive Server Anywhere
- Sybase Adaptive Server Enterprise
- Firebird

Typical Tasks

Typical tasks you might want to perform when modeling or designing databases include:

- [Creating a Data Model Diagram](#)
- [Creating a Table](#)
- [Setting Properties of a Table](#)
- [Creating Columns](#)
- [Creating Primary Keys](#)
- [Creating Foreign Keys](#)
- [Creating Stored Procedures](#)
- [Creating Views](#)
- [Creating Indexes and Triggers](#)
- [Generating DDL for a Table](#)
- [Generating DDL for a Package](#)
- [Converting Datatype for a Table](#)
- [Converting Datatype for a Package](#)
- [Customizing Datatypes for a DBMS](#)
- [Importing a Database Schema from an ODBC Data Source](#)

Note: The **UML Data Modeling Profile** is not currently a ratified standard; however it has wide industry support and is a useful method for bridging the gap between the UML and conventional relational database modeling.

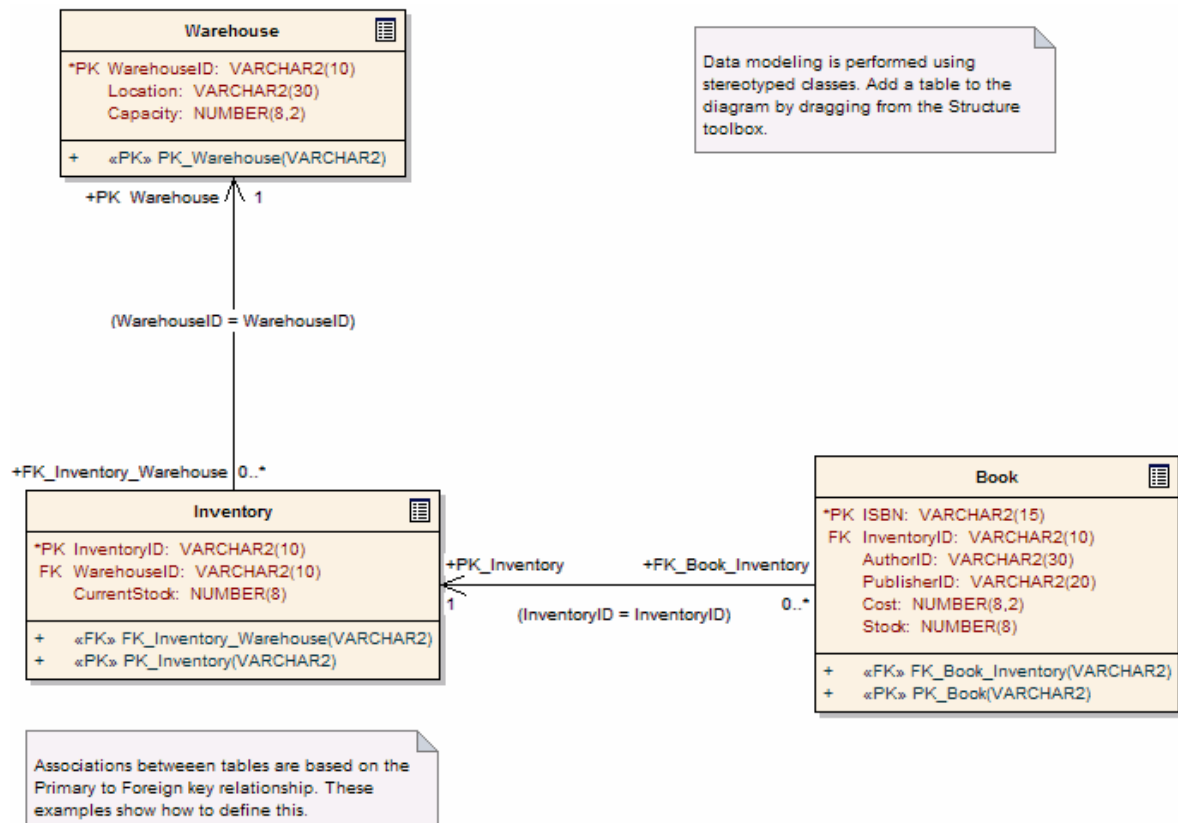
Note: Firebird 1.5 database tables can be modelled and generated as InterBase tables. Firebird tables can be imported but are treated as InterBase tables.

13.1 A Data Model Diagram

The diagram below is an example of a Data Model diagram. It shows three tables which are linked on primary to foreign key pairs with associated multiplicity.

Note also the use of stereotyped operations for Primary (PK) and Foreign (FK) keys. Operations could also be added for:

- Validation (check)
- Triggers
- Constraints
- Indexes



A Data Model diagram is represented in Enterprise Architect as a Class diagram, and is created [in exactly the same way](#) as other diagrams.

13.2 Create a Table

What is a Table?

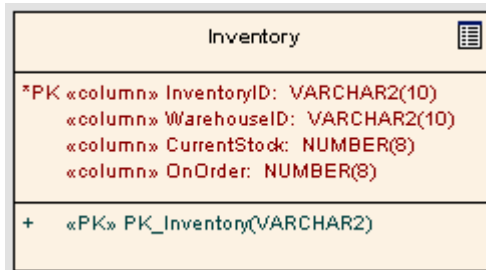
The basic modeling structure of a relational database is the Table. A table represents a set of records, or rows, with the same structure.

The *UML Data Modeling Profile* represents a table as a stereotyped class - that is, a class element with a stereotype of "table" applied to it. A pictorial table icon is shown in the upper right corner of the image when it is shown on a data model diagram.

Create a Table

To create a table, follow the steps below:

1. Select a diagram.
2. Open the *Structure* group on the UML Toolbox.
3. Left click on the *Table* element in the list of elements and then left click on the diagram.
4. Give the table a name and [set any other properties](#) as required.



13.3 Set Table Properties

Once you have created your table, you may set its properties. Most table properties can be set from the *Properties* dialog, as described below. Some properties though will need to be entered as tagged values, and these are described elsewhere, i.e. setting the value of the [Table Owner](#), and for MySQL databases setting the [Table Options](#).

Set the Database type

The most important property to set for a table (after its name) is the *Database* type. This defines the list of datatypes that will be available for defining columns, and also declares which dialect of DDL will be generated. Enterprise Architect supports the following databases: DB2, InterBase, MS Access, MySQL, Oracle 9i and 10g, PostgreSQL, SQL Server 2000 and 2005, SQLServer7 and Sybase.

General Table Detail Require Constraints Link Scenario Files

Name:

Stereotype: ... Abstract

Author: Status:

Scope: Complexity:

Alias:

Persistence:

Database:

Keywords:

Phase: Version:

Note:

To set the Database type, follow the steps below:

1. Double-click on the table element in a diagram to open the *Properties* dialog.
2. Select the *General* tab.
3. Select the Database type from the *Database* dropdown list.
4. Press *OK* to save changes.

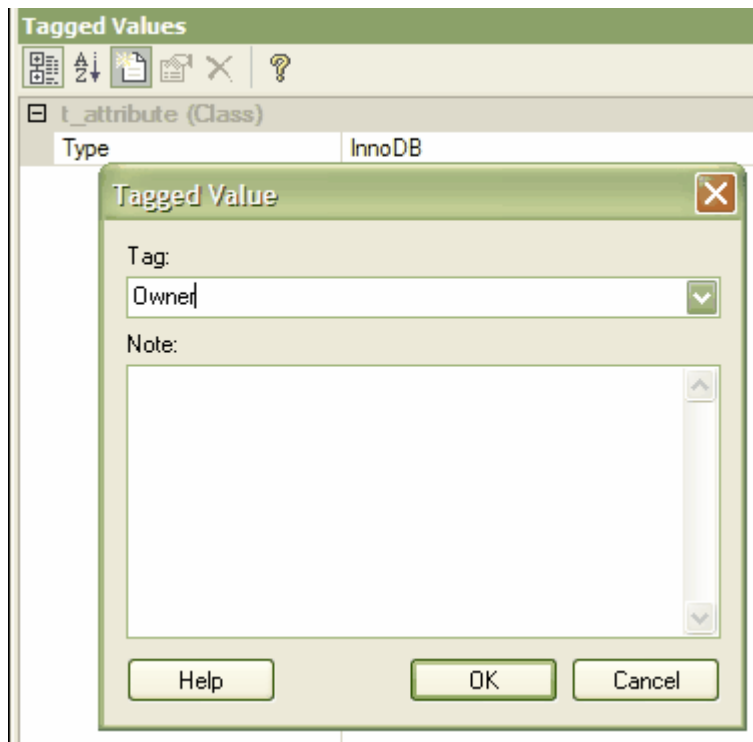
By clicking on the *Table Detail* tab on this dialog, you can then go to the [Columns dialog](#) or the [Operations dialog](#) or you can [Generate DDL](#) for this table.

Table Space:

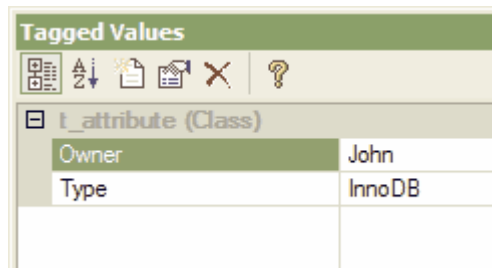
13.3.1 Set Table Owner

To define the owner of a table, follow the steps below:

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the table by clicking on it in a diagram or the *Project View* window. The *Tagged Values* window will now show the tags for the table selected.
3. Press the *New Tag* button.
4. Define a tag named "Owner" in the *Tag* field. Add a comment in the *Note* field, if required.



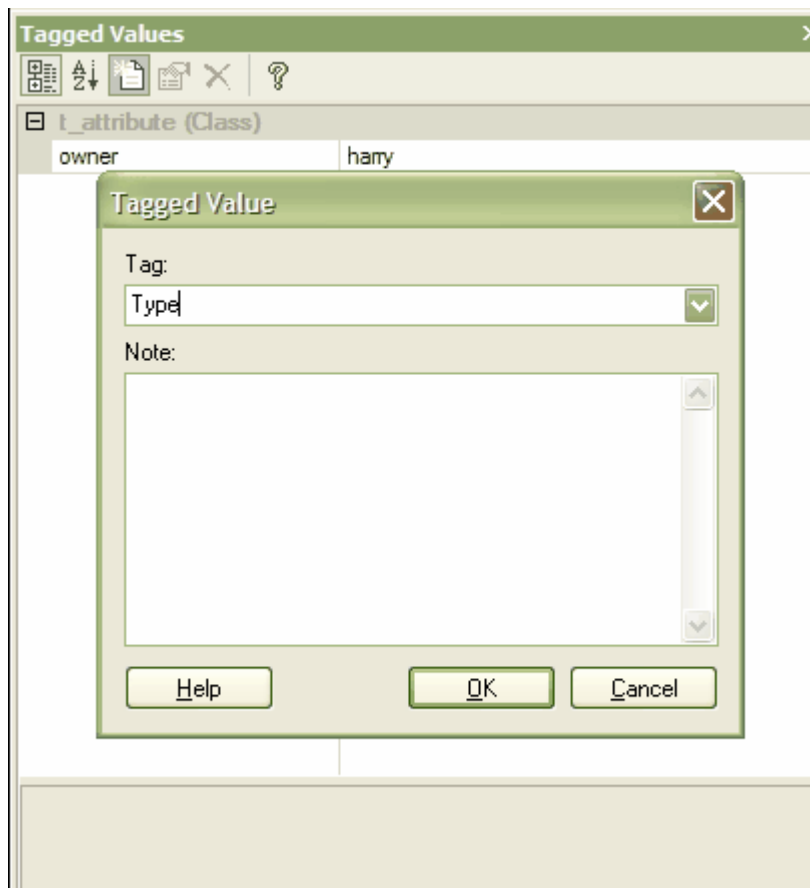
5. Press *OK* to confirm the operation.
6. In the *Tagged Values* Window enter a value for the "Owner" tag. Generated DDL will include the table owner in the SQL script.



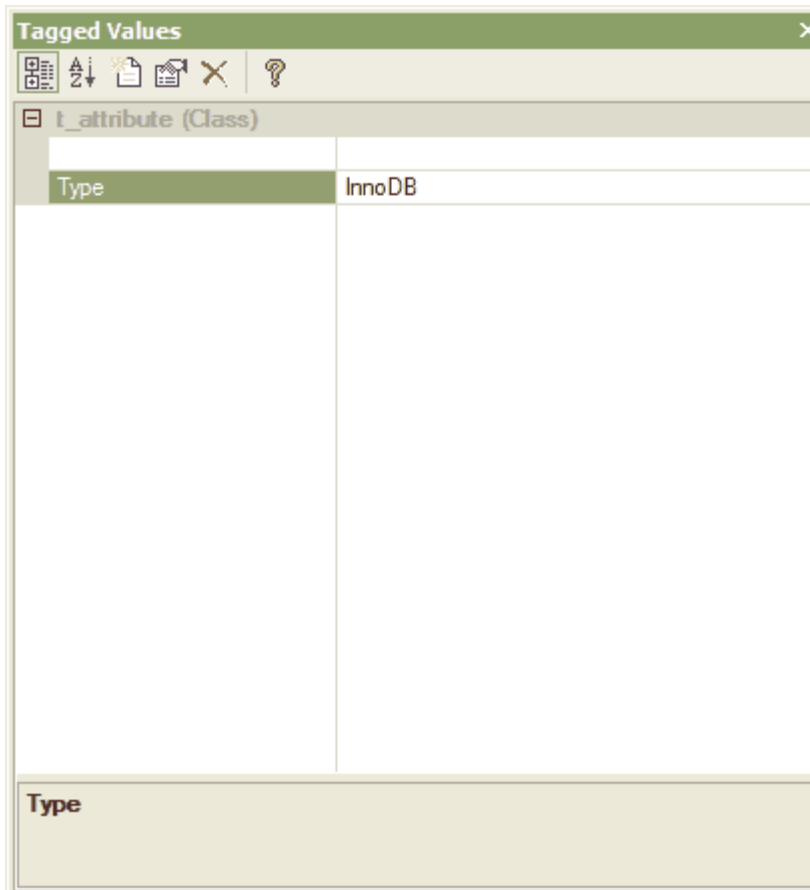
13.3.2 Set MySQL Options

In MySQL, if you wish to make use of foreign keys you will need to declare the table type as "InnoDB". To do this, follow the steps below:

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the table by clicking on it in a diagram or the *Project View* window. The *Tagged Values* window will now have the table selected.
3. Press the *New Tag* button.
4. Define a tag named "**Type**" in the *Tag* field. Add a comment of required in the *Note* field.



5. Press *OK* the button to confirm the operation.
6. In the *Tagged Values* Window, enter the Value "**InnoDB**" as the value for the "Type" tag. Generated DDL will include the table type in the SQL script.



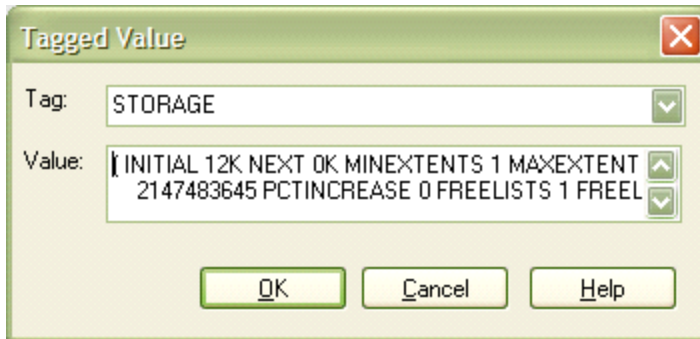
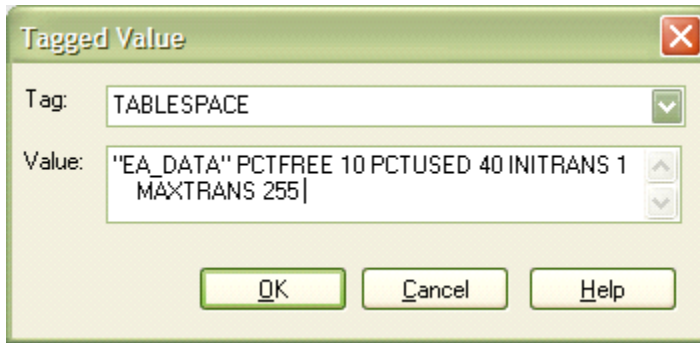
- To allow for later versions of MySQL, additional table options that can be added in the same manner include the following

| Tag | Value (Example) |
|---------------|-------------------|
| ENGINE | InnoDB |
| CHARACTER SET | latin1 |
| CHARSET | latin1 |
| COLLATE | latin1_german2_ci |

13.3.3 Set Oracle Table Options

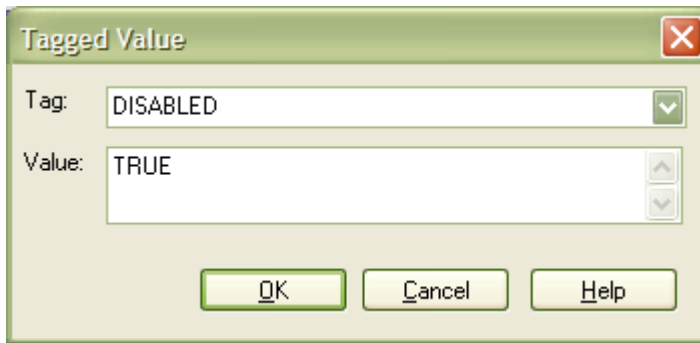
For Oracle 9i and 10g, TABLESPACE and STORAGE table options can be set as follows:

- Ensure the *Tagged Values* window is open by selecting *View | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
- Select the table by clicking on it in a diagram or the *Project View* window. The *Tagged Values* window will now have the table selected.
- Press the *New Tag* button.
- Define a TABLESPACE or STORAGE tag as shown in the examples below:



Set Oracle Constraint Options

Similarly, Oracle constraint options can be set by adding the following tag/value pairs to a highlighted constraint.



| Tag | True Value and DDL Generated | False Value and DDL generated |
|--------------------|------------------------------|-------------------------------|
| DISABLED | DISABLE | ENABLE |
| DEFERRABLE | DEFERRABLE | NOT DEFERRABLE |
| INITIALLY DEFERRED | INITIALLY DEFERRED | INITIALLY IMMEDIATE |
| RELY | RELY | NORELY |
| VALIDATE | VALIDATE | NOVALIDATE |

13.4 Create Columns

What is a Column?

The basic organisational element of a relational database is the Column. Every individual item of data entered into a relational database is represented as a value entered in a column of a row in a table. Columns are

represented in the *UML Data Modeling Profile* as a stereotyped attribute; that is, an attribute with the "column" stereotype.

Create Columns

To create columns, follow the steps below:

1. Right click on the table in a diagram to open the context menu.
2. From the *Element Features* submenu, select *Attributes*.
3. The *Attributes* dialog will appear.
4. Enter a column *Name* and *Data Type* and press *Save*.

Tip: If the pulldown list of datatypes is empty, this means that you have not selected a target database for the table. Close the Attributes dialog and re-open the Table Properties dialog to set a *Database* type before continuing. To prevent this recurring, [set the default database type](#).

5. For each column you can optionally set:
 - *Primary Key*. Tick this box if the column represents the [primary key](#) for this table.
 - *Not Null*. Tick this box if empty values are forbidden for this column.
 - *Unique*. Tick this box if it is forbidden for any two values of this column to be identical.
 - *Initial Value*. Enter a value in here which will used as a default value for this column, if required.
 - *Access*. Select a scope of *Private*, *Protected* or *Public* from the pulldown list (this defaults to *Public*).
 - *Alias*. Enter an alternative name for the field (for display purposes), if any.
 - *Notes*. This field is available for you to enter any other information necessary to document the column.

Tip: Some datatypes, for example the Oracle **NUMBER** type, will require a precision and scale. These fields will appear where required and should be filled in as appropriate. For Oracle, create the **NUMBER** type by setting Precision = 0 and Scale = 0; create **NUMBER(8)** by setting Precision = 8 and Scale = 0; and create **NUMBER(8,2)** by setting Precision = 8 and Scale = 2.

The screenshot shows a dialog box with three tabs: 'General', 'Detail', and 'Constraints'. The 'General' tab is active. It contains the following fields and options:

- Name: StaffID
- Data Type: Integer (dropdown menu)
- Primary Key:
- Not Null:
- Unique:
- Stereotype: column (dropdown menu)
- Initial: 0
- Access: Public (dropdown menu)
- Alias: (empty text field)
- Notes: (empty text area with scroll arrows)
- Buttons: Column Properties..., New, Save, Delete

Below the fields is a table with the following data:

| PK | Name | Type | Not ... | Unique |
|----|---------------|---------|---------|--------|
| PK | StaffID | integer | Yes | No |
| | StaffName | Text | No | No |
| | Staff_Address | Text | No | No |

At the bottom of the dialog are buttons for OK, Cancel, and Help.

Set the Column Order

To change the column order, follow the steps below:

1. Highlight a column name in the list of defined columns.
2. Press the upward pointing finger button to move the column up one position.
3. Press the downward pointing finger button to move the column down one position.

13.5 Primary Key

What is a Primary Key?

Keys are used to access tables, and come in two varieties: **Primary Keys** and **Foreign Keys**. A Primary Key uniquely identifies a record in a table, while a Foreign Key accesses data in some other related table via its Primary Key. This page describes Primary Keys; Foreign Keys are described [elsewhere](#).

Define a Simple Primary Key

If a primary key consists of a single column it is very easy to define.

1. Right click on the table in a diagram to open the context menu and from the *Element Features* submenu, select *Attributes*.
2. In the *Attributes* dialog, select the column which makes up the primary key.

3. Check the *Primary Key* checkbox and press the *Save* button.

It will be seen that a stereotyped operation is automatically created. It is this operation that defines the primary key for the table. To remove a primary key, simply delete this operation.

Define a Complex Primary Key

It will often be the case that a primary key needs to consist of more than one column. For example, a column "LastName" might not be unique within a table, so a primary key is created from the "LastName", "FirstName" and "DateOfBirth" columns. Perform the following steps to create a complex primary key:

1. Follow the steps above to create a Simple Primary Key. It doesn't matter which column you choose to do this.
2. Right click on the table in a diagram to open the context menu and from the *Element Features* submenu, select *Operations*.
3. Select the Primary Key operation (its name will begin "PK_") and then select the *Columns* tab.
4. To add a column to the primary key, press the *New* button, select a column from the *Column Name* list box, and then press the *Save* button.
5. Use the buttons with the up and down pointing hands to change the order of columns in the primary key, if necessary.

See also

- [define a primary key as non-clustered for a SQL Server table](#)

13.5.1 Primary Key Extended Properties

Set SQL Server Clustered /Non Clustered Primary Keys

To define a primary key as non-clustered for a SQL Server table, follow the steps below:

1. Right click on the table in a diagram to open the context menu.
2. From the *Element Features* submenu, select *Operations*.
3. The *Table Operations* dialog will appear.
4. Highlight the Primary Key Operation and select *Extended Properties*.
5. Check the *SQL Server Non Clustered Primary Key* check box.

The screenshot shows a dialog box with three sections:

- Index Properties:**
 - Unique
 - Clustered
 - Ascending
 - Descending
- Foreign Key Properties:**
 - Cascade Delete
 - Cascade Update
- Primary Key Properties:**
 - SQL Server Non Clustered Primary Key

Buttons at the bottom: Cancel, Save, Save & Close.

13.6 Foreign Keys

What is a Foreign Key?

Keys are used to access tables, and come in two varieties: Primary Keys and **Foreign Keys**. A Primary Key uniquely identifies a record in a table, while a Foreign Key accesses data in some other related table via its Primary Key.

Foreign keys are represented in Enterprise Architect UML using **stereotyped operations**. A Foreign Key is a collection of columns (attributes) that together have some operational meaning (they enforce a relationship to a Primary Key in another table). A foreign key is modeled as an operation stereotyped with the "**FK**" **stereotype** - the operation parameters become the columns involved in the key.

Note: It isn't necessary to define a Foreign Key in order to access another table through its Primary Key. Foreign Keys are a feature of some database management systems, providing "extras" such as **referential integrity checking** which prevents the deletion of a record if its primary key value exists in some other table's foreign key. The same thing can be achieved programmatically.

Create a Foreign Key

To create a foreign key, follow the steps below:

1. Locate the required tables in either a diagram or in the Project Browser.
2. Select an associate link in the *Class Toolbar*.
3. Click on the table that will contain the foreign key (source) and draw a connector to the other table (target).
4. Use the link context menu to open the *Foreign Key* dialog.

Name:

Source: Inventory Target: Warehouse

| Key | Column | Type |
|-----|--------------|---------|
| PK | InventoryID | VARCHAR |
| | WarehouseID | VARCHAR |
| | CurrentStock | INTEGER |
| | OnOrder | INTEGER |

| Key | Column | Type |
|-----|-------------|---------|
| PK | WarehouseID | VARCHAR |
| | Location | VARCHAR |
| | Capacity | NUMERIC |

Referential Integrity

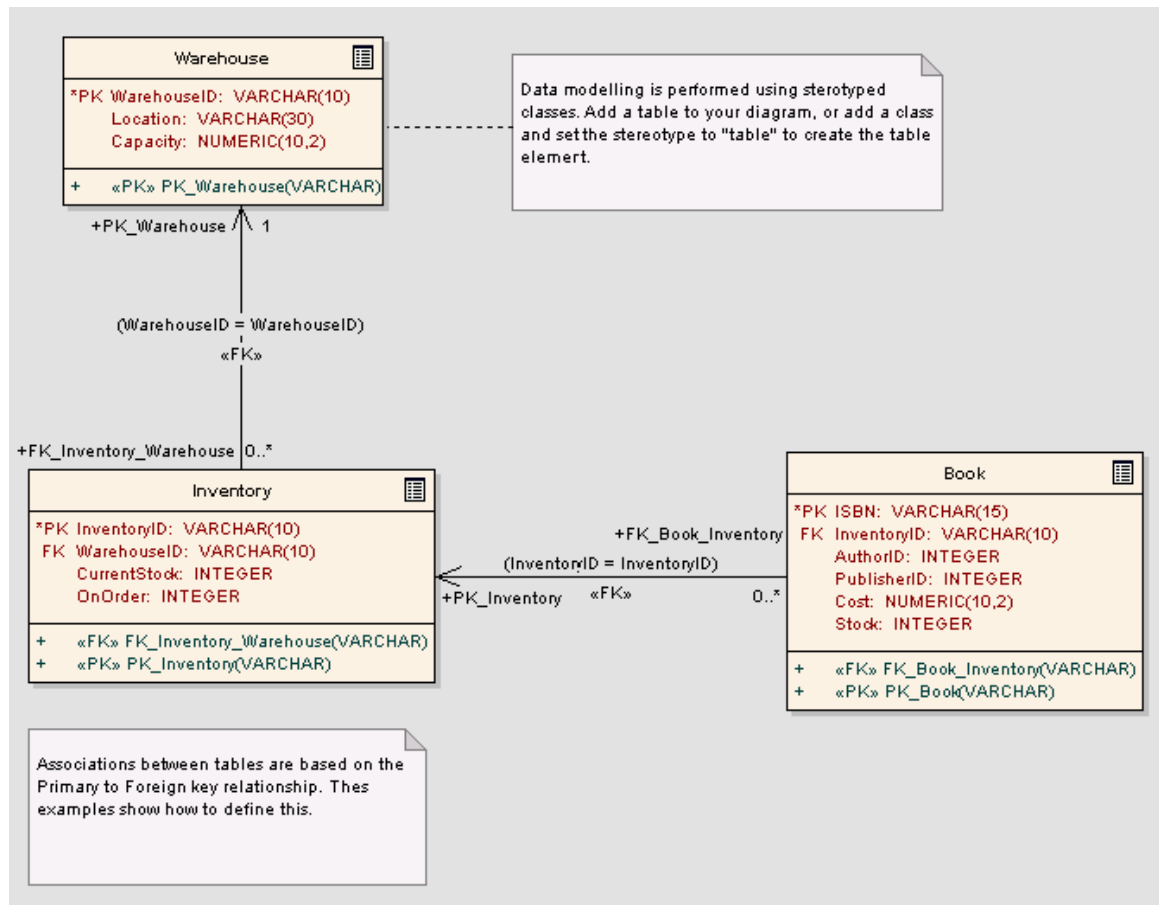
Delete Cascade Update Cascade

| Column | Type |
|-------------|---------|
| WarehouseID | VARCHAR |

| Column | Type |
|-------------|---------|
| WarehouseID | VARCHAR |

5. The default name for the foreign key can be edited.
6. Highlight the columns involved in the foreign key relationship.
7. Press **Save** to automatically generate the foreign key operations.

You have created a Foreign Key. The example below shows how this will look in a diagram:



- To create a composite foreign key, select the appropriate columns and press **Save**. The foreign key columns will be sorted according to datatype to match the datatypes of the composite targeted primary key. If desired, the order of the key columns can be changed using the **move up** and **move down** buttons.

Foreign Key Constraint ✕

Name:

Source: Table2 Target: Table1

| Key | Column | Type |
|-----|---------|----------|
| | t2_date | datetime |
| | t2_id | int |
| PK | t2_pk | int |
| | t2_name | varchar |

| Key | Column | Type |
|-----|----------------|----------|
| PK | t1_id | int |
| PK | t1_name | varchar |
| PK | t1_date_cre... | datetime |

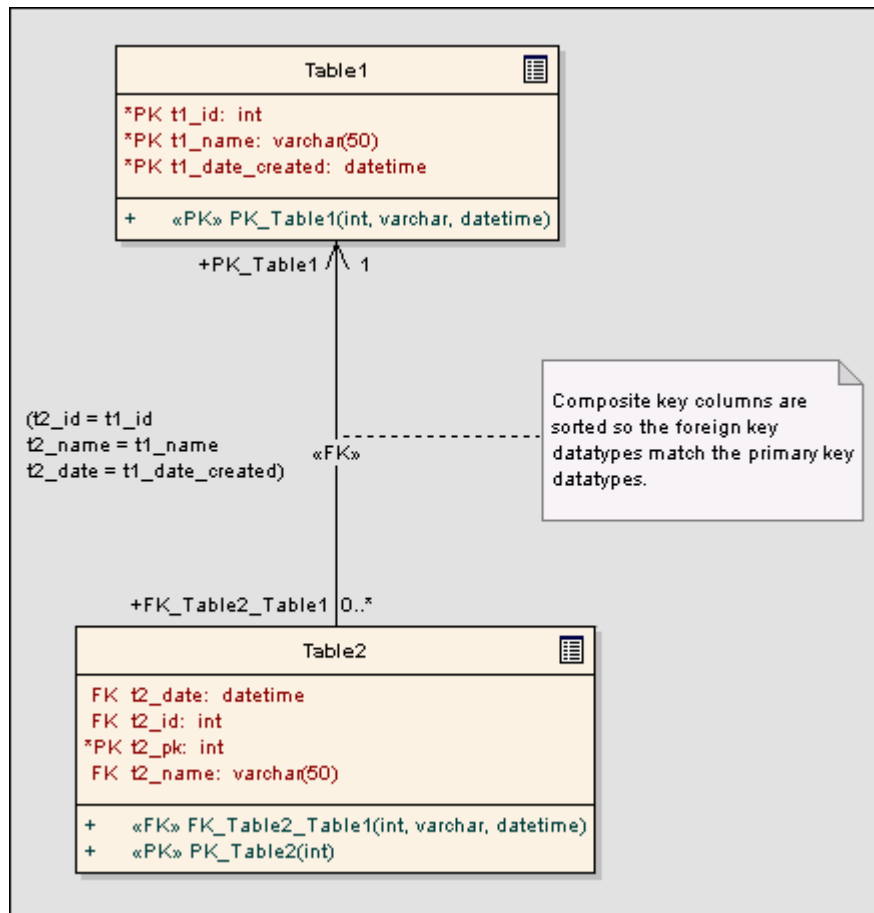
Referential Integrity

Delete Cascade Update Cascade

| Column | Type |
|---------|----------|
| t2_id | int |
| t2_name | varchar |
| t2_date | datetime |

| Column | Type |
|-----------------|----------|
| t1_id | int |
| t1_name | varchar |
| t1_date_created | datetime |

By using the feature a Foreign Key will be created. The example below shows how this will look in a diagram:



Hint: If you are defining a MySQL database and wish to use foreign keys, you will have to [set the table type](#) to allow this.

13.7 Stored Procedures

What is a stored procedure?

A stored procedure is a group of SQL statements that form a logical unit and perform a particular task. Stored procedures are used to encapsulate a set of operations or queries to execute on a database server. Stored procedures can be compiled and executed with different parameters and results, and they may have any combination of input, output, and input/output parameters.

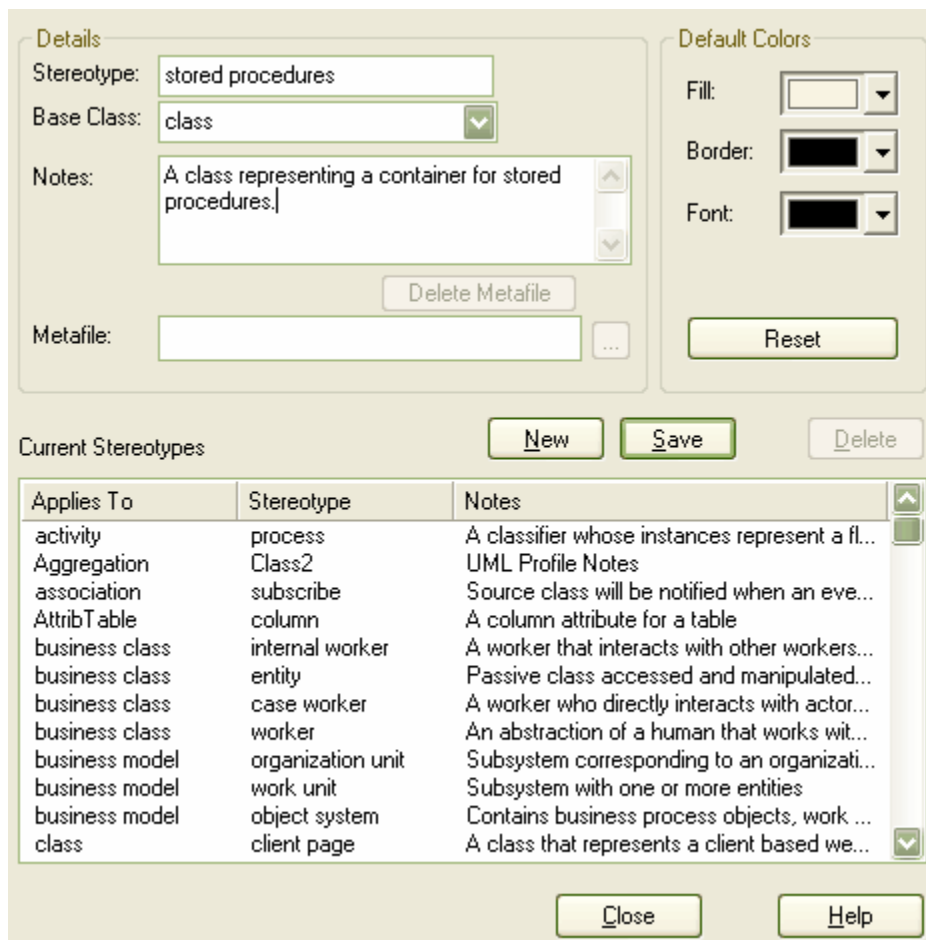
Enterprise Architect models stored procedures as operations of a class in accordance with the UML Profile for Data Modelling.

Note: Stored procedures are currently supported only for SQL Server and Oracle 9i and 10g.

Create a Stored Procedure Container Class

To create a stored procedure container class, follow the steps below:

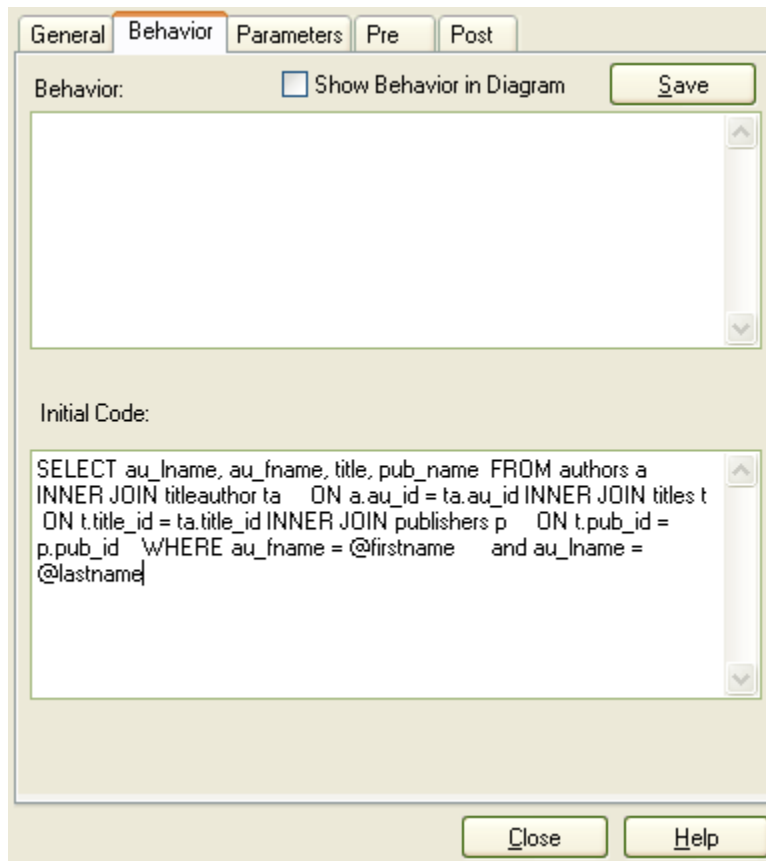
1. From the **Settings** menu, select UML | Stereotypes... and add a <<stored procedures>> stereotype as shown, selecting "class" as the Base Class.



2. Select a suitable diagram.
3. Open the *Structure* group on the UML Toolbox.
4. Left click the *Class* element in the list of elements and then click on the diagram.
5. In the Class Properties dialog, from the Stereotype dropdown list select "stored procedures".
6. Enter a name for the class.
7. Click OK to close the dialog. You now have a stored procedures container.
8. Open the Properties dialog again and from the Database dropdown list, select the target DBMS to model. The default database will appear if it has already been set.

Creating a Stored Procedure

1. On the Properties dialog, select the Procedures Detail tab and press the Stored Procedures... button.
2. Alternatively, select the stored procedures container and press F10 or select Features | Operations from the context menu.
3. Enter the name of the stored procedure, select the return type ("resultset" is the default), and ensure the stereotype is "proc".
4. Save.
5. To add parameters, select the procedure from the Operations list and select the Parameters tab.
6. Enter the parameter name and type. If the parameter is a length type, add the length in the drop down. For example, select varchar from the dropdown list, and add the length including the brackets. The type and length can also be entered directly into this field.
7. Save.
8. Select the *Behavior* tab and enter the body of the procedure into the Initial Code field. NB: Only add the statements required after the AS clause.
9. An example is shown below:



13.8 Views

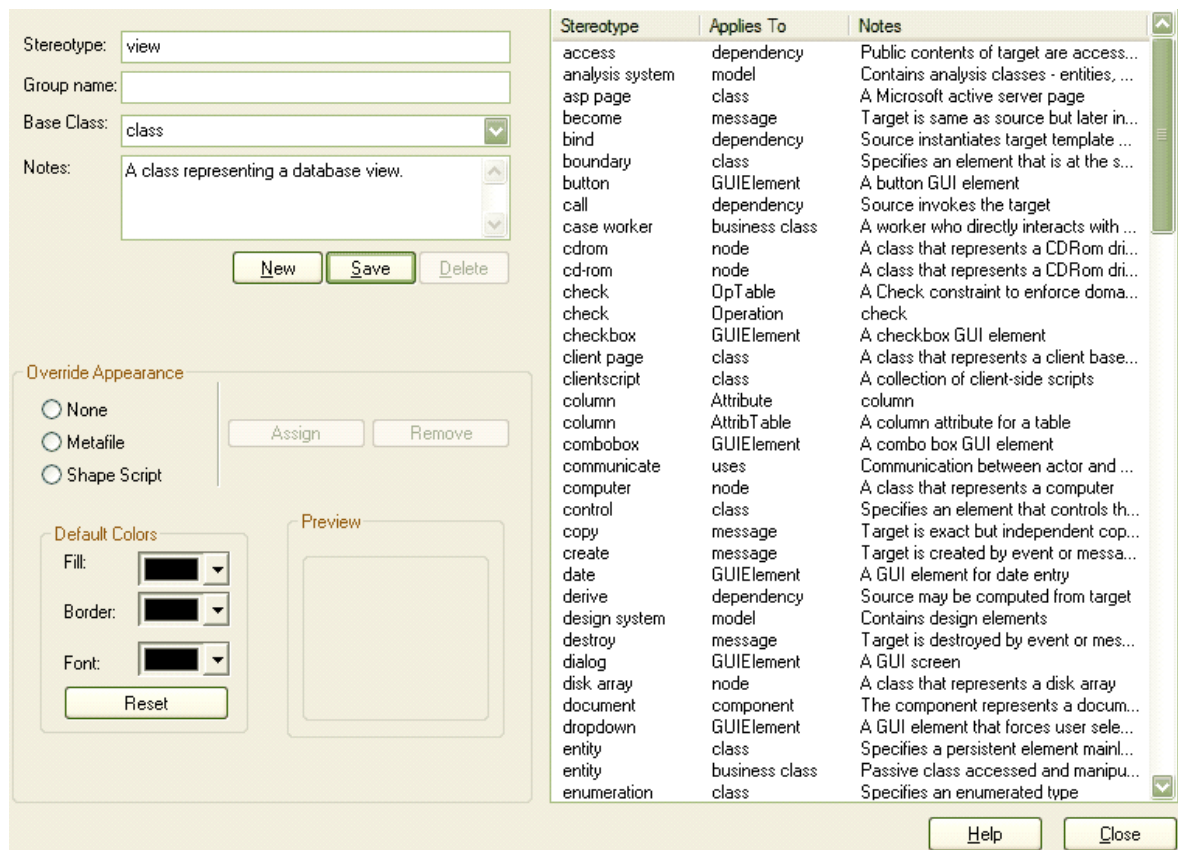
What is a View?

Note: Views are currently supported only for SQL Server, Oracle 9i and 10g and Sybase Adaptive Server Enterprise.

Create a View Class

To create a database view, follow the steps below:

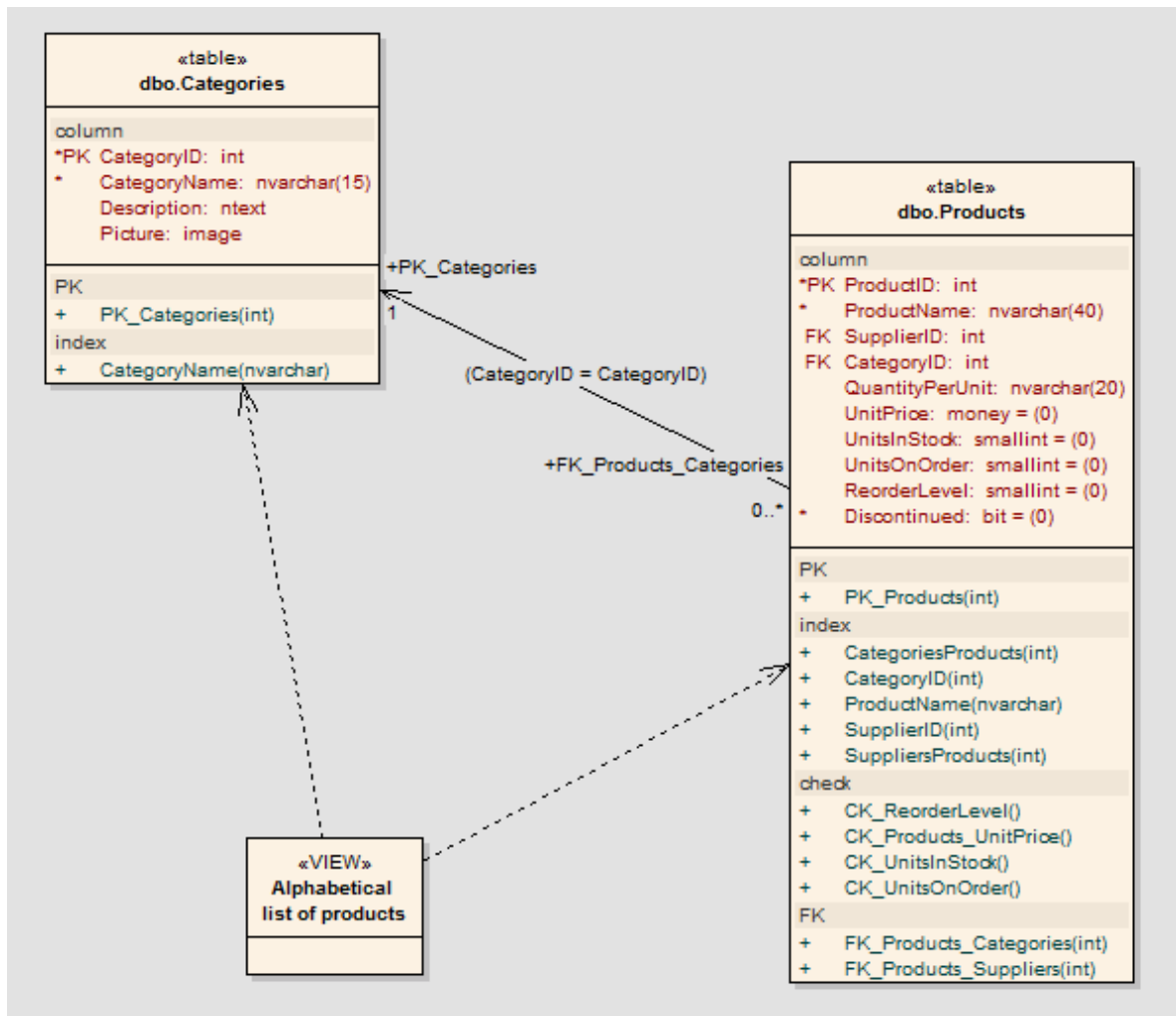
1. From the *Settings* menu, select *UML | Stereotypes...* and add a <<view>> stereotype as shown, selecting "class" as the Base Class.



2. Select a suitable diagram.
3. Open the **Structure** group on the UML Toolbox.
4. Left click the **Class** element in the list of elements and then click on the diagram.
5. In the Class **Properties** dialog, from the **Stereotype** dropdown list select **view**.
6. Enter a name for the view.
7. Click **OK** to close the dialog. You now have a database view.
8. Open the **Properties** dialog again and from the **Database** dropdown list, select the target DBMS to model. The default database will appear if it has already been set.

Creating a View

1. Create a **Dependency** link from the view class to the table or tables on which the view will depend.
2. On the view **Properties** dialog, select the **View Detail** tab and press the **View Definition...** button.
3. Enter the full view definition in the **View Definition** field.
4. Click **Save** to save your definition.
5. An example is shown below:



13.9 Indexes, Triggers and Check Constraints

What is an Index?

An index is a sorted look-up for a table. When it is known in advance that a table needs to be sorted in a specific order, it is usually worth the small processing overhead to always maintain a sorted look-up list rather than sort the table every time it is needed. In EA, an index is modeled as a stereotyped operation. On generating DDL, the necessary instructions for generating indexes are written to the DDL output.

What is a Trigger?

A trigger is an operation automatically executed as a result of the modification of data in the database, and will usually ensure consistent behavior of the database. For example, a trigger might be used to define validations that must be performed every time a value is modified, or might perform deletions in a secondary table when a record in the primary table is deleted. In EA, a trigger is modeled as a stereotyped operation. Currently EA will not generate DDL for triggers, but nonetheless they aid in describing and specifying the table structure in detail.

What is a Check Constraint?

A CHECK constraint enforces domain integrity by limiting the values that are accepted by a column.

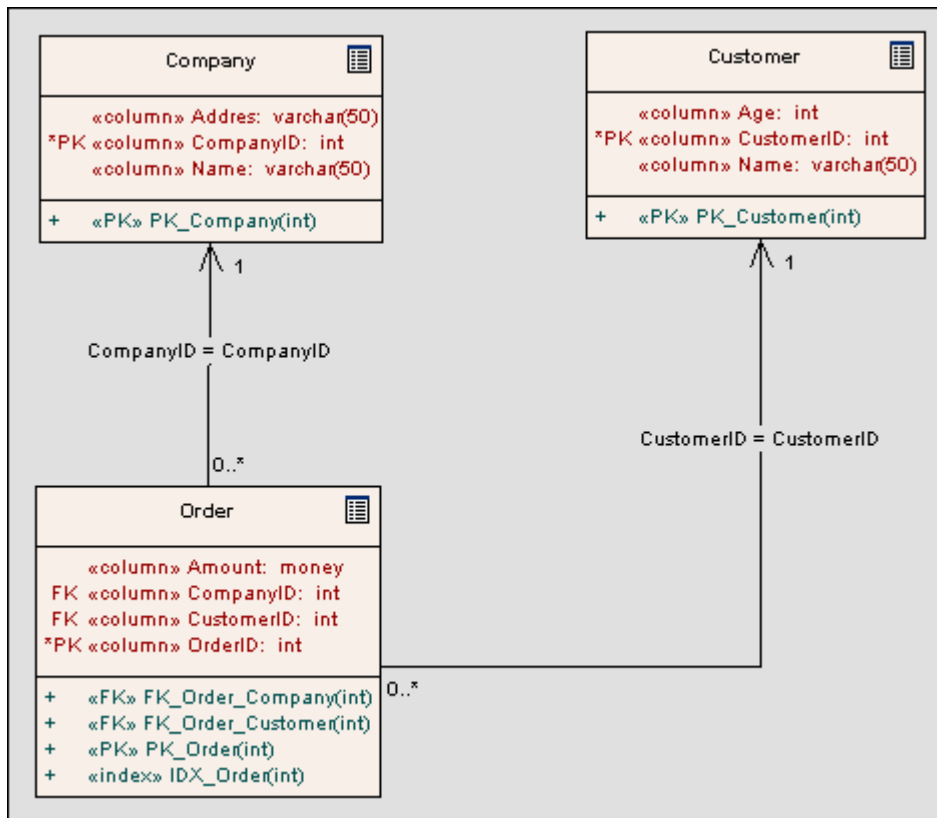
Create an Index or Trigger

1. Locate the required table in either a diagram or in the Project Browser.
2. Use the context menu to open the *Operations* dialog.
3. Add an operation (such as "IDX_CustomerID or TRG_OnCustomerUpdate" - the IDX_ and TRG_ prefixes are optional but help identify the operation).
4. Set the *Stereotype* for the Operation to "index" or "trigger" as appropriate - "check", "proc" and "unique" are also supported.
5. Enter the body of the trigger, procedure, or details of the check constraint in the Notes field of the Operation.
6. Select the *Operation* and go to the *Columns* tab.
7. Add the required columns in the desired order then press *Save* to save changes.

Create a Check Constraint

1. Locate the required table in either a diagram or in the Project Browser.
2. Use the context menu to open the *Operations* dialog.
3. Add an operation (such as "CHK_ColumnName").
4. Set the *Stereotype* for the constraint to "check" and press *Save* to save changes.
5. Select the constraint operation, then the *Behavior* tab.
6. Enter the entire check constraint clause (eg. col1 < 1000) in the Initial Code field and press *Save* to save changes.

The example below shows how an index will look in a diagram:



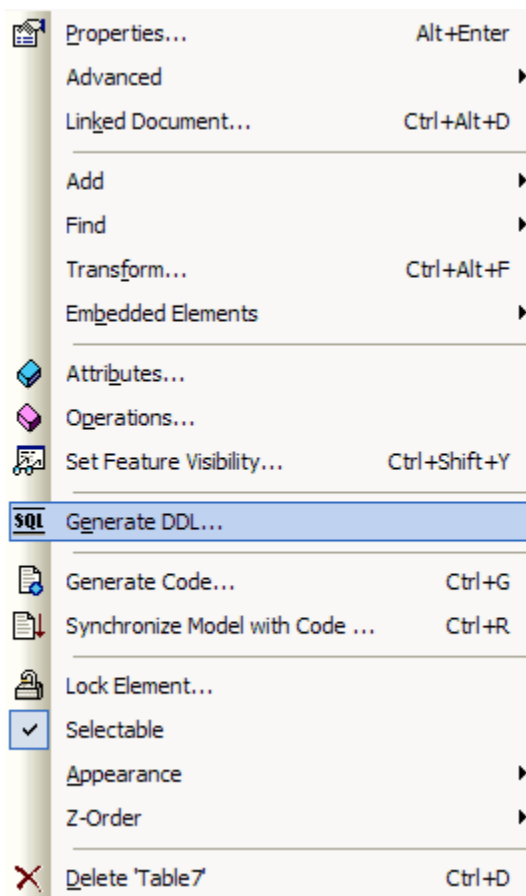
13.10 Generate DDL

Enterprise Architect will generate simple DDL scripts to create the tables in your model.

Generate DDL

To generate DDL, follow the steps below:

1. Select the table in the diagram to generate DDL for.
2. Right click to open the context menu and choose *Generate DDL*.



3. In the *Generate DDL* dialog, enter the *Filename* of the script to create.
4. Check the *Create Drop SQL* if you wish to include a 'drop table' command in the script.
5. Press *Generate* to create the DDL.
6. Press *View* to view the output (you will have to configure a DDL viewer in the Local Settings dialog first).

Table: Staff

Path: C:\Documents and Settings\John Redfen\Desktop\Staff.SQL

Options

Comment Level: None Use [] and [] as comment

Create Primary/Foreign Key Constraints

Generate Index/Constraints

Generate Triggers

Generate Stored Procedures

Create Drop SQL Use ; as SQL Separator

Use [] and [] around names

Generate Table Owner

Use Database: <database>

Use Alias if Available

View Generate Close Help

13.11 Generate DDL for a Package

To generate DDL for a package, follow the steps below:

1. Locate the required package in the Project Browser.
2. Right click to open the context menu.
3. From the *Code Engineering* submenu, select *Generate DDL*.

Note: Alternatively you can select *Generate Package DDL* from the *Project | Database Engineering* submenu.

4. The *Generate* dialog will appear.
5. You can check the *Include All Child Packages* to recursively generate DDL.
6. Press *Generate* to proceed. You will be prompted for file names as the process proceeds if required.

Root Package:

Options

Comment Level Use and as comment

Create Primary/Foreign Key Constraints

Generate Index/Constraints

Generate Triggers

Generate Stored Procedures

Create Drop SQL Use as SQL Separator

Use and around names

Generate Table Owner

Use Database

Use Alias if Available

File Generation

Single File

Individual file for each table

Select Objects to Generate Include all Child Packages

| Object | Type | Target File |
|------------|-------|---|
| People | Class | C:\Documents and Settings\John Redfen\De... |
| Staff | Class | C:\Documents and Settings\John Redfen\De... |
| Staff_Room | Class | |

13.12 Data Type Conversion Procedure

Once a database schema has been set up on an EA diagram (either by importing through ODBC or manually setting up the tables), the DBMS can be changed to another type and the column datatypes will be mapped accordingly.

To Map the DBMS type of a table to another DBMS type:

1. Open the Table Properties Dialog. The Database drop down will show the current DBMS for this table.
2. To map the column datatypes to another DBMS, select the target from the Database drop down and press the **Apply** button.

- The datatypes will be converted to match those of the new DBMS, and these will be reflected in any DDL generated from this table.

The screenshot shows a dialog box with the following fields and values:

- Name: Staff
- Stereotype: table
- Author: John Redfern
- Status: Proposed
- Scope: Public
- Complexity: Easy
- Database: MSAccess
- Keywords: DB2, InterBase, MSAccess, MySQL, Oracle, PostgreSQL, SQL Server 2000, SQLServer7, Sybase
- Phase: 1.0
- Version: 1.0

13.13 Data Type Conversion for a Package

The DBMS Package procedure or mapper allows the user to convert a package of database tables from one DBMS type to another DBMS type as well as providing the ability to change the ownership of table owners .

To Map the DBMS types of a package to another DBMS type:

- Right click on the package in the project tree, select *Code Engineering | Reset DBMS Options*.
- In the resulting Manage DBMS Options dialog, select the current DBMS from the *Current DBMS* drop down menu and the target DBMS from the *New DBMS* drop down menu.
- Ensure that the *Convert DBMS Type* checkbox is selected.
- If there are child packages that also require changing, check the "*Process Child Packages*" check box.
- Click *OK*. All Tables in the selected packages will be mapped to the new DBMS.

Convert DBMS Type

Current DBMS: MSAccess

New DBMS: Oracle

Change Table Owner

Current Owner:

New Owner:

Process Child Packages

OK

Cancel

To change the owner of the table or all of the tables in a package use the following instructions:

1. Right click on the package in the project tree, select *Code Engineering | Reset DBMS Options*.
2. In the resulting Manage DBMS Options dialog, enter the name for the new table owner in the *New Owner* field
3. From the *Current Owner* drop down select the current owner that is to be changed or select *<All>* to change the ownership of all of the tables in the package to the name defined in the *New Owner* field.
4. Ensure that the *Change Table Owner* checkbox is selected.
5. If there are child packages that also require changing, check the "*Process Child Packages*" checkbox.
6. Click *OK*. All Tables in the selected packages will have their ownership changed.

Convert DBMS Type

Current DBMS:

New DBMS:

Change Table Owner

Current Owner: <All>

New Owner: Andrew Loynes

Process Child Packages

OK

Cancel

For more information relating to setting the table owner see the Set Table Properties topic, to display the [Set Table Owner](#) in the current diagram see the [Set Appearance Options](#) section.

13.14 DBMS Datatypes

When setting up your data modeling profile, you can customize the datatypes associated with a particular DBMS using the Database Datatypes screen. This screen allows you to add and configure custom data types. For some data types you will need to add the size and precision, defaults and maximum values.

The Database Datatypes screen is available by selecting *Database Datatypes* from the *Configuration* menu. You may also add a DBMS product and configure the inbuilt data types.

| Product | Datatype | Size Unit | Default | Max |
|---------|---------------|---------------------|---------|-----|
| MySQL | BIGINT | | | |
| MySQL | BIT | | | |
| MySQL | BLOB | | | |
| MySQL | BOOL | | | |
| MySQL | CHAR | Length | 10 | 255 |
| MySQL | DATE | | | |
| MySQL | DATETIME | | | |
| MySQL | DECIMAL | Precision and Scale | (10,0) | 24 |
| MySQL | DOUBLE | Precision and Scale | (10,2) | 53 |
| MySQL | DOUBLE PRE... | Precision and Scale | (10,2) | 53 |
| MySQL | FLOAT | Length | 0 | 24 |
| MySQL | INTEGER | | | |
| MySQL | NUMERIC | | | |

13.15 Import Database Schema from ODBC

Enterprise Architect supports importing database tables from an ODBC data source. Tables will be imported as stereotyped classes with suitable data definitions for the source DBMS.

Import Database Tables and Stored Procedures

To import database tables and stored procedures, follow the steps below:

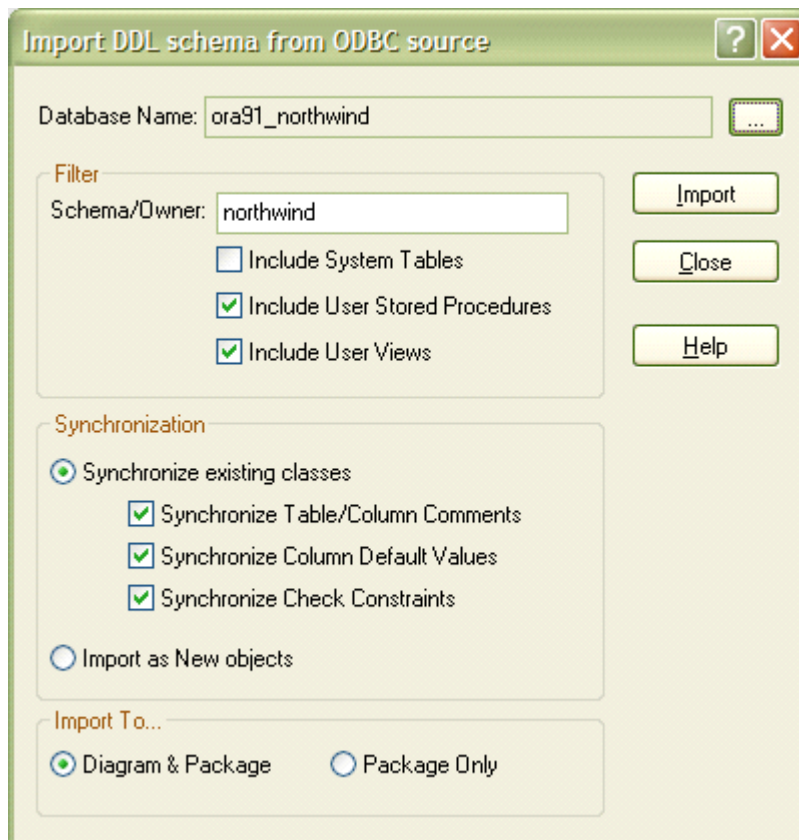
Note: Import of stored procedures and views is supported only for SQL Server, Oracle 9i and 10g and Sybase Adaptive Server Enterprise.

1. Select any package in the *Logical View*.
2. To import into the package only, right-click on the package to bring up the context menu.
3. Select *Code Engineering* | *Import DDL schema from ODBC*.

4. To import into a diagram, select a suitable diagram in the selected package.
5. Right click on the diagram to open the context menu.
6. Select *Import DDL schema from ODBC*.

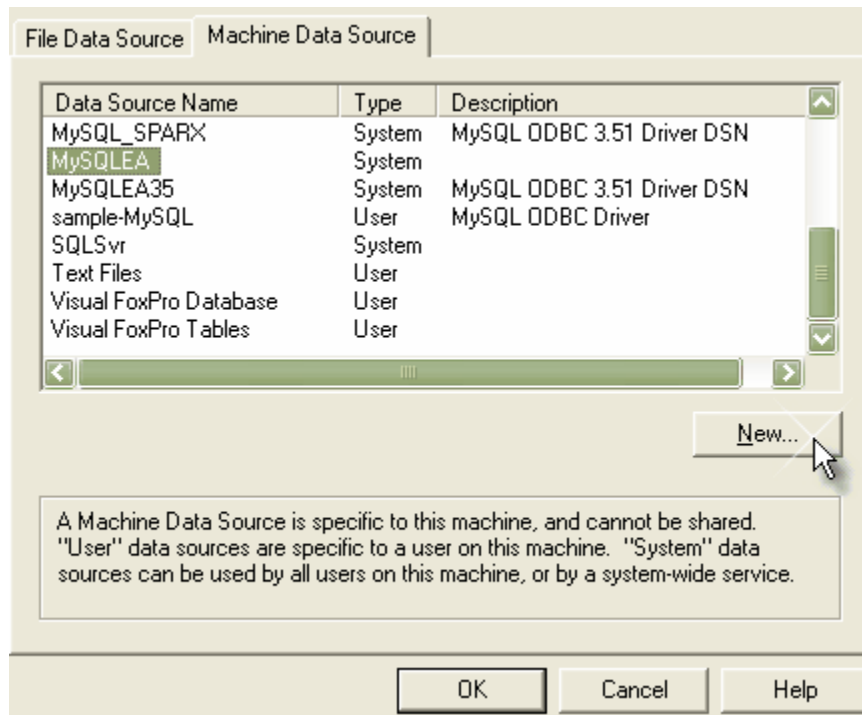
Note: Alternatively you can select *Import DDL from ODBC* from the *Project | Database Engineering* submenu.

7. Select an ODBC data source, then press Import to start the import. When synchronizing existing classes, select the appropriate check box to determine whether the model comments, default values or constraints are to be synchronized with the ODBC tables. Select "Include User Stored Procedures" and/or "Include User Views" to also import stored procedures and views.



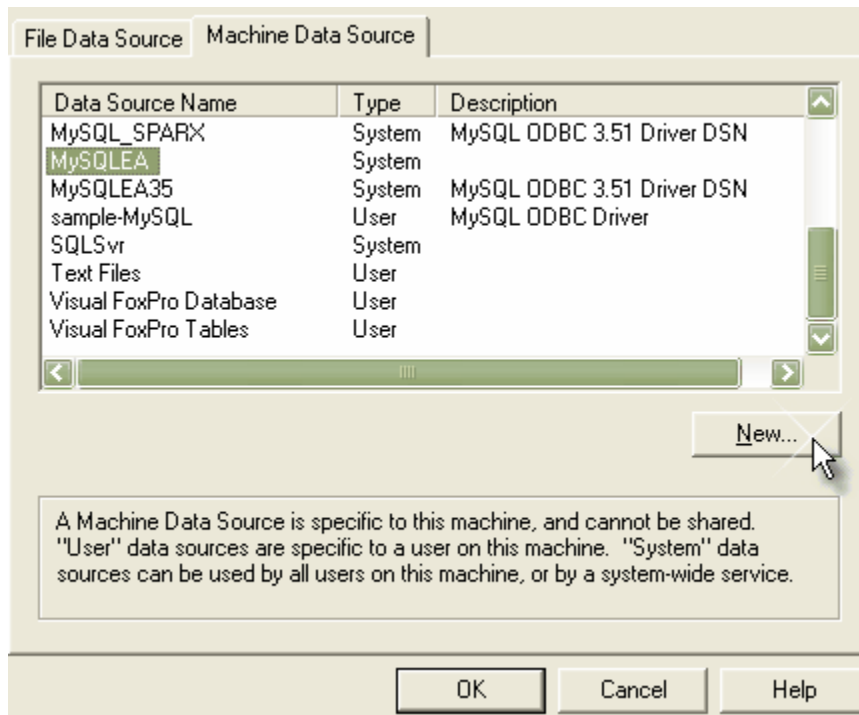
Note: It is only possible to import into a diagram if it is in the selected package. If a diagram from another package is open, a message will give the option to cancel the import or to continue importing into the package only. The Dialog includes options to import into *Diagram & Package*, or *Package Only*. If no diagram is open, the *Package Only* button will be selected and the options will be disabled. If the open diagram is in the selected package, either option can be selected.

8. Choose an ODBC data source. Select a suitable data source from the ODBC dialog (ODBC must be installed and configured on your machine for this to work correctly).



13.15.1 Select a Data Source

Importing DDL from existing data sources requires you to have installed and configured a suitable ODBC connection. From the Import DDL dialog you can select the ODBC data source using the standard windows ODBC set-up dialog.



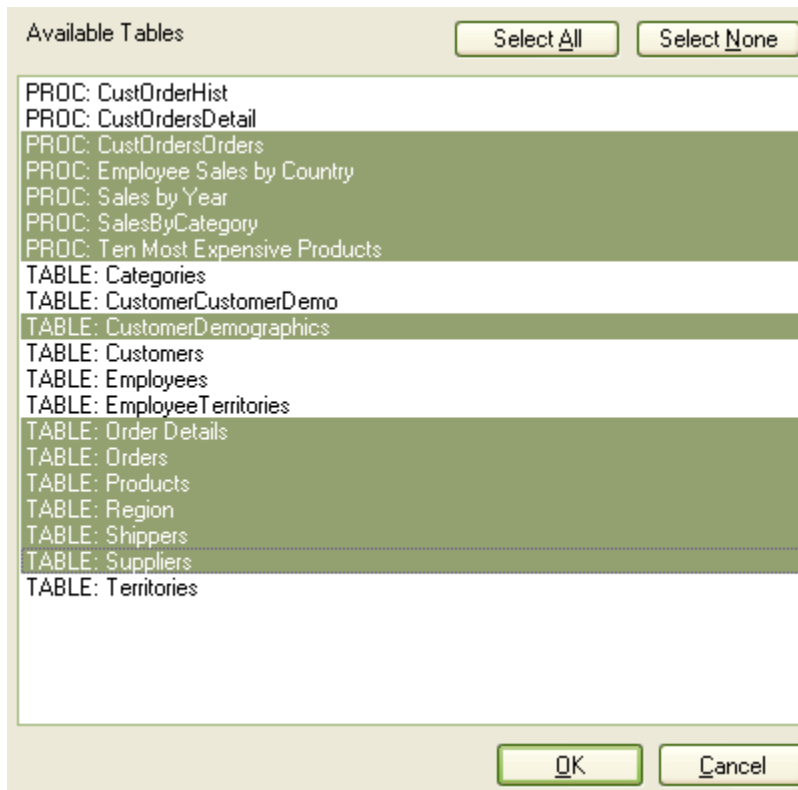
13.15.2 Select Tables

Once you have opened the ODBC data source, EA will acquire a list of tables and stored procedures suitable for importing. This is presented in a list form for you to select from. Highlight the tables and/or stored procedures you wish to import and clear those you do not require.

Selection shortcuts:

- To select all tables, press **Select All**
- To clear all tables, press **Select None**
- Hold down the **Ctrl** key while clicking on tables to multiple select
- Hold down **Shift** and click on table to select a range

The selection dialog:



13.15.3 The Imported Class Elements

When you import DDL table definitions they are converted to stereotyped classes according the UML Data Modeling Profile.

If any Stored Procedures are imported, a class stereotyped as a <<stored procedures>> container is created having the same name as the database being imported. The stored procedures are represented as operations in this class.

The image below shows some example tables imported into the model using an ODBC data connection:



Part

14

14 Creating Documents

The production of documentation is essential to realizing the full benefit of EA. EA outputs high quality documentation in either [RTF](#) or [HTML](#) format. The RTF formatting can be modified directly with [RTF Style templates](#) to alter the look and feel of generated output. Using MS Word you can further enhance the output by connecting and interweaving model output in linked documents.

There are many ways to specify the EA content being documented:

- A package and/or its child packages can be documented by manually highlighting a package and selecting a documentation control.
- Embedded packages can be specified for [exclusion](#) if child packages are recursively documented.
- A package can be linked to an RTF document template to simplify generating consistent types of documentation (e.g., Use Case Reports) using the [Documents feature](#).
- Packages can be selected, grouped, and ordered together in a manner different from the project view by creating "[Virtual Documents](#)".

14.1 RTF Documents

Rich Text Format Documentation

Rich text reports are documents produced by Enterprise Architect in Rich Text Format (RTF), a format common to many word processors. In particular it is targeted at Microsoft Word™, which provides the ability to link together a number of rich text documents into a single master document.

Typically you will create a Word master document, then some Enterprise Architect RTF reports. These are linked back into sub-sections of the master document and refreshed as required during project development. In this way the project document becomes an easy to manage and feature-rich work product.

By adding tables of contents, figure tables, header & footers and sections you can manage a complex document with relative ease. Simply update the Enterprise Architect RTF reports then refresh the links in MS Word.

Features

Enterprise Architect v 5.0 has introduced a new fully-featured RTF Document generator. Some of the features include:

- Powerful WYSIWYG RTF template editor support allowing:
 - Headers and Footers
 - Images
 - Indexes
 - Tabular Sections
 - Nested Sections
 - All model elements, connectors, diagrams and their properties
 - Template import and export using XML
 - Basic templates supplied for customization
- New document generator:
 - Simplified options
 - Generates complex documents based on new RTF templates
- New embedded RTF viewer
 - View RTF documents generated in EA directly within EA

More Information

For More Information, see:

- [Create a Rich Text Document](#)
- [RTF Dialog Options](#)
- [RTF Document Options](#)
- [RTF Templates Dialog](#)

See Also

- [The Legacy RTF Dialog](#)
- [Using MS Word](#)
- [Other Documents](#)
- [Custom Language Settings](#)

14.1.1 Generate RTF Document

Creating a rich text document is a simple and flexible process. An RTF document is always based on a package in your project. To produce a document, you need to select the package you wish to report on in the Project Browser window.

Tip: Reports can be configured to include all packages within a parent package, or just the top level.

Once you have your package selected, you will use the context menu to open the RTF Report dialog and configure the details of your document. This next section will guide you through creating a rich text report.

How to Open the RTF Report Dialog

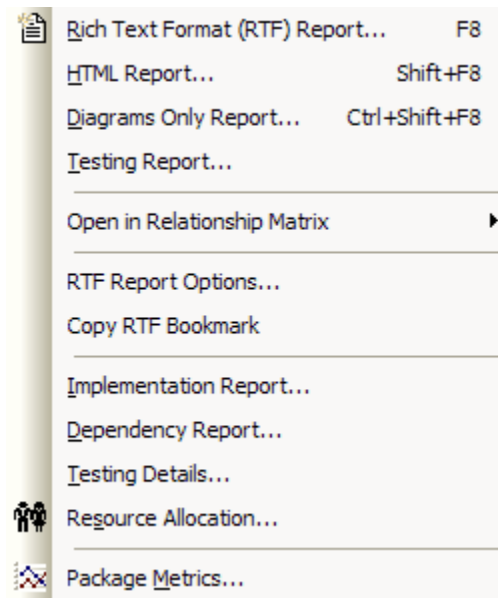
1. In the Project Browser, right click on the package you need to create documentation for, to open the context menu. Select *Rich Text Documentation* from the *Documentation* submenu.

-OR-

2. Select *Rich Text Format Report* from the *Project | Documentation* submenu.

-OR-

3. Use the keyboard shortcut *F8*.

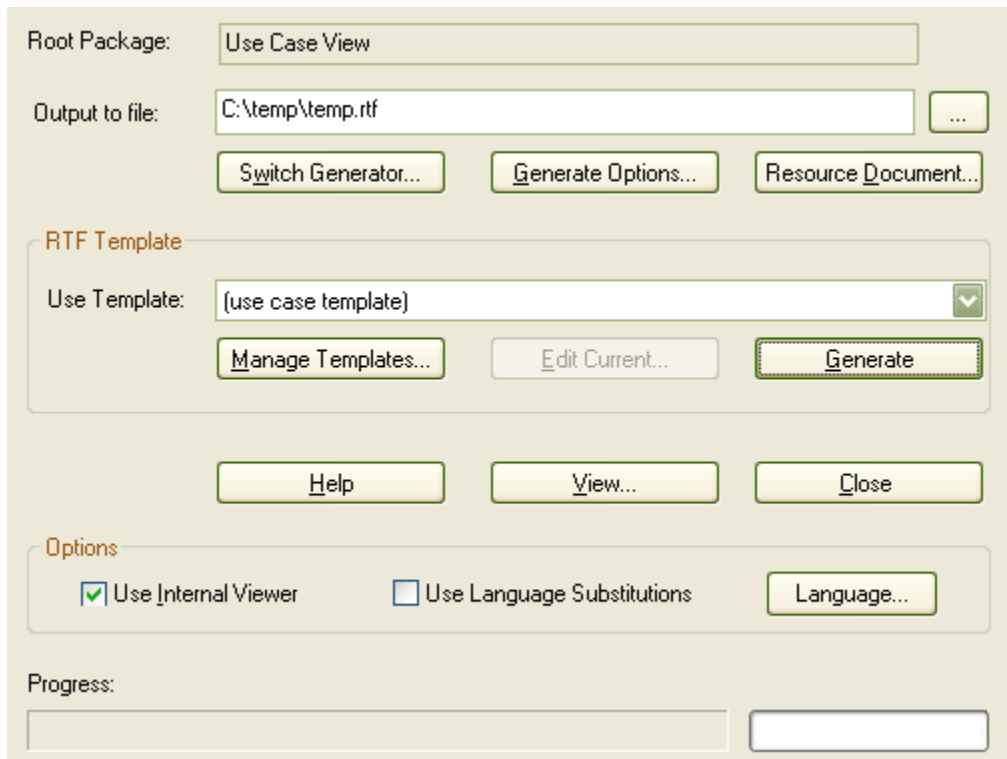


See [The RTF Report Dialog](#) and related topics for more information.

14.1.1.1 RTF Report Dialog

The RTF Report dialog allows you to set the exact contents and look and feel of your report. The dialog has the following functions:

Note: *The enhanced RTF Document Generator is not available with the EA Lite versions of EA and the Desktop Edition of EA.*



Control

Root Package
 Output to file

 Switch Generator
 Generate Options
 Resource Document
 Use Template

 Manage Templates
 Edit Current

 Generate
 Help
 View

 Close
 Use Internal Viewer

 Use Language Substitutions

Function

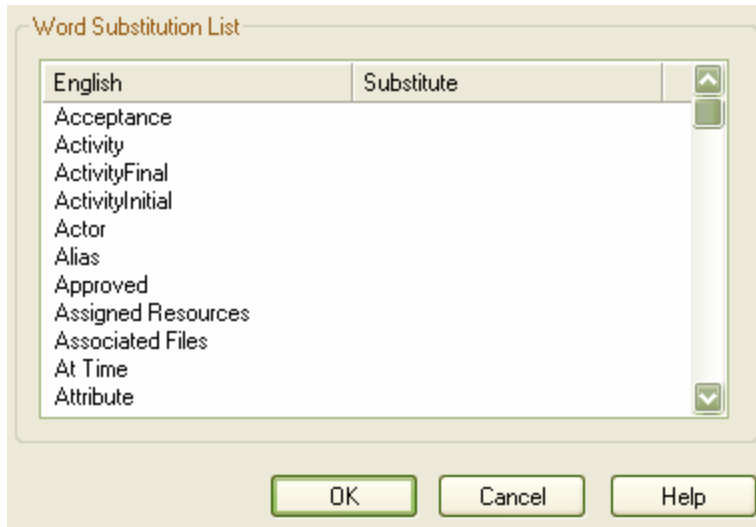
The currently selected package in the Project View.
 The location and filename of the generated documentation. The Browse [...] button navigates to the location.
 Toggles between the RTF Legacy Generator and the Enhanced Template driven Generator.
 Opens the [Document Options dialog](#).
 Saves the current options as a document definition.
 Specifies which RTF template to apply to the document generation.
 Opens the [RTF Templates dialog](#).
 Opens the currently named template in the RTF Template Editor.
 Generates the document.
 Opens this help page.
 Launches the generated RTF Documentation in the Windows default RTF file viewer, or in the EA internal viewer if the "Use Internal Viewer" option is selected.
 Closes this dialog.
 If set then pressing the View button launches the generated RTF Documentation in the EA internal viewer. Otherwise, launches the generated RTF Documentation in the Windows default RTF file viewer.
 Switches on Custom Language word substitutions

Language

Goes to the [Word Substitution](#) dialog. This allows you define translations of technical terms from English into any other language for direct substitution into RTF documents.

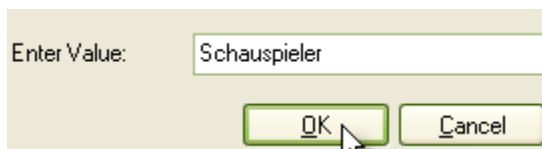
14.1.1.2 Word Substitution

The Word Substitution dialog allows you to define translations of technical terms, in particular field names used in EA, into languages other than English for direct substitution into RTF documentation.



To add a translation for a term:

1. Double-click on the term in the *English* column in the *Word Substitution List*.
2. Enter the foreign language translation and click OK.

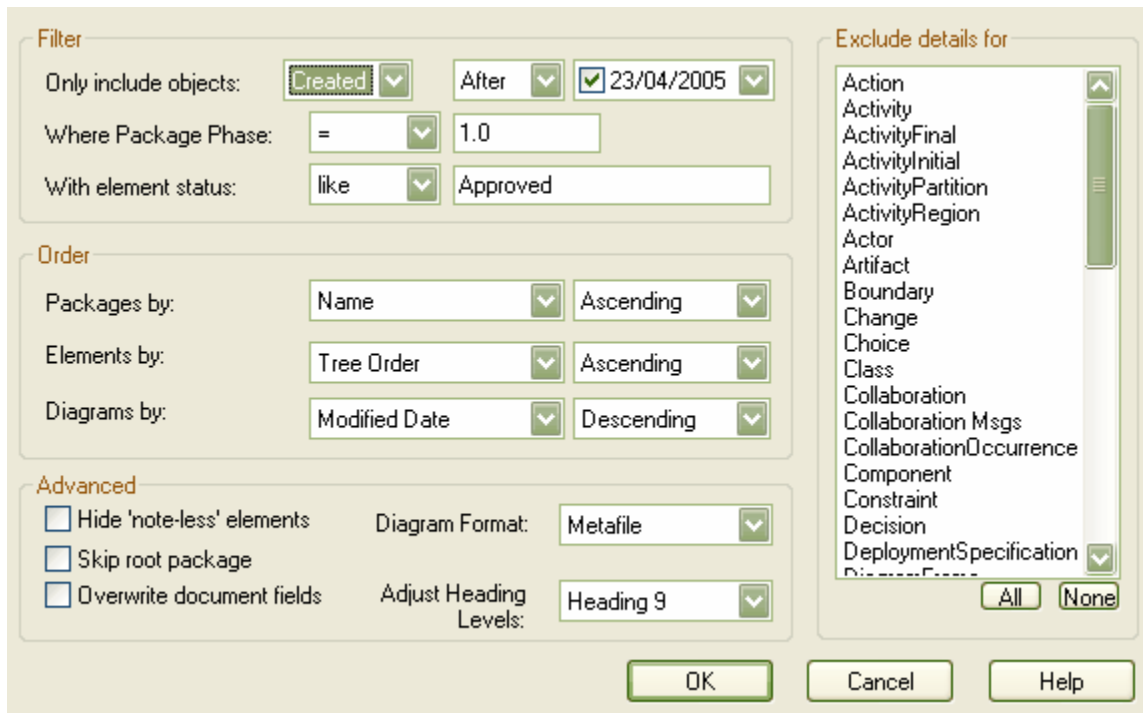


14.1.2 RTF Document Options

The RTF Document Options dialog allows you to set the filter and order the elements. This dialog can be opened from two different places, which affects the persistence of options selected:

1. When you open this dialog by pressing the *Generate* button on the [Generate RTF Documentation](#) dialog, you can create filtering settings here for the current document set to be run. Selections are non-persistent, and are reset when you select a different template.
2. When you open this dialog by using the *File | Document Options* command from the main menu of the [Template Editor](#) dialog, settings made here will be saved with the template as the default settings for any run of this report.

The Dialog has the following functions:



Control

- Only include objects
- Where Package Phase
- With element status
- Packages by
- Elements by
- Diagrams by
- Hide 'note-less' elements
- Diagram Format
- Skip root package
- Overwrite document fields
- Adjust Heading Levels

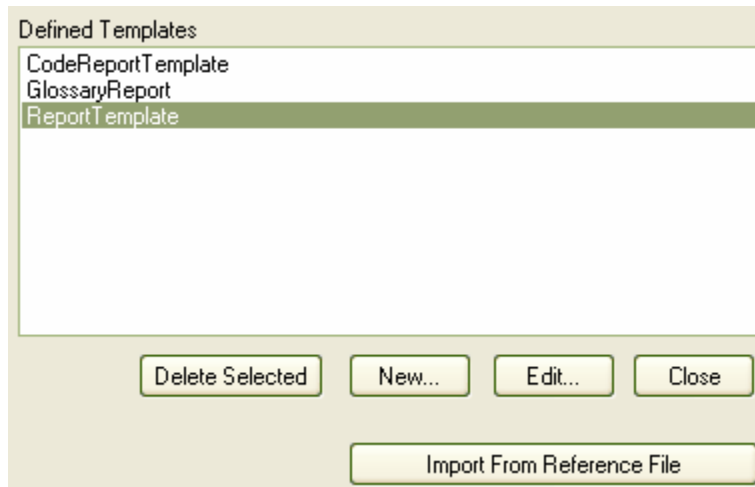
Function

- Allows elements to be filtered according to date created or modified.
- Allows elements to be filtered according to the value of the package phase field.
- Allows elements to be filtered according to its status.
- Orders packages in the generated documentation in either ascending or descending order of Name, Tree Order, Modified Date or Created date.
- Orders elements in the generated documentation in either ascending or descending order of Name, Tree Order, Modified Date or Created date.
- Orders diagrams in the generated documentation in either ascending or descending order of Name, Tree Order, Modified Date or Created date.
- Excludes all elements without notes from the documentation.
- Sets the diagram format for the images included within the documentation to either Metafile or Bitmap.
- Excludes the parent package from the documentation and includes only the child packages.
- When this option is unchecked, fields defined in a document section are generated with the appropriate values populated in these fields. Turning on this option will replace the fields with actual text.
- Setting this option will allow the RTF Generator to automatically adjust template headings based on the model depth

| | |
|---------------------|---|
| Exclude details for | Excludes all elements of the selected type or types to be excluded from the generated document. |
| OK | Applies the options to the documentation to be generated. |
| Cancel | Closes the dialog without applying the options. |
| Help | Opens the help to this page. |

14.1.3 RTF Templates Dialog

The *RTF Style Editor* allows you to edit the RTF associated with various sections of the RTF Report facility in Enterprise Architect. You would typically use this functionality to customize a report look and feel for your company or client. The **RTF Templates Dialog** is opened by pressing the *Manage Templates* button on the [Generate RTF Documentation](#) dialog. The dialog has the following functions:



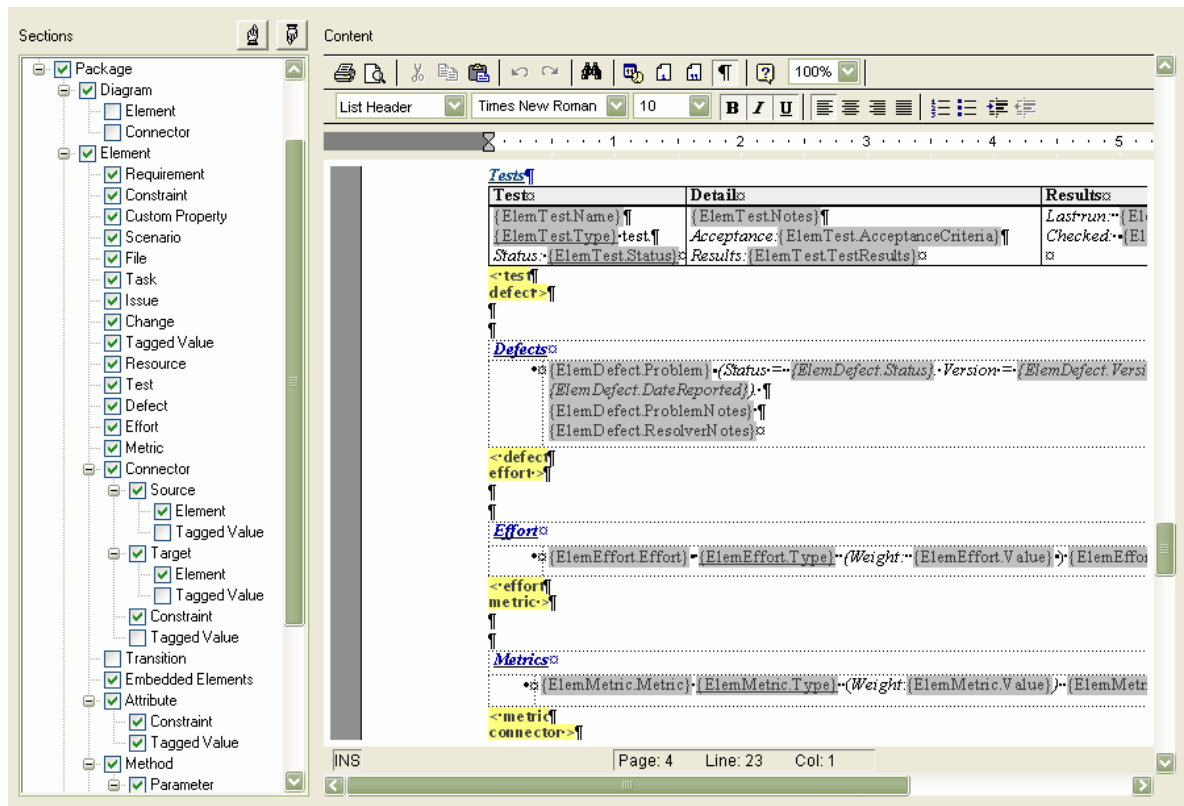
| Control | Function |
|----------------------------|--|
| Delete Selected | Deletes the selected template. |
| New | Creates a new template. These can be based on an existing template or you can start from scratch with a blank template. To make it easier to get up-and-running, EA provides a basic template with default settings that you may like to base your templates on. |
| Edit | Opens the RTF Template Editor . |
| Close | Closes this dialog. |
| Import From Reference File | Allows you to import RTF Templates saved to XML files using the <i>Tools Export Reference Data</i> command. |

14.1.3.1 New RTF Style Template Editor

The RTF Style Template editor allows for the creation and editing of custom RTF templates to define any outputted RTF documentation. The Style Template Editor lets you select particular model elements and then to specify, from the element type, the fields for inclusion into the generated document.

Formatting styles may be defined in the Style Editor, and items such as tables of contents, headers and more may be added to the document.

For information regarding specific commands used to alter the format of the RTF documentation, see the entries under the [RTF Template Editor Commands](#) section.

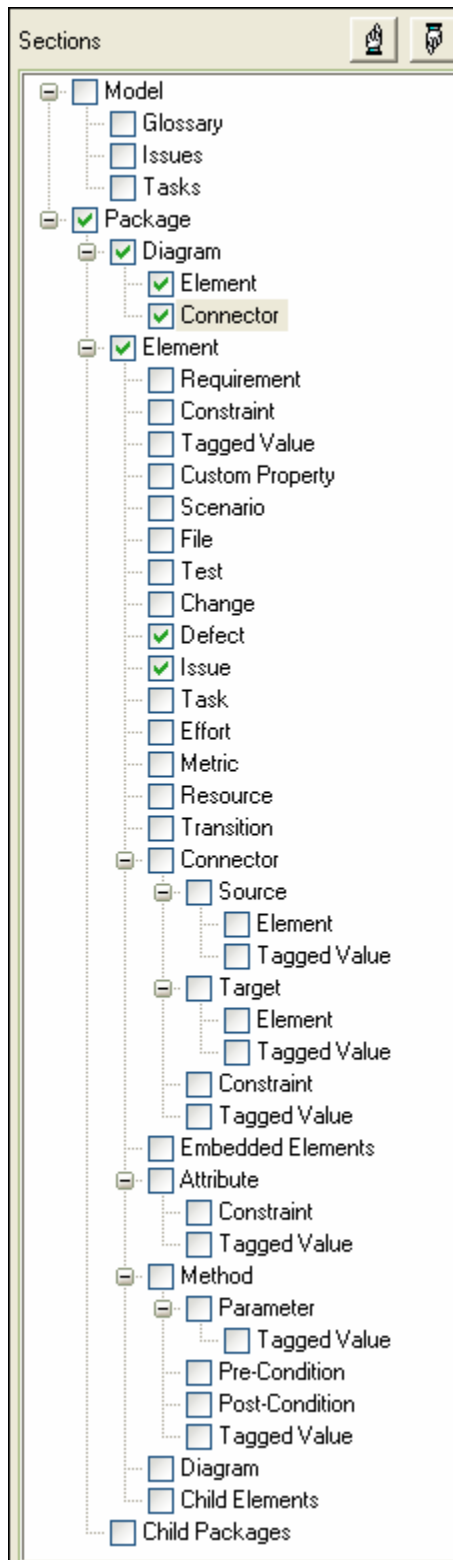


14.1.3.2 Selecting Model Elements for Documentation

To select Model elements for documentation, open the **Sections** tree of the RTF Template Editor. To add an element, check the box next to the element's name. Selecting a model element will add that element as section tags in the Content pane of the Document Template Editor. The position of the section tags within the Sections tree determines the position of the model element in the Content section of the RTF Template editor. For encapsulated elements, selecting a child element will automatically select the parent also.

To move a model element to a different position in the documentation template, select the element and then use the up and down arrows to move that element.

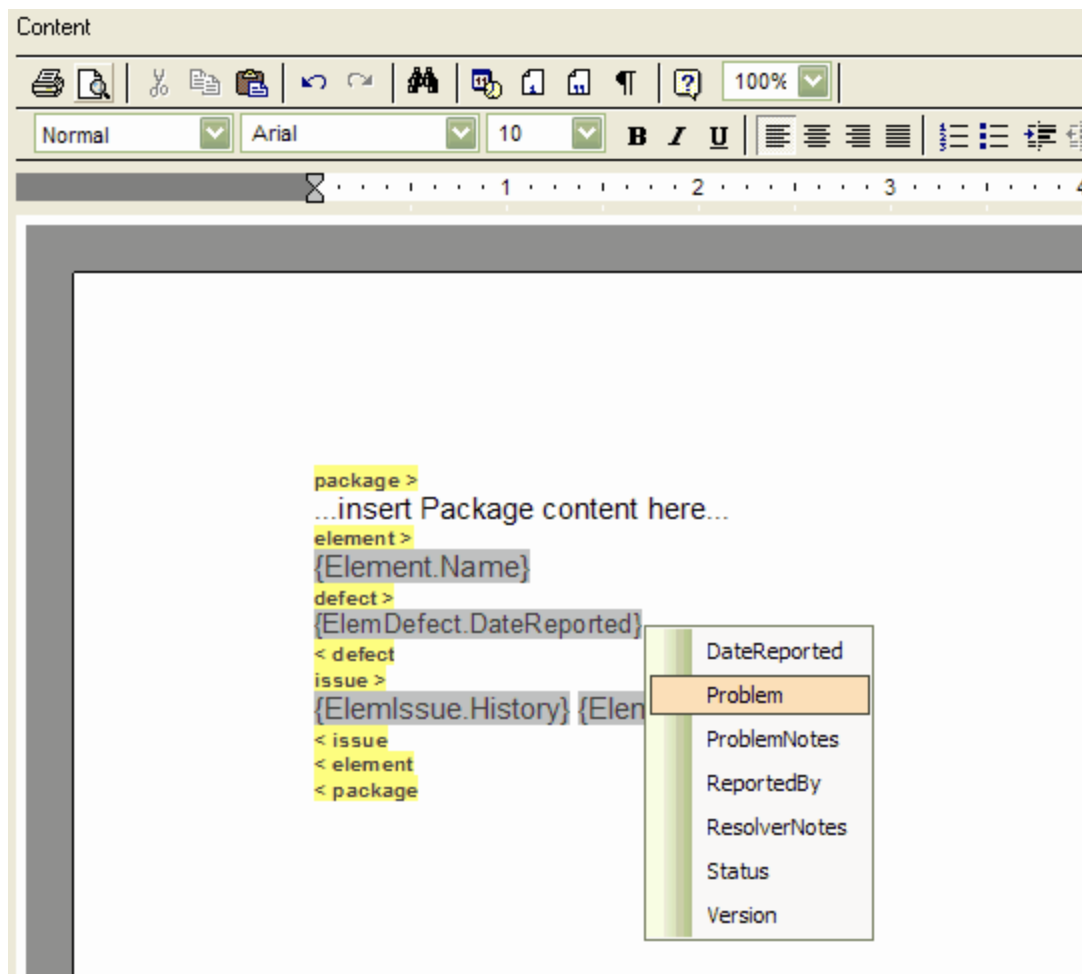
For example, the section tag for Diagram->Element will appear in the content section of the template editor before the section tag for Diagram->Connector.



14.1.3.3 RTF Template Editor - Adding Content

The RTF Template editor uses tags to arrange the content layout of the documentation. To insert a model element tag, use the [Selection](#) section of the Template editor. When a selection has been made, model element tags will be inserted into the *Content* section of the Editor. The Yellow highlighted model element names specify the beginning of a tag, which is represented by "sectionname >", and the end of the model element is shown as "< sectionname"

To add model element content, right click in the area between Tag Heading and Tag End. This will display a model element "type"-sensitive list of element fields that can be added to the RTF Documentation. Any additional information entered between the document tags will be included in the generated RTF Documentation.



14.1.3.4 RTF Template Editor Tabular Sections

The RTF Template Editor supports rendering a section as a table. This section describes how to render Document Sections in a tabular format. A tabular section is defined as a table containing any number of columns, *but with only TWO rows*. The first row is used to describe the headings of the columns. The Second row defines the output, which is then rendered iteratively for every occurrence of the section in question.

Example tabular section:

```
model >
```

Model Glossary

```
glossary >
```

| Term | Type | Meaning |
|----------------------|----------------------|-------------------------|
| {ModelGlossary.Term} | {ModelGlossary.Type} | {ModelGlossary.Meaning} |

```
< glossary  
< model
```

In this example the Model->Glossary section is defined as a tabular section. This will render the following document output:

Model Glossary

| Term | Type | Meaning |
|-------------------------|-----------|--|
| Accounting Periods | Business | A defined period of time whereby performance reports may be extracted. (normally 4 week periods). |
| Association | Technical | A relationship between two or more entities. Implies a connection of some type - for example one entity uses the services of another, or one entity is connected to another over a network link. |
| Class | Technical | A logical entity encapsulating data and behaviour. A class is a template for an object - the class is the design, the object the runtime instance. |
| Component Model | Technical | The component model provides a detailed view of the various hardware and software components that make up the proposed system. It shows both where these components reside and how they inter-relate with other components. Component requirements detail what responsibilities a component has to supply functionality or behavior within the system. |
| Customer | Business | A person or a company that requests An entity to transport goods on their behalf. |
| Deployment Architecture | Technical | A view of the proposed hardware that will make up the new system, together with the physical components that will execute on that hardware. Includes specifications for machine, operating system, network links, backup units &etc. |
| Deployment Model | Technical | A model of the system as it will be physically deployed |

14.1.3.5 RTF Template Editor Child Sections

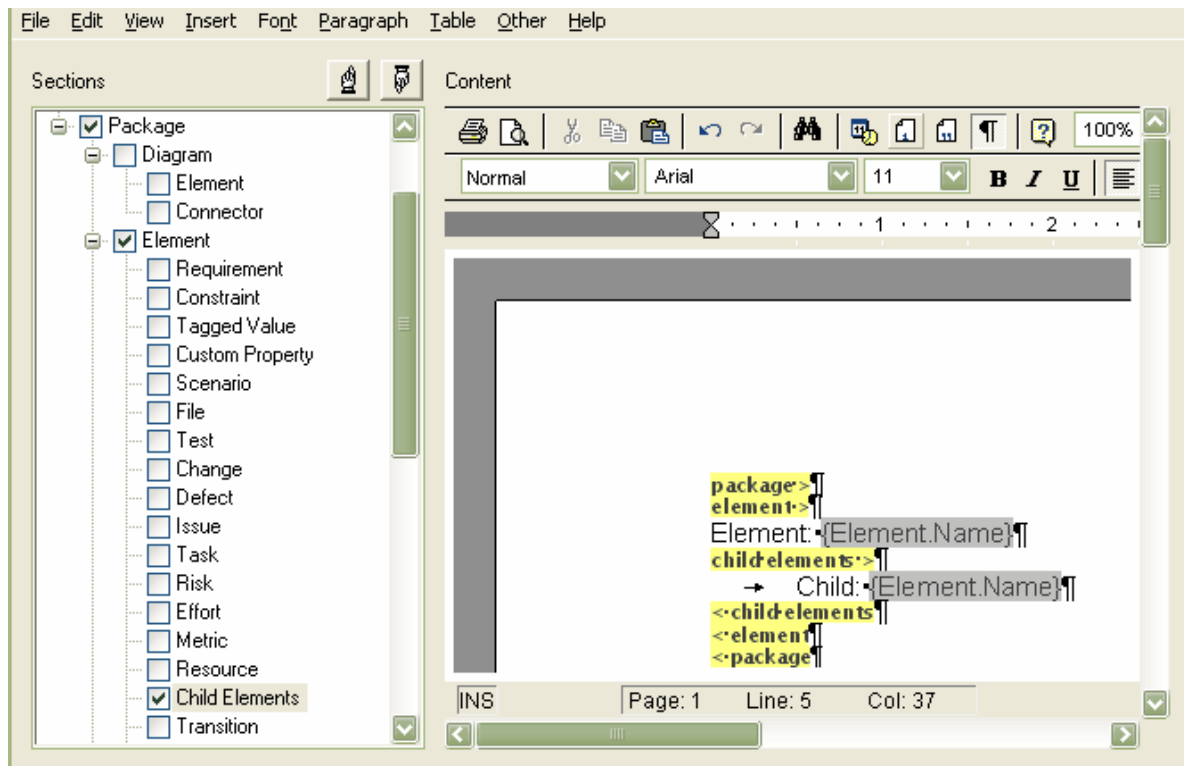
Rendering Child Sections

Child sections may be rendered in RTF documentation using one of two different methods. One way is to render model elements directly into the RTF as defined by the section's content and fields. The other is to

render indirectly to the RTF by using a parent section to describe the content. The latter option occurs as a result of the creation of a section which has a placeholder section tag (i.e. no content within the tags). This method is used to create recursive documentation of child packages.

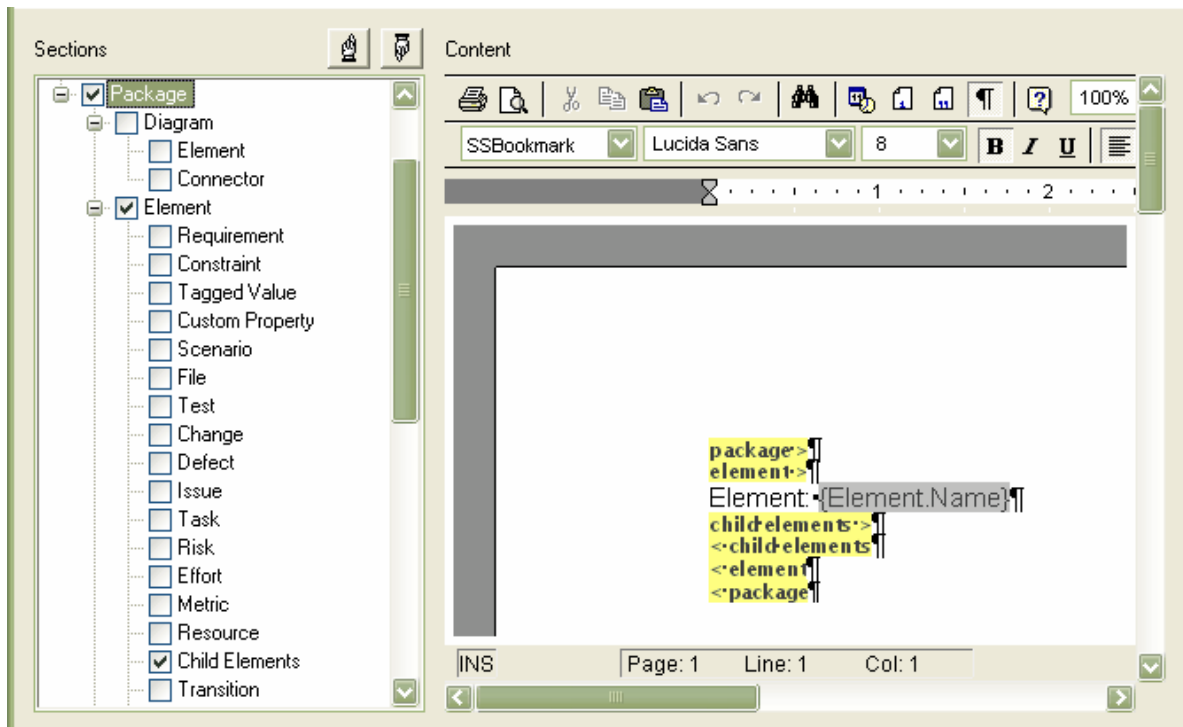
Example: Rendered Subsection

This first example shows a template with content between the child element tags. In this example child elements of the parent are rendered using the *child elements* section because it contains valid content and fields:



Example: Non-rendered Subsection

This second example shows a template with no content between the child element tags. In this example child elements of the parent are rendered using the *element* section because the *child elements* section is empty. The *child elements* section is used as a placeholder:



Child Document Sections and their corresponding parent sections

Child Section

- Package->Child Package
- Package->Element->Child Element
- Package->Element->Diagram
- Package->Diagram->Element
- Package->Diagram->Connector

Section Rendered when used as a placeholder

- Package
- Package->Element
- Package->Diagram
- Package->Element
- Package->Element->Connector

14.1.3.6 RTF Template Editor Commands

The following topics provide assistance on using the RTF Template Editor.

- [Scrolling through text](#)
- [File and Printing Options](#)
- [Line Editing](#)
- [Block Editing](#)
- [Clipboard](#)
- [Image and Object Imports](#)
- [Character Formatting](#)
- [Paragraph Formatting](#)
- [Tab Support](#)
- [Page Breaks and Repagination](#)
- [Headers, Footers and Bookmarks](#)
- [Table Commands](#)
- [Sections and Columns](#)
- [Stylesheets and Table of Contents](#)
- [Text/Picture Frame and Drawing Objects](#)
- [View Options](#)
- [Navigation Commands](#)
- [Search/Replace Commands](#)
- [Highlighting Commands](#)

14.1.3.6.1 Scrolling through text

Control

Keyboard:

Function

- Use *Up*, *Down*, *Left* and *Right arrow keys* to scroll up or down a line, or left or right one character.
- The *Home* key is used to position at the beginning of the current line.
- The *End* key is used to position at the end of the current line.
- *Ctrl-PgUp* is used to position at the beginning of a file.
- *Ctrl-PgDn* is used to position at the end of a file.
- *Page Up* is used to display the previous page.
- *Page Down* is used to display the next page.
- *Ctrl + Left arrow key* is used to position on the next word.
- *Ctrl + Right arrow key* is used to position on the previous word.
- *Ctrl + Up arrow key* is used to position at the first column of the current line (if not already on the first column) or at the first column of the previous line.
- *Ctrl + Down arrow key* is used to position at the first column of the next line.
- Press the *F10* key and type in a line number to jump to. This function is also available from the Navigation menu.

Mouse

You can click the mouse on the vertical and horizontal scroll bar to accomplish various scrolling function. These functions are available only if the horizontal or the vertical bar has been enabled by the startup parameters:

Vertical Scroll Bar: Click the mouse on the arrows on either end to scroll the screen up or down by one line.

Click the mouse above the elevator to scroll the screen up by one page. Similarly, click the mouse below the elevator to scroll the screen down by one page.

You may also drag the elevator to any position in the bar. As the elevator is dragged, the editor will scroll the screen up or down accordingly to maintain the correct cursor position.

Horizontal Scroll Bar: Click the mouse on the arrows on either end to scroll the screen left or right by one line. Click the mouse on either side of the elevator to scroll the screen left or right by 1/2 screen.

You may also drag the elevator to any position in the bar. As the elevator is dragged, the editor will scroll the screen left or right accordingly to maintain the correct cursor position.

14.1.3.6.2 File and Print Options

Control

Function

| | |
|---------------|--|
| New | <p>This command clears an existing template from the edit window and starts an empty, unnamed template.</p> <p><i>You will be prompted to save any modification to the previous template.</i></p> |
| Open File | <p>This function is used to clear any existing text from the edit window, and open a new document.</p> <p><i>You will be prompted to save any modification to the previous document.</i></p> |
| Save | <p>Use this selection to save the text to the current file name.</p> <p><i>If a file is not yet specified, the editor will prompt you for a template name.</i></p> |
| Save As | <p>This selection is similar to Save File. In addition, it allows you to specify a new template name for saving the template.</p> <p>You can invoke this function by pressing Ctrl + Shift + S, or selecting the option from the menu.</p> |
| Exit | <p>Use this to close an editing session.</p> <p>You can exit by selecting the option from the menu.</p> <p><i>If the current template has been modified, you will be prompted to save the modifications.</i></p> |
| Import | <p>This function imports an existing RTF document into the Template Editor, so as to have model elements from that document inserted into the template.</p> <p><i>This is useful when creating templates from a predefined document with a particular "look and feel".</i></p> |
| Export | <p>This function saves the template as a RTF document rather than as a template.</p> <p><i>This can be useful when the user wishes to save the template for other models.</i></p> |
| Page Layout | <p>Use this option before selecting the Print option to specify the page layout. You can specify margin (left, right, top and bottom) in inches.</p> <p>You can invoke this function by hitting the Ctrl F4 function keys together (or select the option from the menu).</p> |
| Printer Setup | <p>This option invokes a printer specific dialog box for the default printer (the default printer selection is made from the control panel of Windows) You select the parameters from a set of printer specific options. These options include page size, page orientation, resolution, fonts, etc.</p> <p>You can invoke this function by hitting the Ctrl + Shift + P key or select the option from the menu.</p> |
| Print | <p>Use this option to print the contents of the current file. You may also choose to print only the selected part of the file. To print a block of text, the desired text must be highlighted before invoking the print function. This command supports these highlighted blocks:</p> <ul style="list-style-type: none">Line BlockCharacter Block <p>The print function will print on a default printer selected from the Windows' control panel. You can alter the printer setup or Page Layout prior to invoking the print option. You can invoke the printing function by hitting the Ctrl + P key or select the option from the menu. The editor will display a dialog box where you can select the scope of the printing.</p> |

| | |
|---------------|---|
| Print Preview | This option is used to preview the document before printing. The editor displays up to 2 pages at a time. You can scroll to a different page by using the PgUp/PgDn or the scroll bar. By default the preview rectangle is sized to fit the current window. However, you can use the zoom option to enlarge or shrink the preview rectangle as you wish. |
| Close | Closes the template editor, closing the editor will prompt the user to save any unsaved information. |

14.1.3.6.3 Line Editing

| Control | Function |
|----------------------------|---|
| Insert After Current Line | In the text mode this function creates a blank line after the current line. Press the F9 function key to insert a new line directly after the current line. |
| Insert Before Current Line | In the text mode this function creates a blank line before the current line. Press Ctrl + F5 together to insert a new line directly before the current line. |
| Delete Line | Use this function to delete the current line. The remaining lines will be scrolled up by one line. Press Shift + F9 together to delete the current line. |
| Join Lines | In the text mode this function joins the next line at the end of the current line. Hit the Alt J keys together to invoke this function. |
| Split Line | In text mode this function splits one line into two. at the current cursor position. Press Alt + S together to split the line at the current cursor position. Text immediately following the cursor will appear on the next line. |

14.1.3.6.4 Block Editing

| Control | Function |
|-------------------|--|
| Copy a Line Block | This command quickly duplicates a selected block of text from one location to another. Highlight the lines of text to be copied, then move the caret to the target location and press Alt + C , or select the option from the menu. The original text is not deleted. <i>This command provides a quick alternative to using the clipboard copy/paste functions.</i> |
| Move a Line Block | This command moves a highlighted block of text from one location to another Highlight a block of text to be moved, move the caret to the target location and press Alt + M , or select the option from the menu. The original text is deleted. <i>This is a quick alternative to using clipboard cut/paste functions.</i> |

| | |
|--------------------|--|
| Undo Previous Edit | <p>You can use this command to undo your last edit. Invoke this function by pressing Shift + F8, or select the option from the menu. A dialog box will appear, containing the information about the command to be undone. The box displays the <i>line number</i>, <i>column position</i>, <i>type of undo</i> (delete/insert/edit) and the contents of the undo buffer.</p> <p>You may modify the target line number or column position. Confirm the operation by clicking on the OK button.</p> <p><i>This undo feature is not available for column block edits, block move and replace string commands</i></p> |
| Redo Previous Undo | <p>This command reverses an <i>undo</i>. It can only be performed after an undo command.</p> |

14.1.3.6.5 Clipboard

| Control | Function |
|----------------------------|--|
| Cut/Copy Text To Clipboard | <p>Use this command to cut or copy a highlighted block of text to the clipboard.</p> <p>Highlight a block of text to be copied to the clipboard and press Ctrl + X (cut) or Ctrl + C (copy), or select the option from the <i>Edit</i> menu.</p> <p><i>This function also copies the associated formatting information using the RTF format and the native Template Editor format.</i></p> |
| Paste Text From Clipboard | <p>Use this command to paste the contents of the clipboard at the current caret location.</p> <p>Invoke this function by pressing Ctrl + V, or selecting the option from the <i>Edit</i> menu.</p> <p><i>Formatting information, if available, is also copied.</i></p> |

Paste Special Objects

This function displays the clipboard data in a number of available formats:

Native Object Format

If available, this is the first format in the list box. The data in this format can be later edited using the original application by double clicking the object.

This data can be embedded into your application by using the *Paste* option, or you can create a link to the original file by using the *Paste Link* option.

Formatted Text

This is one of the text formats. This option offers the most suitable format if the data is pasted from another text output application, as the font and formatting attributes are reproduced accurately.

Unformatted Text

This is another text format. This option pastes the text without retaining the formatting information.

Picture Format

The data is available in the Picture format. This object can be later edited (by double clicking the object) using the Microsoft MS Draw application.

Note: *This format is preferred over the Bitmap and the Device Independent Bitmap formats.*

Device Independent and regular bitmap formats

The data is available in the bitmap formats. The object can be later edited (by double clicking the object) using the Microsoft MS Draw application.

The editor converts these formats into the Picture format before calling the drawing application.

14.1.3.6.6 Image and Object Imports

| Control | Function |
|---------------|--|
| Embed Picture | <p>Use this command to embed a picture bitmap or Windows metafile from an external disk file at the current caret location.</p> <p>The embedded picture is saved within the document.</p> |
| Link Picture | <p>Use this command to <i>link</i> a picture bitmap or Windows metafile to the document. The linked picture appears at the current caret location.</p> <p>Linked picture data <i>is not</i> saved with the document, only its filename is stored within the document.</p> |
| Edit Picture | <p>Use this command to change the width and height of a picture located at the current caret position. The width and height is specified in inches.</p> <p><i>This function also allows you to align (top, bottom, or middle) the picture relative to the base line of the text.</i></p> |

| | |
|--------------------|--|
| Insert Object | <p>This function is used to embed objects into the text. The list box shows the applications that are available to create the object.</p> <p>When you select an application, the editor will launch that application. You can create the desired object using this application. When you save the application, the editor inserts an icon for the application. This icon indicates the inserted object. You can later edit the object by double clicking at it.</p> <p><i>Note that you can also use the Paste Special function to import the OLE objects, provided that the object is available in the clipboard.</i></p> |
| Drag/Drop Function | <p>This is a method of inserting a file object into the text directly. To insert a file, open the Windows File Manager and locate the file to be inserted. Now click on the icon of the file and keep the mouse button depressed as you drag the mouse cursor to the editor window. Release the mouse button at the location where the object should be inserted. The editor shows an icon to indicate the inserted object.</p> <p>You can edit this object by double clicked at the icon. An object inserted using this method makes use of Microsoft's Packager application to tie the file with the application that originally created it.</p> <p><i>Note that a documented problem with the original Packager application may create errors during this function. Install the corrected version of the PACKAGER.EXE program for proper functioning.</i></p> |
| Background Picture | <p>This option, available from the <i>Other</i> menu, is used to set a background picture for the text. The background picture occupies the entire text area.</p> <p>The picture file can be a Windows' bitmap (.BMP) or Metafile (.WMF).</p> |

14.1.3.6.7 Character Formatting

| Control | Function |
|------------------|--|
| Character Styles | <p>The following character style commands are available:</p> <p>Command/ Keystroke Normal/Alt 0 Bold Formatting/Alt 1 Underlining/Alt 2 Italic/Alt 3 Superscript/Alt 4 Subscript/Alt 5 Strike/Alt 6</p> <p>Character style options allows you to apply one or more style formats to the current character, or to all characters in a highlighted block of text.</p> <p>To apply a format to the current character, hit the appropriate keystroke (<i>Alt + 1</i> through <i>Alt + 6</i>), or select the option from the menu. To apply this format on a block of characters, highlight a block using the Line Block or Character Block options. Now, hit the applicable keystroke, or select the option from the menu.</p> <p>When you type on the keyboard, new characters will automatically assume all the formatting characteristics of the preceding character.</p> <p>The template editor allows multiple formats for a character. To apply more than one format, repeat the procedure described in the previous paragraphs.</p> <p>To reset all character formats, highlight the characters and select the 'Normal' option from the menu, or hit the <i>Alt + 0</i> keystroke.</p> |

| | |
|----------------|---|
| Fonts | <p>Use this option to change the font typeface and point size of the current character, or a highlighted block of text.</p> <p>If you wish to change the font for a highlighted block of text, select the block using the Line or Character highlight function. If you wish to change the font of a single character, simply position the cursor on that character. Now select the <i>Font</i> option from the menu, or press <i>Alt + F10</i> keys together. A dialog box will appear that shows the list of typefaces and point sizes to select from. Make the desired selection now.</p> |
| Colors | <p>Use this selection to change the text color of the current character, or a highlighted block of text.</p> <p>If you wish to change the color of a highlighted block of text, select the block using the Line or Character highlight function. If you wish to change the color of a single character, simply position the cursor on that character. Now select the color option from the menu. A dialog box will appear that shows the color selections. Make the desired selection now.</p> |
| Hidden Text | <p>Text formatted with this attribute is treated as hidden text. Normally the hidden text, as the name implies, does not appear on the screen or printer. However you can display hidden text by selecting the <i>Show Hidden Text</i> option from the <i>View</i> menu.</p> |
| Protected Text | <p>The text formatted with this attribute are protected from the editing changes. The protected text appears with a light shade in the window. This function is available only when the <i>protection lock</i> is turned off. The <i>protection lock</i> can be turned off by using an option from the <i>Other</i> menu.</p> |

14.1.3.6.8 Paragraph Formatting

| Control | Function |
|-------------------------------|--|
| Reset Paragraph Format | <p>Resets all paragraph formats for the current paragraph, or for a highlighted block of text.</p> <p>To reset the paragraph formats for the current paragraph, press the <i>Alt + P</i> keys together, or select the option from the menu. To reset the formats for a block of lines, highlight a block and press <i>Alt + P</i>, or select the option from the menu.</p> |
| Paragraph Centering | <p>Centers all lines in the current paragraph or all lines in a highlighted block of text.</p> <p>To center the current paragraph, press <i>Alt + 8</i> keys together, or select the option from the menu. To center a block of lines, highlight a block of text and press <i>Alt + 8</i>, or select the option from the menu.</p> |
| Paragraph Right Justification | <p>Right justifies all lines in the current paragraph, or all lines in a highlighted block of text.</p> <p>To right justify the current paragraph, press <i>Alt + 9</i> keys together, or select the option from the menu. To right justify a block of lines, highlight a block of text and press <i>Alt + 9</i>, or select the option from the menu.</p> |
| Paragraph Justification | <p>Block-justifies text on both left and right margins.</p> <p>To justify the current paragraph, select the option from the paragraph menu. To justify a block of lines, highlight a block of text and then select this option from the menu.</p> |

| | |
|-------------------------------|--|
| Paragraph Double Spacing | <p>Double space all lines in the current paragraph or all lines in a highlighted block of text. A double spaced paragraph has a blank line of between each text line.</p> <p>To double space the current paragraph, press Alt + O, or select the option from the menu. To double space a block of lines, highlight a block of text and hit the Alt + O, or select the option from the menu.</p> |
| Paragraph Indentation (Left) | <p>Use this selection to create a left indentation for all lines in the current paragraph, or for all lines in a selected block of text. The successive use of this option increases the amount of left indentation.</p> <p>To apply the left indentation to the current paragraph, press the Alt + L, or select the option from the menu. To apply the left indentation to a block of lines, highlight a block of text and press Alt + L, or select the option from the menu.</p> <p>To create the left indentation using the mouse, click the left mouse button on the indentation symbol on the lower left end of the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button. The indentation created using this method is applicable to every line in the paragraph <i>except</i> the first line.</p> |
| Paragraph Indentation (Right) | <p>Use this selection to create a right indentation for all lines in the current paragraph or for all lines in a highlighted block of text. The successive use of this option increases the amount of right indentation.</p> <p>To apply the right indentation to the current paragraph, press Alt + R, or select the option from the menu. To apply the right indentation to a block of lines, highlight a block of text and press Alt + R or select the option from the menu.</p> <p>To create the right indentation using the mouse, click the left mouse button on the indentation symbol on the lower right end of the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button.</p> |
| Paragraph Hanging Indentation | <p>This option is similar to paragraph left indentation, except that the indentation is <i>not</i> applied to the first line of the paragraph.</p> <p>To apply the hanging indentation to the current paragraph, press Alt + T, or select the option from the menu. To apply the left indentation to a block of lines, highlight a block of text and press Alt + T, or select the option from the menu.</p> <p>To create the hanging indentation using the mouse, click the left mouse button on the indentation symbol on the upper left end of the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button.</p> |
| Paragraph Keep Together | <p>When this attribute is turned on for a paragraph, the editor attempts to keep all lines within the paragraph on the same page.</p> |
| Paragraph Keep with Next | <p>When this attribute is turned on for a paragraph, the editor attempts to keep the last line of the current paragraph and the first line of the next paragraph on the same page.</p> |
| Widow/Orphan Control | <p>When this attribute is turned on for a paragraph, the editor attempts to avoid widow/orphan paragraphs. An 'orphan' paragraph results when the last line of the paragraph lies on the next page. A 'widow' paragraph results when the first line of the paragraph lies on the previous page.</p> |

14.1.3.6.9 Tab Support

The RTF Template Editor supports *left*, *right*, *center*, and *decimal* tab stops. The tab stops are very useful for creating columns and tables. A paragraph can have as many as 20 tab positions.

The *left* tab stop begins the text following a tab character at the next tab position. To create a left tab stop, click the left mouse button at the specified location on the ruler. The left tab stop is indicated on the ruler by an arrow with a tail toward the right.

The *right* tab stop aligns the text at the current tab stop such that the text ends at the tab marker. To create a right tab stop, click the right mouse button at the specified location on the ruler. The right tab stop is indicated on the ruler by an arrow with a tail toward the left.

The *center* tab stop centers the text at the current tab position. To create a center tab stop, hold the shift key and click the left mouse button at the specified location on the ruler. The center tab stop is indicated on the ruler by a straight arrow.

The *decimal* tab stop aligns the text at the decimal point. To create a decimal tab stop, hold the shift key and click the right mouse button at the specified location on the ruler. The decimal tab stop is indicated on the ruler by a dot under a straight arrow.

Tab stops can also be created by using the *Set Tab* selection from the *Paragraph* menu. This allows you to specify the tab position, tab type (left, right, center, or decimal) and tab leader (dot, hyphen, underline, or none).

To move a tab position using the mouse, simply click the left mouse button on the tab symbol on the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button.

To clear a tab position, simply click at the desired tab marker, or select the option from the menu. You can also clear all tab stops for the selected text by selecting *Clear All Tabs* option from the menu.

The *Snap To Grid* option in the *Other* menu affects the movement of the tabs (and the paragraph indentation markers) on the ruler. When this option is checked, the movements of these markers are locked on to an invisible grid at an interval of 1/16 inch.

A tab command is generally applicable to every line of the current paragraph. However, if you highlight a block of text before initiating a tab command, the tab command is then applicable only to the lines in the highlighted block of text.

14.1.3.6.10 Page Breaks and Repagination

A forced page break can be inserted into the document by pressing the *Ctrl + Enter* keys together, or selecting from the *Edit* menu *Break->Section Break*. A page break places the text after the page break on the following page. A forced page break is indicated by a solid line in the editing window.

In the *Print View* editing mode, the editor also creates automatic page breaks when the text overflows a page. An automatic page break is indicated by a dotted line in the editing window. As the name implies, these page breaks are calculated automatically by the editor between the keystrokes. The repagination process is time consuming, and sometimes there may not be enough time for a large document to complete the repagination between the edits. Therefore, the menu also provides an option to initiate complete repagination on demand.

Control

Inserting Page Number

Function

The *Page Number* selection from the *Insert* menu allows you to insert the page number into the document. The page number string is inserted at the current cursor position. This string is displayed using a gray color.

| | |
|----------------------|--|
| Inserting Page Count | The <i>Page Count</i> selection from the <i>Insert</i> menu allows you to insert the total number of pages into the document. The page count string is inserted at the current cursor position. This string is displayed using a gray color. |
| Show Page Border | When option is turned on, the editor displays the borders around the text on the screen. This option is available in <i>Page</i> mode only. The <i>FittedView</i> option must be turned off. |

14.1.3.6.11 Header, Footers and Bookmarks

The page header/footer functionality is available in the *Page* Mode only.

| Control | Function |
|-----------------------------|---|
| Show Page Header/ Footer | Normally, the editor does not show the header and footer for a page. Check this option in the <i>View</i> menu to display the page header and footer. This option <i>does not</i> allow you to edit the content of the header/footer. Every section in a document can have its own page header and footer. If a section does not have a page header/footer of its own, this option shows the header/footer from the preceding section. |
| Edit Page Header/ Footer | Enables editing of the text for the page header and footer. This option is available from the <i>Edit</i> menu. |
| Insert Footnote | Inserts a footnote at the current cursor location. The footnote text is displayed at the bottom of the page. |
| Edit Footnote Text | This option displays the footnote text in-line with the regular text. It allows you to edit the footnote text. The modified footnote is displayed at the bottom of the page. |
| Insert Bookmark | This dialog box is activated from the <i>Insert</i> menu. It allows you to place a bookmark (new or existing) at the current text location. You can also position the cursor at a specified bookmark, and delete existing bookmarks. |

14.1.3.6.12 Table Commands

The table menu is available in the *Page* or *Print View* modes only (see Editing Modes). This menu contains commands for the creation of a new table, or to edit an existing table's attributes.

| Control | Function |
|------------------------------|---|
| Insert Table | Inserts a new table in the document. This option prompts the user for the initial number of rows and columns in the table. The editor initially creates cells of equal width. The user can, however, change the cell width by dragging the cell borders using the mouse. In <i>Page Mode</i> , the table cells are arranged by rows. In <i>Print View Mode</i> , the table structure is not visible. |
| Insert Row | Inserts a new row before the current table row. The new table row has the same number of columns as the current row. |
| Insert Column Merge Cells | Inserts a new column to the left of the current column. Merges together highlighted cells. The width of the resulting cells is equal to the sum of all merged cells. If the highlighted cells span more than one table row, this operation creates multiple merged cells, each within its row. |
| Split Cell | Splits the current table cell into two cells of equal width. The entire text of the original cell is assigned to the first cell. The second cell is created empty. |

| | |
|---|--|
| Delete Cells | <p>Deletes selected cells from a table. A dialog box allows the user to select the cells for the deletion.</p> <p>The dialog box has three options: <i>cells</i>, <i>columns</i>, and <i>rows</i>. <i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> <p>A table is automatically deleted when all its cells are deleted.</p> |
| Row Position | <p>Use this option to position the table or selected table rows. A dialog box lets you position the table as left justified, centered, or right justified.</p> |
| Row Height Header Row Keep Row Together Row Text Flow Cell Width | <p>Use this option to sets the width of the cells.</p> <p><i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> |
| Cell Border Width | <p>Use this option to sets the border width of the cells.</p> <p><i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> |
| Cell Border Color | <p>Use this option to sets the border color of a cell.</p> <p><i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> |
| Cell Shading | <p>Use this option to shade the selected cells. A dialog box allows the user to select the cells for this operation.</p> <p>The dialog box has three options: <i>cells</i>, <i>columns</i>, and <i>rows</i>. <i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> <p>The shading is specified in terms of a shading percentage. A value of 0 indicates a white background, where as the value of 100 indicates a black background.</p> |
| Cell Color | <p>Use this option to sets the background color of a cell.</p> <p><i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> |
| Cell Vertical Align Cell Rotate Text | |

| | |
|-----------------------|---|
| Cell Border | <p>Use this option to create borders around selected cells. A dialog box allows the user to select the cells for this operation.</p> <p>The dialog box has three options: <i>cells</i>, <i>columns</i>, and <i>rows</i>. <i>Cells</i> selects the current cell or all the cells in a highlighted block of text. <i>Columns</i> selects all the cells in the current column, or the cells of all columns in the highlighted block of text. <i>Rows</i> selects all the cells in the current row, or the cells of all rows in the highlighted block of text.</p> <p>The user can specify the width of each border (top, bottom, left and right). The border width should be less than the cell text margin. The cell text margin is the distance from the left edge of the cell to the beginning of the text in the cell. The border width is specified in twips (1440 twips equal to one inch).</p> |
| Select Current Column | Selects the whole column at the current cursor location. |
| Show Grid | Enables or disables the display of table grid lines. The table grid lines are for display purpose only and will not appear on the printed document. |

14.1.3.6.13 Sections and Columns

The editor allows you to divide a document into multiple sections. A multiple section document is useful when:

- a) you need to vary the page margins from one page to another
- b) you need to create multiple columns of text.

| Control | Function |
|--------------------------------|--|
| Creating a New Section | To create a new section, select the <i>Break</i> submenu option from the <i>Edit</i> menu. A section break line (double solid line) is created before the current line. The new section begins at the text following the break line. |
| Editing the Section Parameters | <p>The following section parameters can be edited:</p> <ul style="list-style-type: none"> Number of columns and column spacing. Portrait or Landscape orientation. Placement of the text on the next page. Page Margins <p>The first three parameters can be edited by selecting <i>Section Edit</i> from the <i>Edit</i> menu. The last parameter can be edited by selecting <i>Page Setup</i> from the <i>File</i> Menu.</p> |
| Deleting a section break line | To delete a section break line, simply position the cursor on the section break line and hit the key. |

Multiple Column Editing *This option is available in the Print View and Page Modes only (See Editing Modes)*

To create multiple columns for a section select **Section Edit** from the **Edit** menu and specify the number of columns to create. You can also specify the space between the columns.

Text in a multiple column section wraps at the end of the column. When the text reaches the end of the page, or the end of a section, new text is placed in the next column.

In the *Print View* mode, the multiple columns are not actually seen in the window. In the *Page Mode*, columns are visible as they would be when the text is printed. Therefore, the *Page Mode* is useful when editing multiple column text.

Column Break

Normally in a multiple column section, the text flows to the next column at the end of the current column. The column break option can be used to force the text to the next column before the current column is completely filled.

A column break can be inserted by selecting the option from the menu (**Edit->Insert Break...**). A column break is indicated by a line with a 'dot and dash' pattern. The text after the column break line is placed on the next column. To delete the column break line, simply position the cursor on the line and hit the key.

14.1.3.6.14 Stylesheets and Table of Contents

The editor supports Character and Paragraph-type stylesheet style items. The Character stylesheet style constitutes a set of character formatting attributes and is applied to a character string. The Paragraph stylesheet style constitutes not only a set of character formatting attributes, but also a set of paragraph formatting attributes. The paragraph style is applied to one or more paragraphs.

Control

Create and edit styles

Function

A stylesheet style is created and modified using the **Edit Style** option from the **Edit** menu. This opens a dialog box which allows you to choose between a character style or a paragraph style.

You can select an existing style to modify from the list box or enter a name for a new style. Once you click the 'Ok' button, the recording of the stylesheet properties begins. You can use the ruler, toolbar, or the menu selections to modify the stylesheet items. The ruler, toolbar, and menu also reflect the currently selected properties for the stylesheet item. Please note that the paragraph properties are allowed only for the paragraph type of stylesheet item.

After you have selected the desired properties, terminate the stylesheet editing mode by either selecting the **Edit Style** selection from the menu again or by clicking anywhere in the document. If the existing stylesheet item was modified, the document automatically reflects the updated stylesheet properties. If a new stylesheet item was created, your next step is to apply the style to the desired text by choosing the **Style** option from the **Font** or **Paragraph** menu selection.

Apply character styles

The **Style** selection in the **Font** menu allows you to apply a stylesheet style to the currently highlighted character string.

Apply paragraph styles

The **Style** selection in the **Paragraph** menu allows you to apply a stylesheet style to the current paragraph. To apply a style to a range of paragraphs, highlight the paragraphs before selecting the menu option.

Table of Contents To insert a table of contents, first create the heading styles using the *Edit Style* option from the *Edit* menu. For example, if you wish to insert a three level deep table of contents, create heading styles 'heading 1', 'heading 2', and 'heading 3'. Then place the cursor at the heading lines and apply a suitable heading style using the *Style* menu selection from the *Paragraph* menu. The last step is to position the cursor where you wish to insert the table of contents and select the *Table of Contents* menu selection from the *Insert* menu.

The table-of-contents are automatically updated whenever repagination occurs.

14.1.3.6.15 Text/Picture Frame and Drawing Objects

A frame is a rectangular area on the page. A frame can contain both text and picture. The text outside the frame flows around the frame.

A drawing object can be a text box, rectangle or a line. The drawing object overlays on top of the text. To embed a Frame or Drawing object, select the appropriate option from the *Insert* menu. The new object will be inserted at the current text position.

To insert text into the frame or a text box, click a mouse button inside the frame to select the frame. Now type the text at the cursor position.

To size a frame, click a mouse button inside the frame to select the frame. Now click the left mouse button on a sizing tab and move the mouse while the mouse button is depressed. Release the mouse when done. The text inside the frame is automatically wrapped to adjust to the new width. If the new height of the frame is not enough to contain all text lines, the frame height is automatically adjusted to include all lines. If the frame contains only a picture, the picture size is automatically adjusted to fill the frame.

To move the frame, click a mouse button inside the frame to select the frame. Now move the mouse cursor just outside the frame until a plus shaped cursor appears. Click the left mouse button. While the mouse button is depressed, move the frame to the new location and release the mouse button.

To edit the base vertical position of the frame, select the 'Vertical Frame Base...' option from the *edit | frame* menu.

Frames locked to the top of the page or the top of the margin retain their vertical position when text is inserted before them. To edit the border and the background of a drawing object, select the 'Edit Drawing Object' option from the *edit | frame* menu. This option is available in the Page Mode only.

14.1.3.6.16 View Options

This menu allows you to turn on and off the following viewing options:

| Control | Function |
|----------------|--|
| Page Mode | In this mode, the editor displays one page at a time. This mode is available when the editor is called with both the <i>Word Wrap</i> and the <i>Page Mode</i> (or the <i>PageView</i> flag) flags turned on. This mode is most useful for the documents containing multiple columns, as the columns are displayed side by side. In addition, this mode provides all the features of the Print View mode. |
| Fitted View | A special case of the page mode, in which the text wraps to the window width and the soft page breaks are not displayed.. |

| | |
|---------------------|--|
| Ruler | The Ruler shows tab stops and paragraph indentation marks. The ruler can also be used to create or delete tab stops. |
| Tool Bar | The tool bar provides a convenient method of selecting fonts, point sizes, character styles and paragraph properties. The tool bar also shows the current selection for font, point size and character styles. |
| Show Status Ribbon | The status ribbon displays the current page number, line number, column number and row number. It also indicates the current insert/overtyping mode. |
| Show Hidden Text | This option displays the text formatted with the hidden attribute (see Character Formatting Options) with a dotted underline. When this option is turned off, the hidden text is not visible. |
| Show Paragraph Mark | This option displays a symbol (an inverted 'P') at the end of each paragraph. This option may be useful when working with lines with many different heights. |
| Hyperlink Cursor | This option is used to display the hyperlink cursor when the cursor is positioned on a hypertext phrase. The hyperlink cursor is an image of a hand with a finger pointing to the text. |
| Zoom | This feature allows you to shrink or enlarge the display of the document text. The editor allows a zoom percentage between 25 and 200. |

14.1.3.6.17 Navigation Commands

| Control | Function |
|----------------|--|
| Jump | Use this function to reposition to a desired line number. You can invoke this function by pressing the F10 key, or selecting the option from the menu. The editor will then display a dialog box so that you can enter the line number to jump to. See Scrolling Through the Text for other navigation functions. |

14.1.3.6.18 Search and Replace Commands

| Control | Function |
|----------------------|---|
| Search a Text String | Use this function to locate a string of characters in the current file. The editor will search for the first instance of the given character string. To find the subsequent instances of the same character string, use Search Forward or Search Backward commands. You can invoke this function by pressing the F5 key, or selecting the option from the menu. The editor will display a dialog box where you can enter the character string to locate. You can specify the search to be in a backward or forward direction from the current cursor position, or you can specify the search to take place from the beginning of the file. <i>You can also force a non-case-sensitive search, in which case the string is matched irrespective of the case of the letters in the string.</i> |

| | |
|-----------------------|--|
| Search Forward | <p>Use this function to locate the next instance of a previously located string using the Search Function. If the Search Function is not yet invoked, this function will call the Search Function instead.</p> <p>You can invoke this function by pressing the <i>Control</i> + <i>F</i> Keys together (or select the option from the menu).</p> |
| Search Backward | <p>Use this function to locate the previous instance of a previously located string using the Search Function. If the Search Function is not yet invoked, this function will call the Search Function instead.</p> <p>You can invoke this function by pressing the <i>Control</i> + <i>Shift</i> + <i>F</i> Keys together (or select the option from the menu).</p> |
| Replace a Text String | <p>Use this function to replace a character string with another character string.</p> <p>You can invoke this function by pressing the <i>F6</i> key, or selecting the option from the menu. The editor will show a dialog box where you will enter the old and new character strings.</p> <p>You may also choose to conduct the replace only within a selected part of the file. To choose such a block of text, the desired text must be highlighted before invoking the replace function.</p> <p><i>The dialog box also offers you an option to force the editor to verify each replace.</i></p> |

14.1.3.6.19 Highlighting Commands

| Control | Function |
|-----------------------------|---|
| Highlight a Character Block | <p>Use this function to highlight a block of characters.</p> <p>Mouse: Position the mouse cursor on the first character of the block and depress the left button. While the left button is depressed, drag the mouse to the last character of the block and release the mouse.</p> <p>Keystroke: Position the caret on the first character of the block and press the <i>Shift</i> key. While the key is pressed, use the position keys to move the caret to the last character of the block and release the <i>Shift</i> key. Normally, you can also use any position key in combination with the <i>Shift</i> key to create, expand, or shrink the text selection.</p> <p>Normally, a function that utilizes a character block also erases any highlighting. To explicitly erase the highlighting click a mouse button again or press any position key.</p> |

Highlight a Line Block

Use this function to highlight a block of lines.

Mouse: Position the mouse cursor at any position on the first line of the block and depress the right button. While the right button is depressed, drag the mouse to the last line of the block and release the mouse.

Keystroke: Position the caret at any position on the first line of the block and press **F8**. Use the **Up** and **Down arrow keys** to position the caret on the last line and press **F8** again.

Normally, a function that utilizes a line block also erases any highlighting. To explicitly erase the highlighting click a mouse button again or press the **F8** key again.

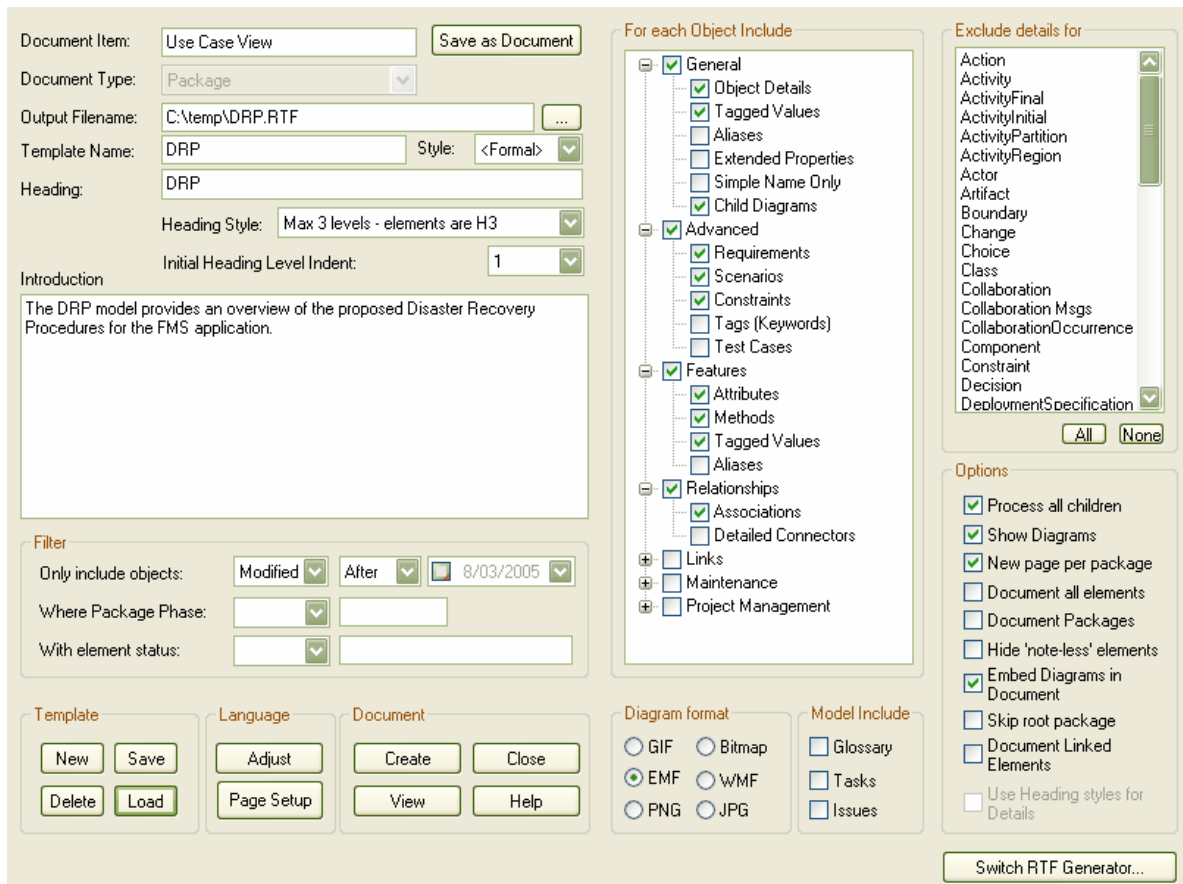
Highlight a Word

Double click any mouse button on the desired word to highlight the word.

14.1.4 The Legacy RTF Dialog

The following topics provide assistance on using the Legacy RTF Report dialog to document your project.

- [Create a Rich Text Document](#)
- [Document a Single Element](#)
- [The RTF Report Dialog](#)
- [Set the Main RTF Properties](#)
- [Apply a Filter](#)
- [Exclude Elements](#)
- [RTF Diagram Format](#)
- [Model Include](#)
- [RTF Options](#)
- [RTF Selections](#)
- [Generate the Report](#)
- [Diagram Only Report](#)
- [Report Templates](#)
- [Include or Exclude a Package from Report](#)
- [Save as Document](#)



14.1.4.1 Document a Single Element

RTF documentation may also be generated for a single element.

Select the element you wish to generate the documentation for, and then select *Documentation* from the *Element* menu. This will open the *Rich Text Format Report* dialog. See [The RTF Report Dialog](#) and related topics for further information.

14.1.4.2 The RTF Report Dialog

The RTF Report dialog allows you to set the exact contents and look and feel of your report. Enter the file name of the report, a heading, additional notes, template name (for saving the set-up) and other options. You may also select the style of the report - either plain or formal.

Optionally, set up a filter, the details to include, element types to exclude, whether to process child packages, whether to show diagrams and the diagram format. Users of the Professional and Corporate Editions of EA can switch to the Advanced Template editor by pressing the Switch RTF generator button.

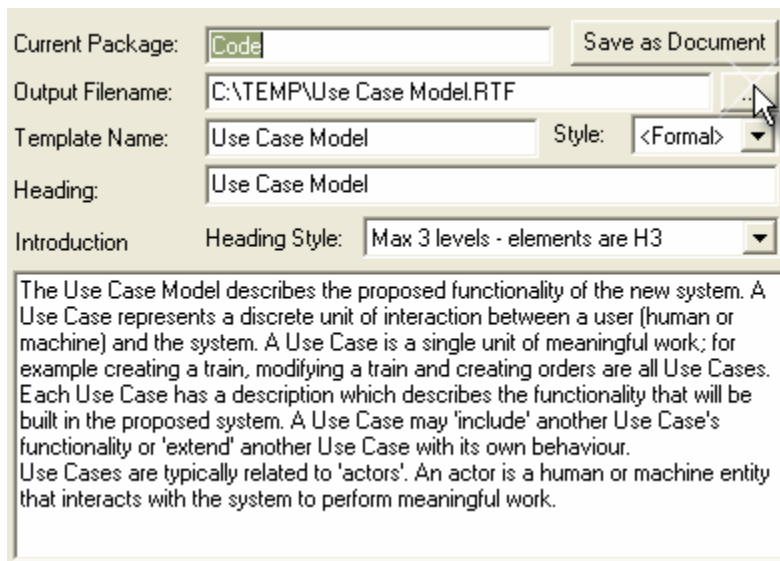
Note: The RTF dialog has a lot of options. Get to know them all to produce output at the level of detail suited to your project.

14.1.4.3 Set the Main RTF Properties

The main section of the RTF Report dialog allows you to set the output location and appearance of the final RTF document.

Setting Options for the RTF Document

1. Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The main section of this dialog is shown below.
2. Supply an **Output Filename** to save the report into - always include the extension .RTF as part of the filename.
3. Provide a **Template Name** if you wish to save this report set-up.
4. Select a report **Style** - Formal or Basic.
5. Enter a **Heading** for your report - appears as the first heading item in your output.
6. Select your desired **Heading Style** and **Initial Heading Level Indent** from the drop lists.



Note: It is recommended that you enter a full path name for your report. The images in your report are saved externally in an images directory, and supplying the full directory path avoids confusion over the location of these images. Note also that if you move your report, you must also move the images directory.

14.1.4.4 Apply a Filter

You may apply a filter on the RTF Report dialog to include or exclude elements by date modified, phase or status. This helps to track changes and break a document into multiple delivery phases.

Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The **Filter** section of this dialog is shown below.

- To enable the Date filter, check the box in the drop down list. Set the selection criteria as required (modified/created, before/after).
- The Phase filter applies at the Package level (not the element level) - and ignores the phase of the root package that you are documenting. If you enable the phase filter by selecting an operator from the phase operator combo, EA will filter out all packages that do not meet the selection criteria. All elements in that package are ignored - regardless of their individual phase.
- The Status filter lets you limit the output by element status. Unlike the phase filter, this filter applies to every element. You can filter where a status is like or not like a criterion, eg. "like proposed" or using the in and not in operators eg. in approved, validated. When using the in and not in operators, enter a comma separated list of status types as your criteria expression.

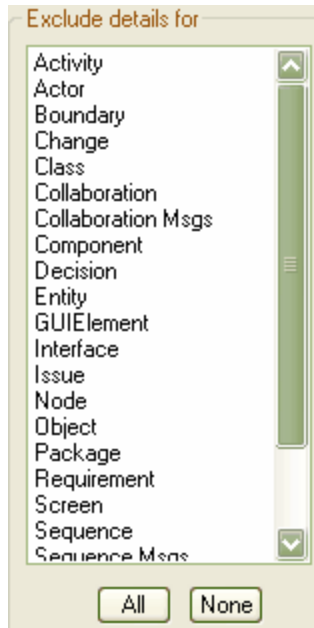


14.1.4.5 Exclude Elements

The RTF Report dialog allows you to exclude any element types you desire from your final output. This is useful when you wish to highlight particular items and not clutter up a report with too much detail.

Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The *Exclude details* section of this dialog is shown below.

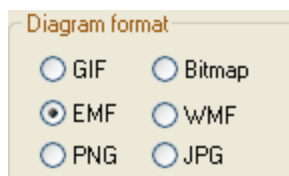
Left mouse click to select the elements you wish to exclude. Press *None* to clear your selections, or press *All* to select all elements.



14.1.4.6 RTF Diagram Format

Diagrams may be output to Bitmap files, GIF files or Windows Metafiles.

Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The *Diagram format* of this dialog is shown below. Select the format you prefer for your report



Bitmap files are raster images with a high level of detail but large size. They do not scale up or down very well. GIF files are raster images with reasonable detail and small size. They scale a little better than bitmaps.

Metafiles are vector images with high detail and small size (but often have compatibility problems with some printers or software). Metafiles scale very well.

PNG files are raster images with reasonable level of detail and smaller file sizes than GIF.

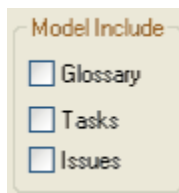
JPEG are lossy raster images with average levels of detail, JPEG does not work very well with line drawings and losses clarity when re sized, JPEG file sizes are typically very small.

Note: Generally metafiles are the best option, although it sometimes pays to experiment.

14.1.4.7 Model Include

The Model Include section of the RTF Report dialog has the following options, check the appropriate box to include the items in the RTF documentation:

- Check the *Glossary* item to include the [model glossary](#) in the generated RTF documentation.
- Check the *Tasks* item to include the [model tasks](#) in the generated RTF documentation.
- Check the *Issues* item to include the [model issues](#) in the generated RTF documentation.

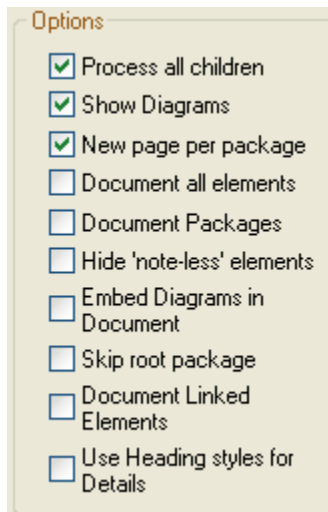


14.1.4.8 RTF Options

Additional *Options* you can select on the RTF Report dialog are shown below.

You can select whether to recursively document packages, to show diagrams or not and to add a page break before each new package or not.

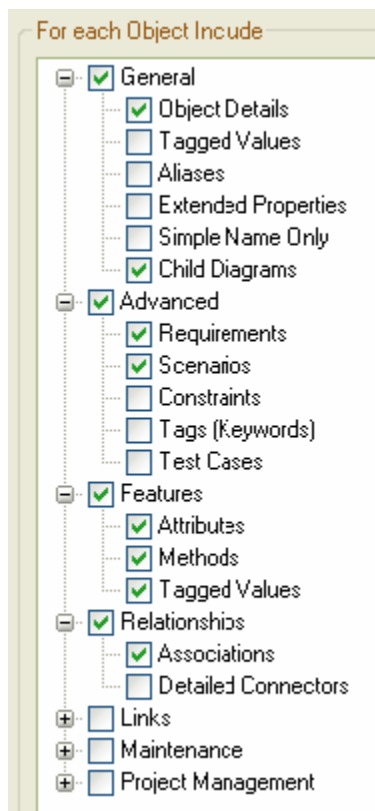
- Check the *Process all Children* item to recursively process all child packages within the main package.
- Check the *Show Diagrams* item to include diagrams in your document. Clear this item for no diagrams.
- Check the *New page per package* item to force a page break on each new package (excepting empty packages).
- Check the *Document all elements* item to include all elements included in the project.
- Check the *Document Packages* item to document the package as an element in addition to the documentation that would normally be produced for package documentation.
- Check the *Hide 'note-less' elements* item to exclude all elements without notes from the documentation.
- Check the *Embed Diagrams in Document* item to insure that the diagram images are contained within the RTF document rather than stored in a linked external file.
- Check the *Skip root package* item to exclude the parent package from the documentation and include only the child package/s.
- Check the *Document Linked Elements* item to include the object details for linked elements that do not originate from the selected package.
- Check the *Use Heading styles for Details* item to ensure that the details are formatted as Heading Styles rather than formatted text. This option is only available when the Heading Style in the [Main section](#) of the RTF Dialog is set to *Max 9 levels - elements are package + 1*.



14.1.4.9 RTF Selections

The *Objects to Include* section of the RTF Report dialog (shown below) allows you to select which documentation sections you want in your report. What you include or exclude will govern how simple or detailed your report is. You may wish to create multiple reports at different levels of detail for different audiences.

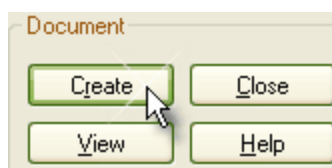
Experiment with these options to see what effect inclusion or exclusion has. Most items are self-explanatory and it is possible to expand the options by pressing the +symbol next to each category in the list. Checking an item in the list will select all of the options that are contained in the category, to exercise greater control over each category of options expand the top level item and then check the required individual items from the list. Sometimes an item will only apply to a certain type of element - eg. *Attributes* are only for classes and a few other element types. *Child diagrams* means show or hide any diagrams which are attached under a model element (eg. a Use Case may have a scenario diagram attached).



Note: Use this feature to produce the right level of detail for your audience. Technical readers may wish to see everything, while management may only wish the general outline.

14.1.4.10 Generate the Report

Once you have set up your document properties as required, you can generate the report. To do this, press **Create**. Once you have generated the document, press **View** to open the RTF in MS Word.



14.1.4.11 Legacy RTF Style Templates

The **RTF Style Editor** allows you to edit the RTF associated with various sections of the RTF Report facility in Enterprise Architect. You would typically use this functionality to customize a report look and feel for your company or client.

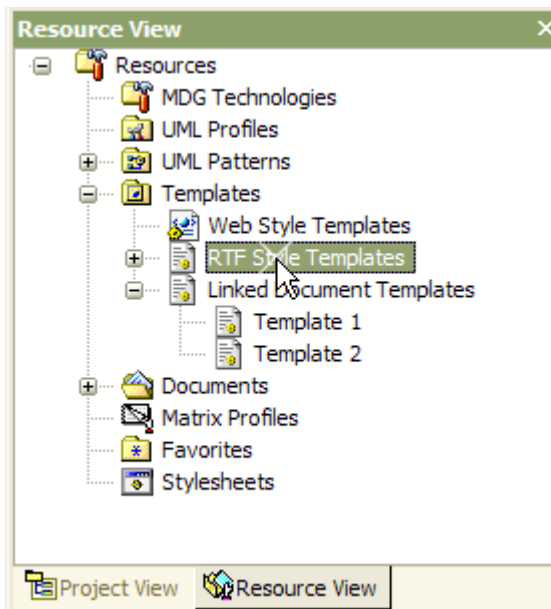
Create or Edit RTF Style Templates

1. In the *Resource View*, expand the *Templates* folder.
2. If you wish to edit an existing template, expand the *RTF Style Templates* tree, double click on the template name, or right click and select *Modify RTF Style Template* from the context menu. The *RTF Style Editor* will open. See below for further details.

-OR-

3. To create a new template, right click on *RTF Style Templates* and select *Create RTF Style Template* from the context menu.
4. Enter a name for the new template when prompted to do so. The *RTF Style Editor* will open. See below for further details.

Tip: To delete a template, right click on it and select *Delete RTF Style Template* from the context menu.

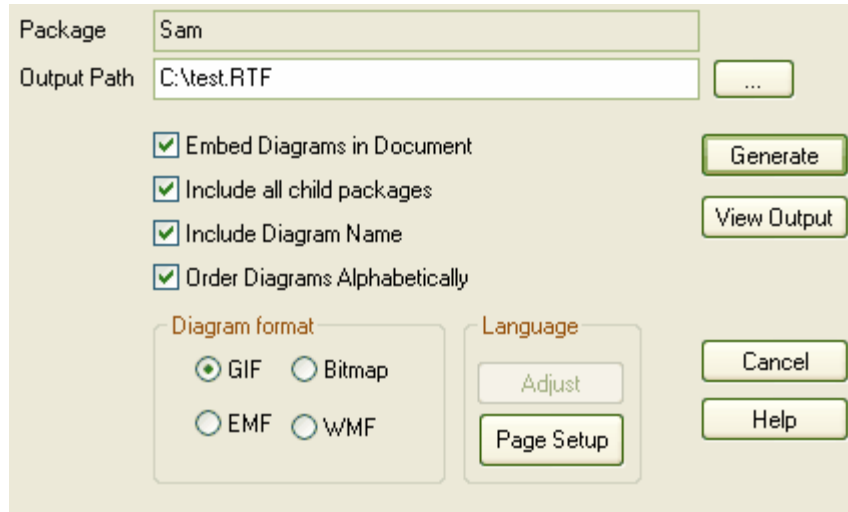


The *RTF Style Editor* (shown below) contains a list of all available RTF fragments for modification and customization.

Each fragment typically contains RTF plus one or more special tag names that EA will replace with information during generation. Currently you cannot alter the content within the tag names, but you can omit a complete tag by removing it, or alter its basic display properties in the surrounding RTF.

Special tag names are delimited by "#" characters - eg. #NOTES#

- From the *Documentation* submenu, select *Diagrams Only...* to open the *Export Diagrams to RTF Document* dialog.



- Enter the required information and press *Generate* to run the report.
- You can press *View* to show the RTF output once generated.

You can select following options to specify options to be included in the generated diagrams only report.

- Check the *Embed Diagrams in Document*, to ensure the diagrams are created within the RTF file not as linked image files.
- Check the *Include all child packages* to document all of the diagrams included in any child package.
- Check the *Include Diagram Name* item have the diagram name included within the generated documentation.
- Check the *Order Diagrams Alphabetically* item to have the documentation generated in alphabetical order.

14.1.4.13 Report Templates

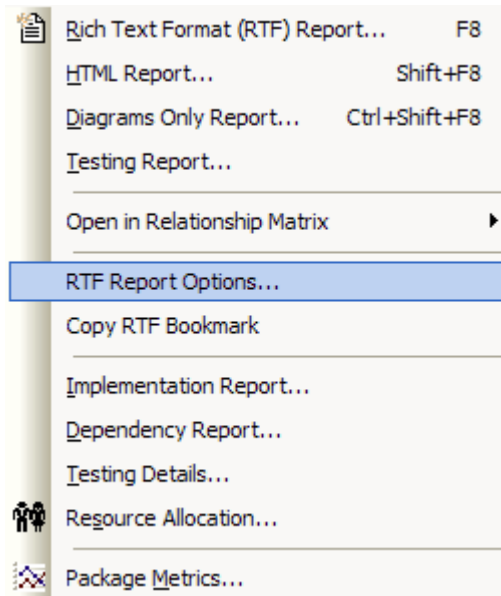
If you have previously defined and saved a template, select the *Load* option on the *RTF Report* dialog to open the list of defined templates. Select one in order to load it as the current template – all the features saved become the current features. This enables you to define a set of standard report types that streamline document production.

14.1.4.14 Include or Exclude a Package from Report

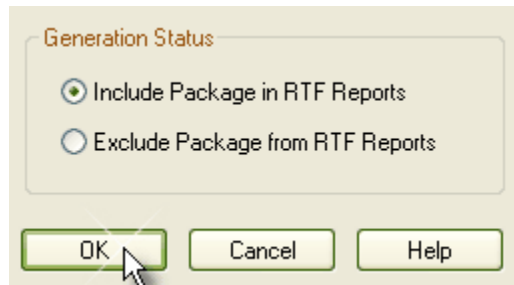
It is possible to exclude a particular package from any RTF reports by marking it for exclusion.

To Mark a Package for Exclusion

- In the Project Browser, right click on the package to open the context menu.
- From the *Documentation* submenu, select the *RTF Document Options* menu item.



3. In the *RTF Generation Options* dialog, select *Exclude Package from RTF reports*.
4. Press *OK* to save changes.



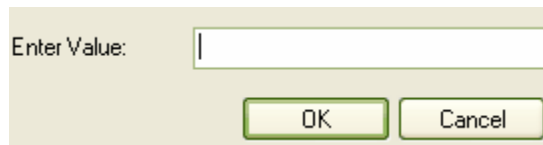
Note: By default, Packages are included in any RTF reports.

14.1.4.15 Save as Document

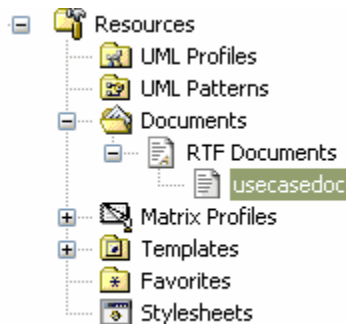
The Document feature allows the particular documentation configuration to be 'remembered', linking the loaded template within the RTF Report dialog to the current highlighted package. Perhaps a certain template is always used with a specific package, and there might exist multiple cases of documentation that need propagation; saving these as 'Documents' can ease document generation later.

To create and use Documents, follow these steps:

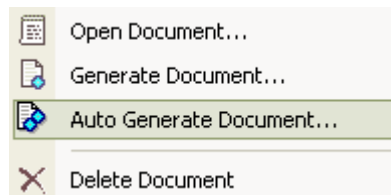
1. Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this).
2. Press *Resource Document* to open the following dialog:



- Supply a name for the document, and press **OK**. This document (the sample document was named 'usecasedoc') will be added to the resource view for easy future access.



- To generate documentation from the resource view, right-click the desired document to see the following menu.



- The available options are:
 - **Open Document**
Opens the corresponding .RTF file, as specified by the RTF template's 'Filename' property.
 - **Generate Document**
Opens the RTF Report dialog, loaded with the specified template.
 - **Auto Generate Document**
Documentation is generated, with the document located at the path specified by the template's 'Filename' property.
 - **Delete Document**
Remove specified document.

14.1.4.16 Custom Language Settings

If you export documents in RTF format from EA in languages other than English, you can customize the standard set of keywords that EA uses when generating RTF. This makes it much easier to generate documentation that is readily acceptable in your country or locale.

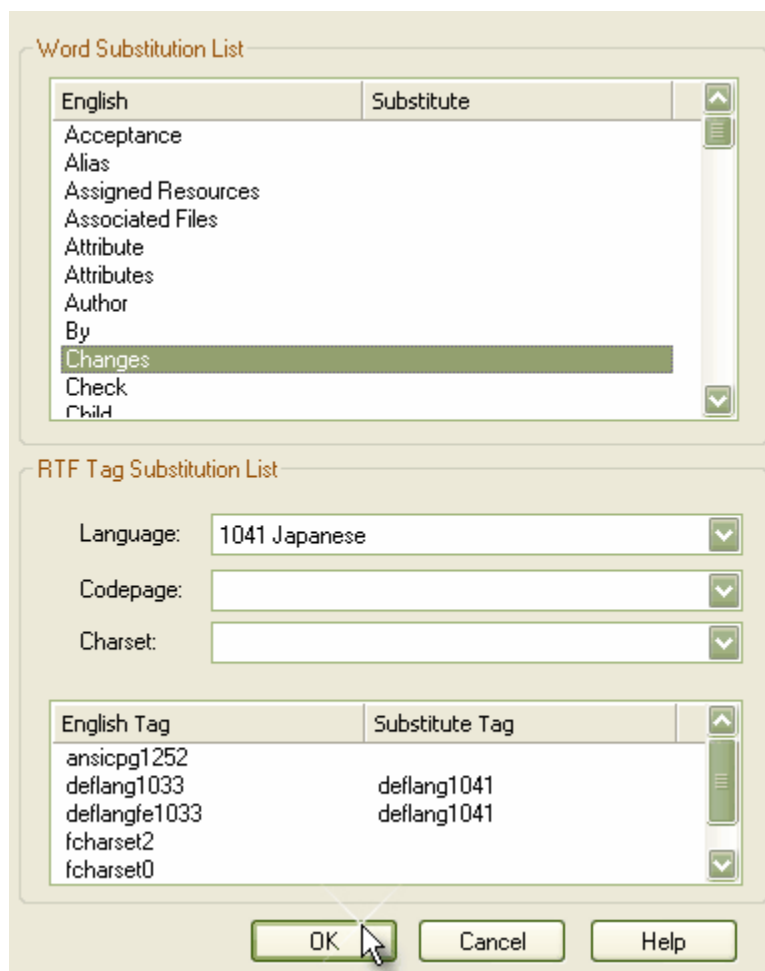
You can also customize the codepage, default language ID and character set to use.

To do this - you will need to set up a list of word substitutions. For instance, where EA would include the word "Figure", you can specify another word to replace it that is either in your language or more meaningful to your

readers.

To Set Up Substitutions

1. In the Project Browser, right click on a package to open the context menu.
2. Select *Rich Text Documentation* from the *Documentation* submenu.
3. In the *Rich Text Format Report* dialog, press *Adjust* (in the *Language* section). This opens the *Customize RTF Language* dialog (below)
4. Double click an item to set or clear its substitute word.
5. When you have finished, press *OK*.



To Set Up Codepage and Character Set

1. From the drop down lists, select the *Language*, *Codepage* and *Character set* that most closely matches your Locale.
2. If required, modify the *Substitute Tags* by double clicking on each and manually setting the value (for advanced use only).

3. To clear the substitution list, double click on each item in turn and delete the substitute value.
4. When you are happy with the settings, press **OK** to save.

Now when you generate RTF documents, the substitute tags will be used in the output.

Tip: Although intended for languages other than English, you can also tailor the look and feel of the RTF documents by substituting your keywords for the default ones used by EA.

14.1.5 Using MS Word

To further enhance and customize RTF documentation it is possible to create a custom master document, which can be used to add a table of contents, table of figures, headers and footers and to refresh linked files. In addition it is possible to create created with sustainable links to generated 'pieces' of EA output, pre-divided by EA using bookmarks.

See Also

- [Open a Report in Microsoft Word](#)
- [Changing Linked Images to Embedded Images](#)
- [Bookmarks](#)
- [Other Features of Word](#)

14.1.5.1 Open a Report in Microsoft Word

To open an RTF file in MS Word, simply load Word and open the file as a normal document - Word will convert it for you. If Word is the default handler of RTF files, then double click on the output file to load up and view the report.

*Tip: If you have Word configured to view RTF files, you can also press **View** on the RTF Report dialog.*

14.1.5.2 Changing Linked Images to Embedded Images

One of the options that is available when generating RTF documentation is the ability to store image files in a separate directory to the RTF document. If at a later stage it becomes desirable to embed the images into the RTF documentation, this is especially important when the document is to be distributed. If the images are stored in a separate directory recipients of document will see only the placeholder of images rather than the actual images.

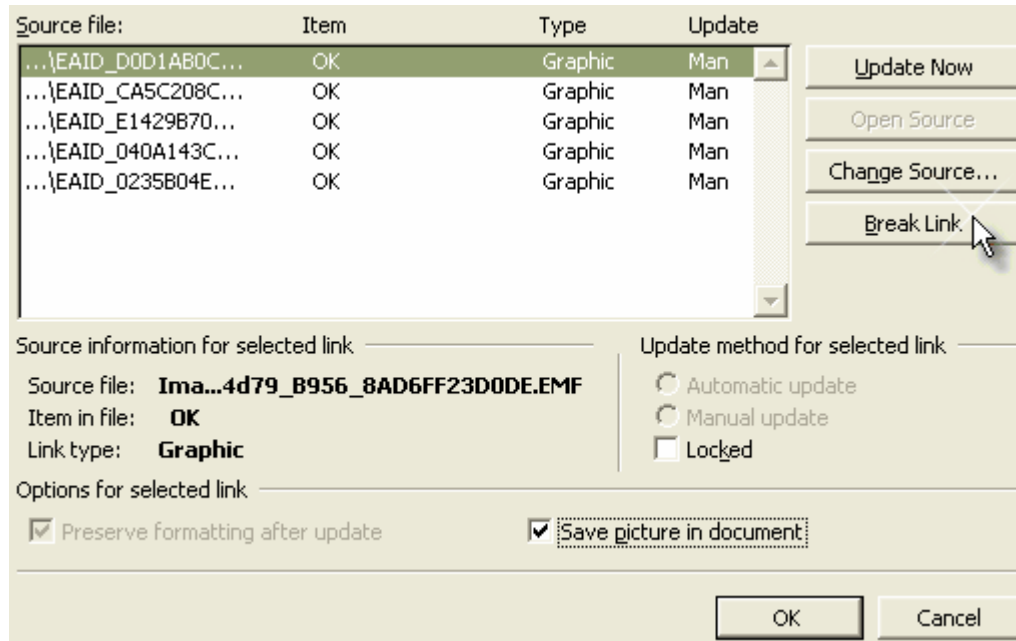
If you import an RTF document into Word with the images not embedded into the document, you have the option of breaking the links to the images and saving the image in the document.

To Break Image Links in Word

1. Open an RTF file in Word.
2. Select **Links** from the **Edit** menu.
3. Highlight all links in the **Links** list.
4. Check the **Save Picture in Document** check box.
5. Press **Break Link**.

- When prompted, press **Yes** to break links.

Word will break links and save copies of the images inside the document. You can distribute this document without the image directory.



14.1.5.3 Bookmarks

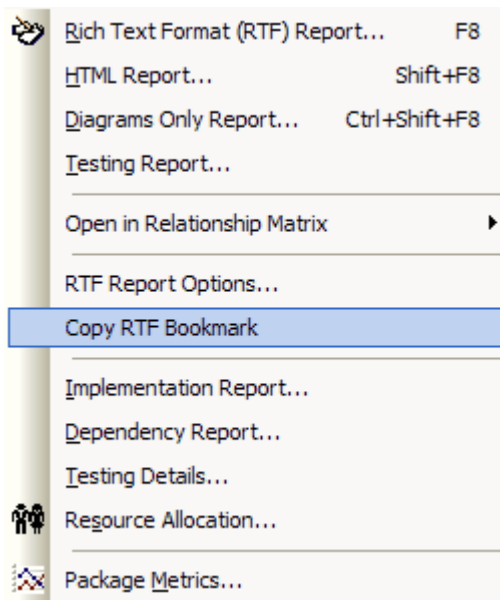
Bookmarks are markers that are automatically placed into your rich text document when it is generated. You can create a master document in Word, and link to sections of an Enterprise Architect report based on bookmarks. For an example, a Word document may have a section for a small part of your component model - using bookmarks you can generate a full component model, and then link into just one section of the report.

This way you can maintain a complex Word document from parts of EA reports. If you link into EA reports, then you can regenerate the report and refresh Word links to update the master document without having manually changed anything. More information relation to refreshing links can be found in the [Refresh Links](#) section

Bookmarks are created from package names. A bookmark applies from the beginning of a package to the end, and includes all child packages and elements underneath.

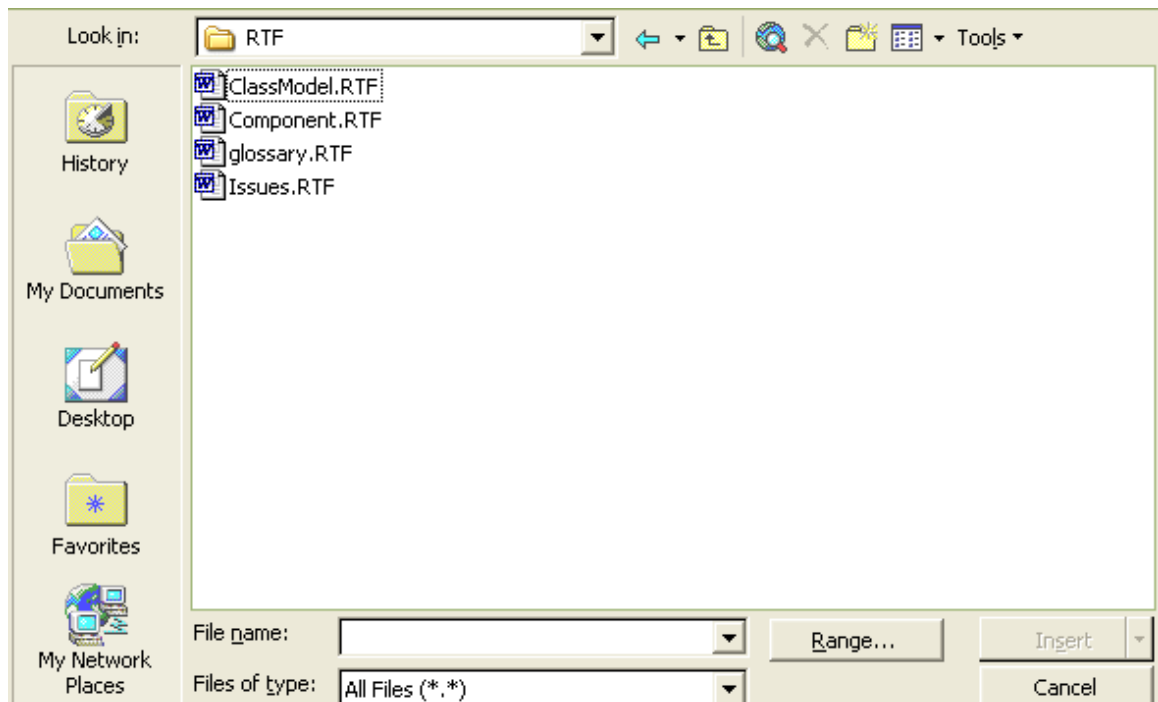
Bookmarking a Section of EA for RTF Documentation

- In EA select the Package in the Project Browser which you wish to include into the documentation and right click it to bring up its context menu.
- Mouse over the Documentation item and from the submenu select the Copy RTF Bookmark. This will paste your package into the clipboard to using in word.

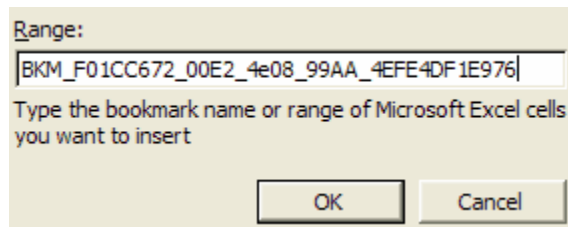


To Insert a Bookmarked Section of an EA RTF Document into Word

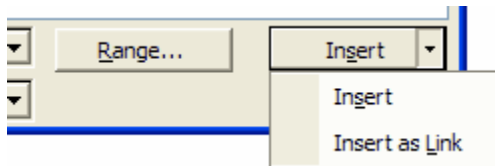
1. Open Word and position the cursor where you wish to insert the file.
2. From the Word *Insert* menu, select *File*.



3. In the *Range* cell type paste the information from the clipboard.



4. Press the **OK** button.
5. From the **Insert** - drop-down to select **Insert as Link**



Insert will set a permanent copy, **Insert as Link** creates a link that is updateable on altering the source document. For Insert as Link to operate you need to have set [Refresh Links](#)

Every package is bookmarked in the RTF document according to the following rules:

- All alphabetic and numeric characters remain the same.
- All other characters (including spaces) are converted to underscores.

For example "UC01: Use Case Model" becomes "UC01__Use_Case_Model"

14.1.5.4 Other Features of Word

Word offers a considerable number of document enhancement tools to complete your project documentation. Here are some of the things you can do with Word and EA generated RTF documentation:

- [Add a Table of Contents](#)
- [Add a Table of Figures](#)
- [Add Headers and Footers](#)
- [Manipulating Tables in Word](#)
- [Refresh Linked Files](#)

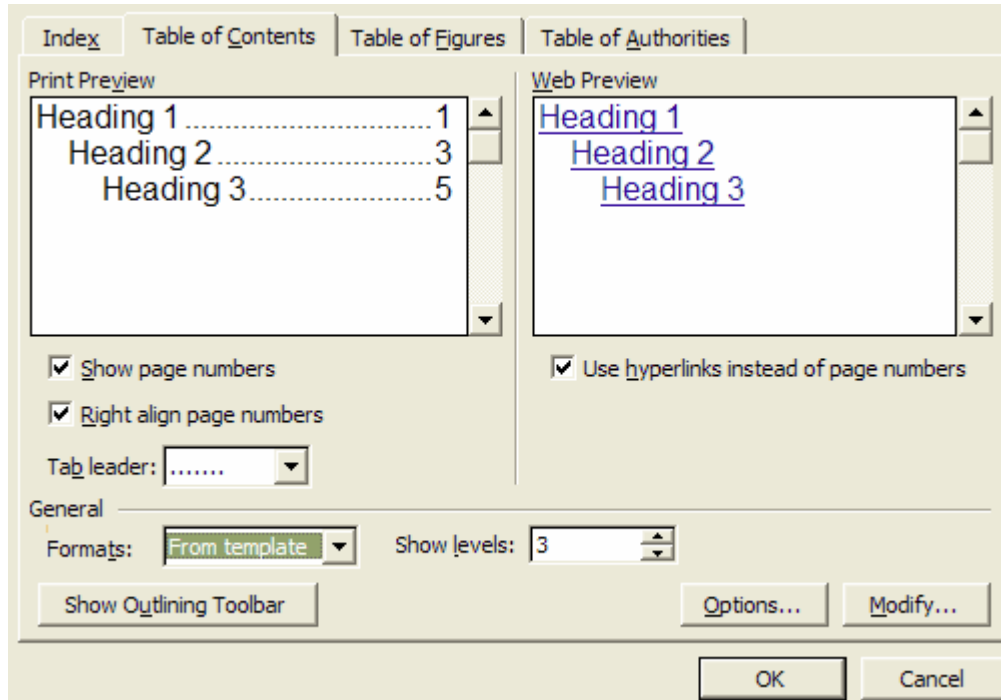
Tip: *Enterprise Architect provides the basic content for your document - use Word to add the presentation and linkages.*

14.1.5.4.1 Add Table of Contents

Among the features of MS word which can be incorporated into generated EA reports is the option to include a table of contents. A table of contents can be used to aid navigation of documentation and enhance the readability of EA RTF reports. This option provide hyperlinks to the diagrams included in the RTF documentation in the electronic version and page numbers for both the printed and electronic documentation.

To include a Table of Contents in the RTF documentation use the following instructions:

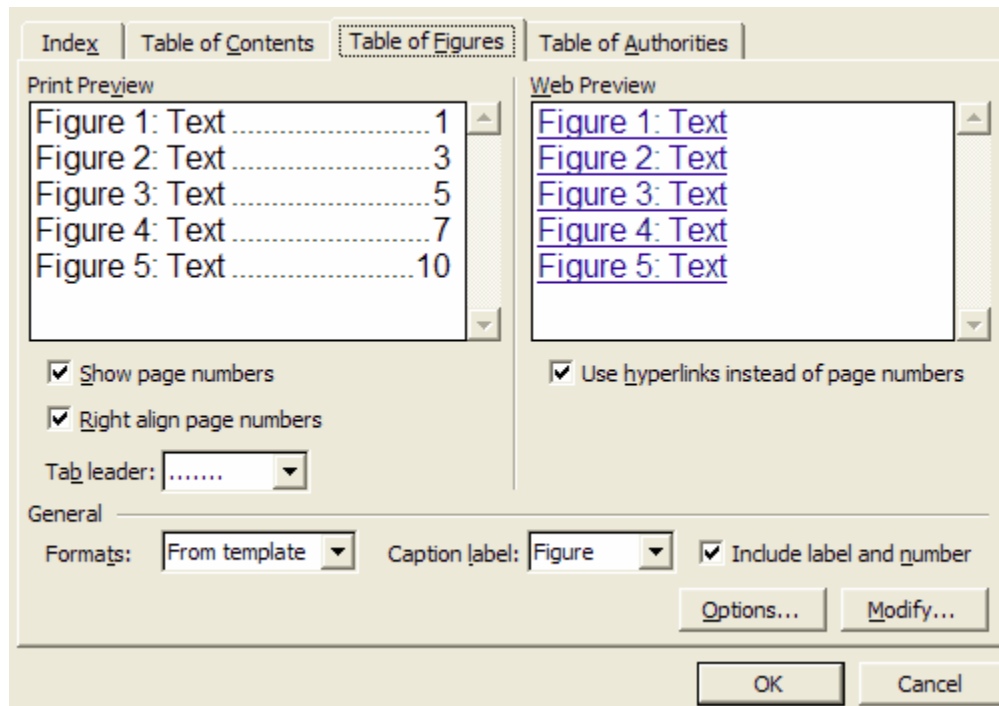
1. Open the EA RTF report that you want to add a Table of Contents in MS Word.
2. From the *Insert* menu mouse over the *Reference* option and from the submenu select the *Index and Tables* option.
3. Open the Table of Contents Tab to set the options that are available for formatting the table of contents. The format of the table of contents is dependant on the level/s of headings created when the RTF is generated, to set the heading style for details in EA RTF documentation refer to the RTF options section



14.1.5.4.2 Add Table of Figures

Among the features of MS word which can be incorporated into generated EA reports is the option to include a table of figures. A table of figures can be used to aid the navigation of the documentation and enhance the readability of EA RTF reports. This option provide hyperlinks to the diagrams included in the RTF documentation in the electronic version and page numbers for both the printed and electronic documentation. To include a Table of Figures in the RTF documentation use the following instructions:

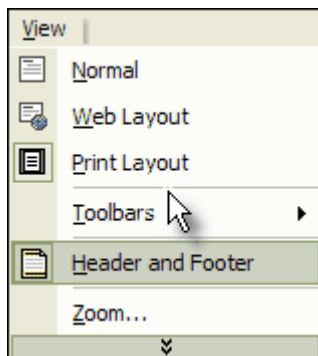
1. Open the EA RTF report that you want to add a Table of Figures in MS Word.
2. From the *Insert* menu mouse over the *Reference* option and from the submenu select the *Index and Tables* option.
3. Open the *Table of Figures* tab to set the options that are available for formatting the table of figures.




14.1.5.4.3 Add Headers and Footers

Among the features of MS word which can be used to enhance the appearance of EA RTF reports is the ability to add Headers and footers to the documentation. To include Headers and Footers in the RTF documentation use the following instructions:

1. Open the EA RTF report that you want to add Headers and Footers in MS Word.
2. From the *View* menu select the *Header and Footer* Option.



3. This will allow the user to enter information into the header section and the footer section of the RTF Documentation.

| | |
|--|---|
| Header | |
| Sparx Systems | |
|  | |
| Class Model | 1 |
| Interactions | 1 |
| Actor1 | 4 |
| Actor2 | 4 |
| Actor3 | 4 |
| Actor4 | 4 |
| Actor5 | 4 |
| Staff_Room | 5 |
| Statecharts | 5 |
| Figure 1 : Interactions | 1 |
| Figure 2 : Interactions | 3 |
| Figure 3 : Statecharts | 6 |

Class Model

The logical model is made up of the Domain Model - a high level model of business objects and relationships between objects suitable for analysing the business process, and the class model - a rigorous model of classes and their inter-relationships, suitable for building a software product.

Interactions

14.1.5.4.4 Manipulating Tables in Word

When generating RTF documentation from EA, tables are included when items such as *Attributes* and *Methods* are selected in the *For each Object* section in the Rich Text Format Report dialog. MS Word offers several levels of customisation for tables and can be used to tidy the formatting of the tables in situations where the margins of the table exceed the dimensions of the page size selected in Word for printing.

Resizing Tables

When the amount of detail for a documented item such as an attribute or operation exceeds the margins of the page in MS Word it is necessary to manually resize the table in order to view all of the details. To manually resize the table use the following instructions:

1. Select the table which exceeds the margin size.
2. Mouse over the border of the table until the mouse pointer changes into the icon shown below.

Message Attributes

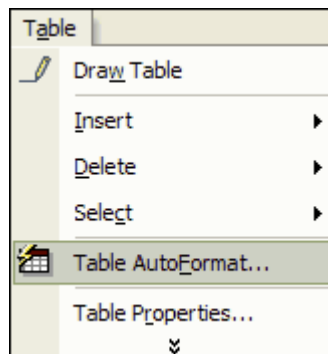
| Attribute | Type | Notes |
|---------------------|------------------------------|-------|
| <u>sentTime</u> | protected : <i>Date</i> | |
| <u>receivedTime</u> | protected : <i>Date</i> | |
| body | protected : <i>String</i> | |

3. Drag the cursor to the left to reduce the width of the table and then use the *File | Print Preview* to confirm that the table borders are within the page margins.
4. Resize all of the tables that overhang the margins of the page by using the steps detailed above.

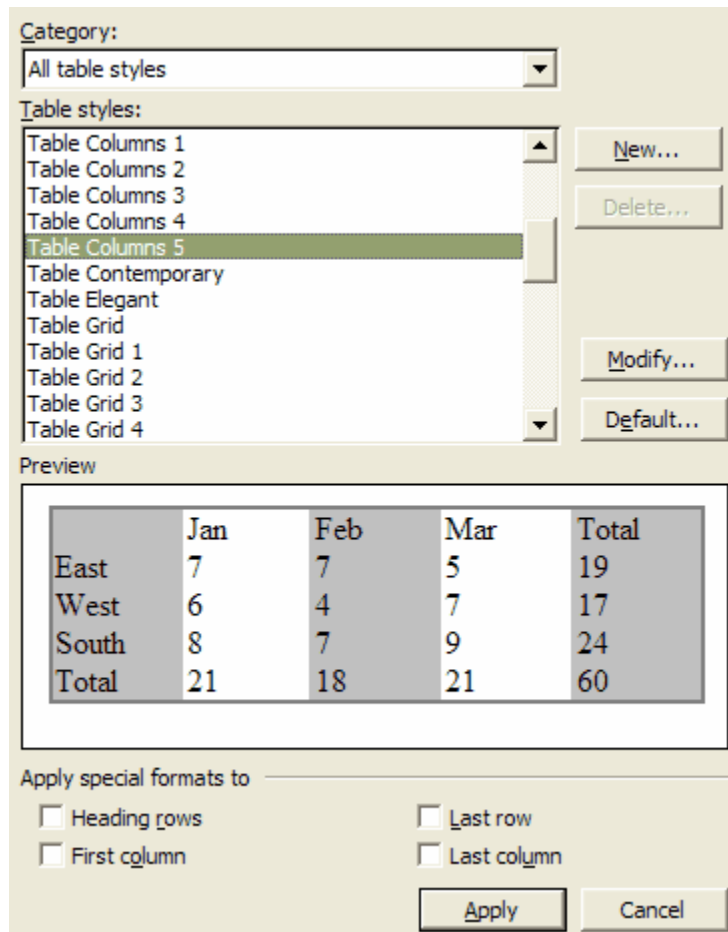
Applying Styles to Tables

One of the customizable properties of MS Word when working with tables is the ability to apply a style to a table which allows the user to rapidly change the appearance of the table to achieve this effect use the following instructions:

1. Open the EA RTF report that you want to change the table styles.
2. Locate the table that you wish to adjust the appearance and select the table.
3. From the *Table* Menu select *Table Auto Format...*



4. This will bring up the Table Autoformat dialog, from here the user may specify a predefined table style from the list of Table styles, or create a new style by pressing the *New* button. The Table styles defined in the Table Autoformat only applies to one table at a time so the user will need to apply the style to each table created individually.

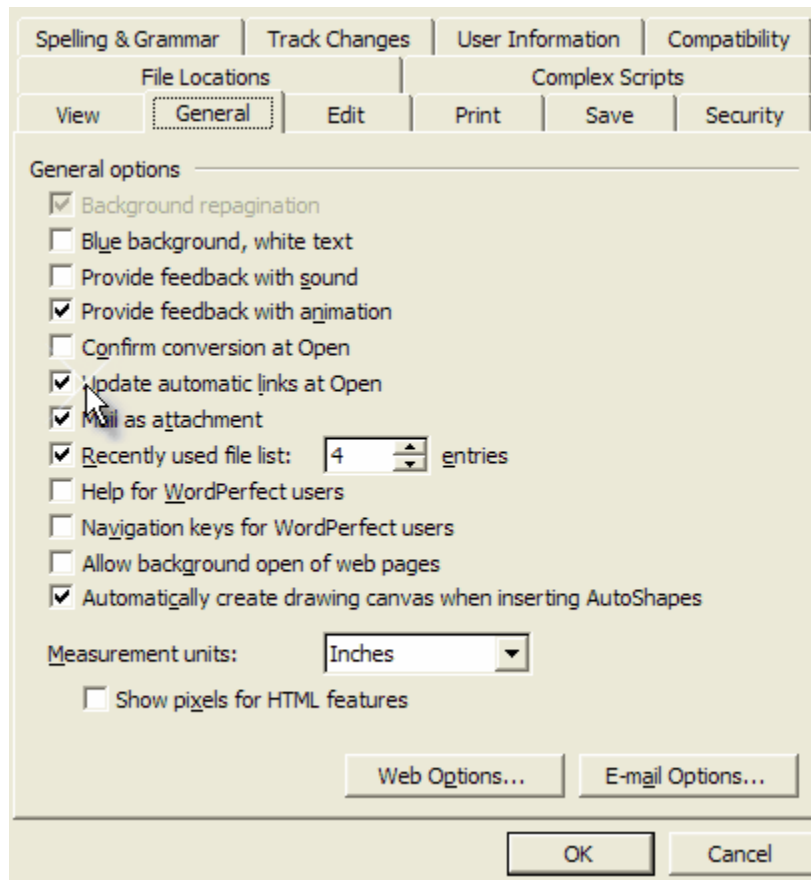


14.1.5.4.5 Refresh Links

If you link into EA reports, then you can regenerate the report and refresh Word links to update the master document without having manually changed anything.

To ensure that links are refreshed in the master document the Update automatic links at Open option must be checked in MS word. To ensure that this setting is established use the following instructions:

1. From within MS Word go to the *Tools* menu and select *Options*.
2. From the *General* tab ensure that the *Update automatic links at Open* is checked.



14.1.6 Other Documents

Enterprise Architect has other RTF based documentation that you can output:

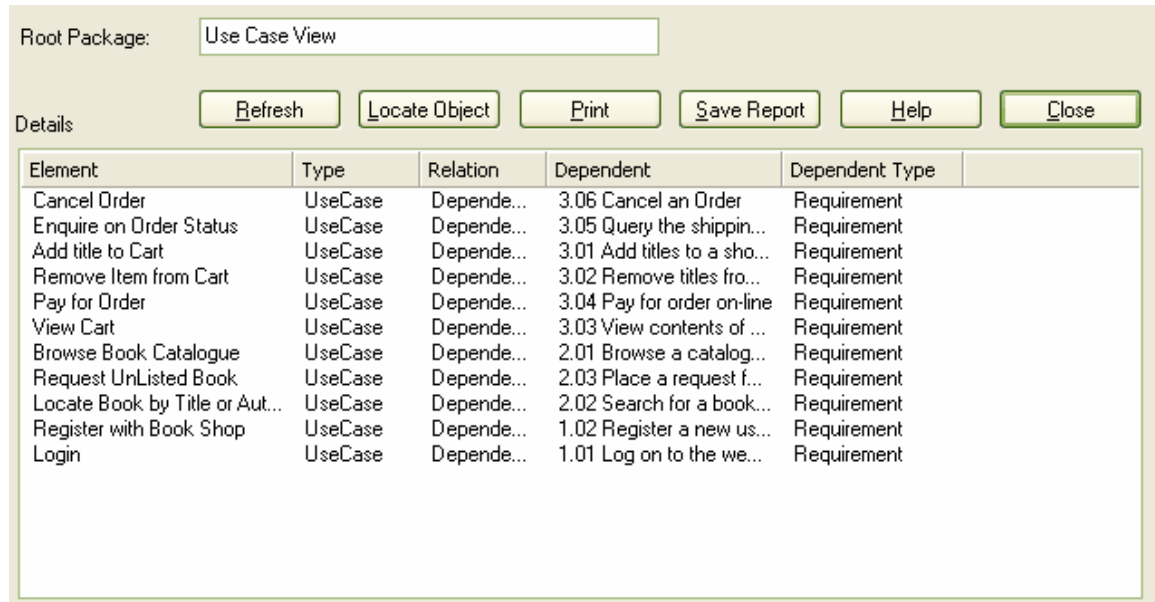
- [Dependency Report](#)
- [Implementation Report](#)
- [Set Target Types Dialog](#)
- [Testing Report](#)

14.1.6.1 Dependency Report

A Dependency Report shows, in a tabular form, a list of elements that realize other elements, i.e. are the target of a Realization connector. A Dependency Report does not show a list of elements which are linked by Dependency connectors - that information is available from the [Hierarchy window](#).

To view a dependency report, follow these steps:

1. In the Project Browser, right click on the package you wish to report on (the report includes all sub-packages as well) to open the context menu.
2. From the *Documentation* submenu, select *Dependency Report...*
3. The *Dependencies* dialog displays a list of all elements that implement other elements in the provided list, together with the elements that are dependent. Save or print the results if required.

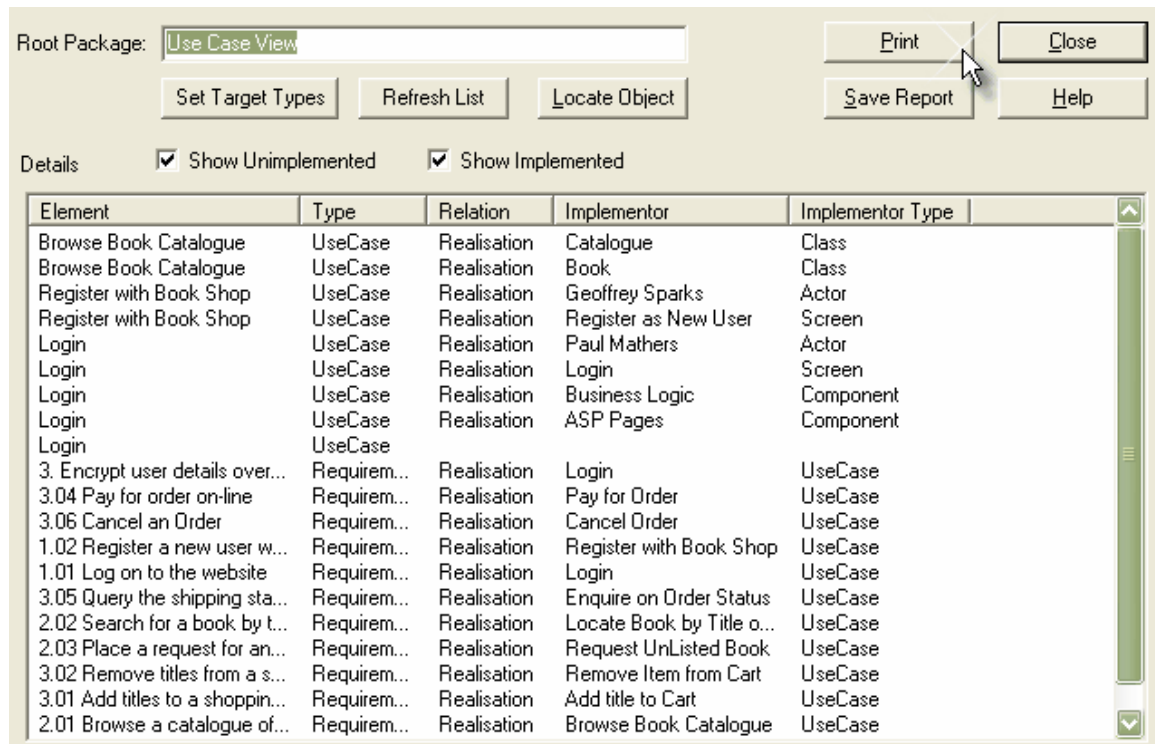


| Control | Description |
|--------------------|--|
| Root Package | The root package. All elements and packages under this will appear in the report. |
| Locate Object | Locate the element selected in the report list in the Project Browser. |
| Refresh | Run the report again. |
| Dependency Details | List of dependency details - lists elements in the current hierarchy and elements that implement them. |
| Print | Print the list. |

14.1.6.2 Implementation Report

To view an implementation report, follow these steps:

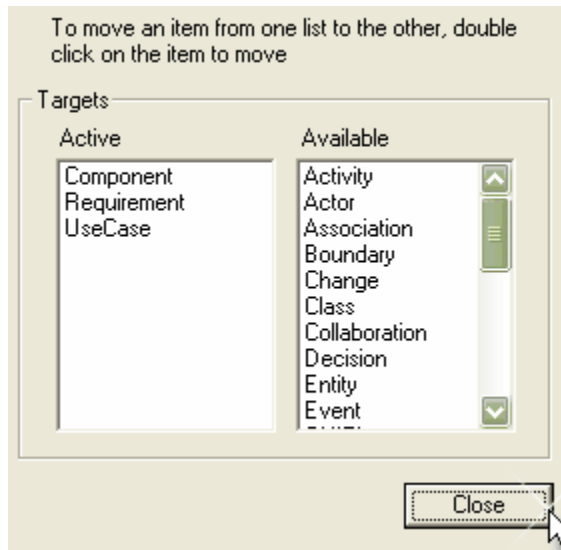
1. In the Project Browser, right click on the package you wish to report on (the report includes all sub-packages as well) to open the context menu.
2. From the *Documentation* submenu, select *Implementation Report...*
3. The *Implementation* dialog displays a list of all elements that require implementers in the provided list, together with the elements that have been connected to those elements by a "Realization" (Implementation) type link. Save or print the results if required.



| Control | Description |
|------------------------|--|
| Root Package | The root package. All elements and packages under this will appear in the report. |
| Show unimplemented | Show non-implemented elements. Implemented elements are those that don't have any other element to realize them (eg. a use case has no component or class to implement the use case behavior) |
| Show Implemented | Show implemented elements. These are elements that do have some element associated with them in a realization relationship. For example a use case has a component that implements it. |
| Locate Object | Locate the element selected in the report list in the Project Browser |
| Refresh | Run the report again |
| Implementation Details | List of implementation details - lists elements in the current hierarchy and elements that implement them |
| Print | Print the list |
| Set Target Types | By default EA only reports on Use Case, Requirement and a couple of other types. You can use this option to set the list of types you want to report on. See further information in the Set Target Types Dialog topic. |

14.1.6.3 Set Target Types Dialog

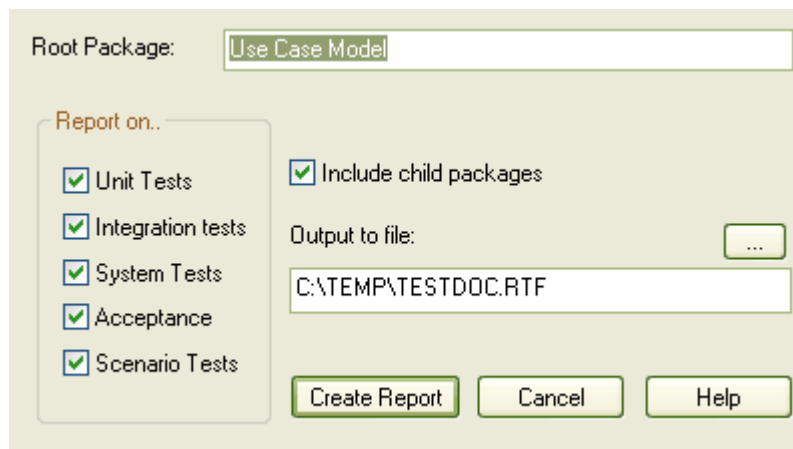
The *Set Target Types* dialog is accessed by pressing *Set Target Types* on the *Implementation* dialog. This dialog allows you to set the types of elements that will appear in the report as requiring implementation. Double click on an element in either list to move it to the other list.



14.1.6.4 Testing Report

To view a testing report, follow these steps:

1. In the Project Browser, right click on the package you wish to report on (the report can include all sub-packages as well, if 'Include child packages' is selected) to open the context menu.
2. From the *Documentation* submenu, select *Testing Documentation...*
3. The *Create Test Documentation* dialog specifies which test types will be documented in the report, and the output file location. The test data originates in the docked testing window, where tests are created and attached to a respective object.



14.2 HTML Reports

Enterprise Architect allows export of an entire model or a single branch of the model to HTML Web pages. The HTML report provides an easy to use, highly detailed, Javascript based model tree. In addition, hyperlinked elements make browsing to related information a breeze.

The current implementation is based on internal and external templates and generated Javascript. The ability to edit all templates will be added in a future version of Enterprise Architect.

Tip: You can create [Web Style Templates](#) to customize your HTML output.

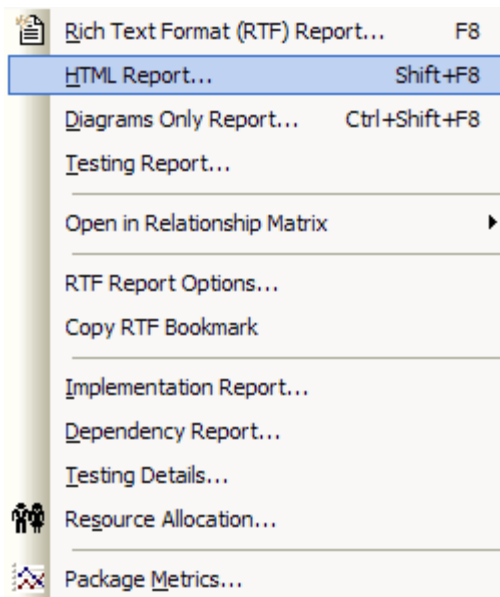
See Also

- [Creating an HTML Report](#)
- [The Generate HTML Report Dialog](#)
- [Making the Output Available on the Web](#)
- [Web Style Templates](#)

14.2.1 Creating an HTML Report

To Create an HTML Report

1. In the Project Browser, right click on the root package for the report - all child packages will be included in the output.
2. From the *Documentation* submenu within the context menu, select *HTML Documentation*. This opens the *Generate HTML Report* dialog.



3. Select an *Output* directory for your report. Set any other required [options](#).
4. Press *Generate* to generate the report - the progress meter will track total percentage complete.
5. Once complete, press *View* to launch your default HTML viewer and view the web pages.

Package: Project Models

Title: Project Models

Output to: C:\Temp ...

Style: <default>

File extension: .html

Preserve Whitespace in Notes

No page for Note and Text items

Image Format

PNG

GIF

Include

Test Cases

Maintenance Items

Resource Allocation

Hyperlinked Files

System

Glossary

Model Tasks

Model Issues

Progress

Generate

View

Close

Help

14.2.2 The Generate HTML Report Dialog

The *Generate HTML Report* dialog is used to generate documentation about your model in HTML format. There are various settings to choose from to control the output - see below.

| Element | Description |
|---|--|
| Package | The package you are creating documentation for. |
| Title | Specify the title for your HTML documentation, set as Package by default. |
| Output to | The directory your documentation will be saved to. |
| Style | Select a web style template to apply to your documentation (optional). |
| File extension | Specify the file extension you want your HTML documentation files to have - the default is .htm. |
| Image Format | Select the image file format you want your images to be saved as - either PNG or GIF. |
| Preserve White space in Notes | Select to preserve existing white space in your notes, deselect to remove white space. |
| No page for Note and Text items | Select if you do not wish to include a page for your notes and text items in the HTML report. |
| Include - <ul style="list-style-type: none"> • Test Cases • Maintenance Items • Resource Allocation • Hyperlinked Files | Select to include any of these areas of your model in your report. |
| System - <ul style="list-style-type: none"> • Glossary • Model Tasks • Model Issues | Select to include any of these areas of your model in your report. |
| OK | Press to generate a report according to the settings you have selected. |
| View | View your report - note that your report can only be viewed after it has been generated (see above). |
| Close | Close the dialog. |
| Help | Opens the Creating an HTML Report help page. |

14.2.3 Making the Output Available on the Web

The web report produced is compatible with any standard web server - either on Unix or Windows platform. Simply bundle up the entire output directory and place within the context of your web server. All path names should be relative and case sensitive.

14.2.4 Web Style Templates

The *HTML and CSS Style Editor* allows you to edit the HTML associated with various sections of the HTML Report facility in Enterprise Architect. You would typically use this functionality to customize a report look and feel for your company or client.

Create or Edit Web Style Templates

1. In the *Resource View*, expand the *Templates* folder.
2. If you wish to edit an existing template, expand the *Web Style Templates* tree, double click on the template name, or right click and select *Modify HTML Style Template* from the context menu. The *HTML and CSS Style Editor* will open. See below for further details.

-OR-

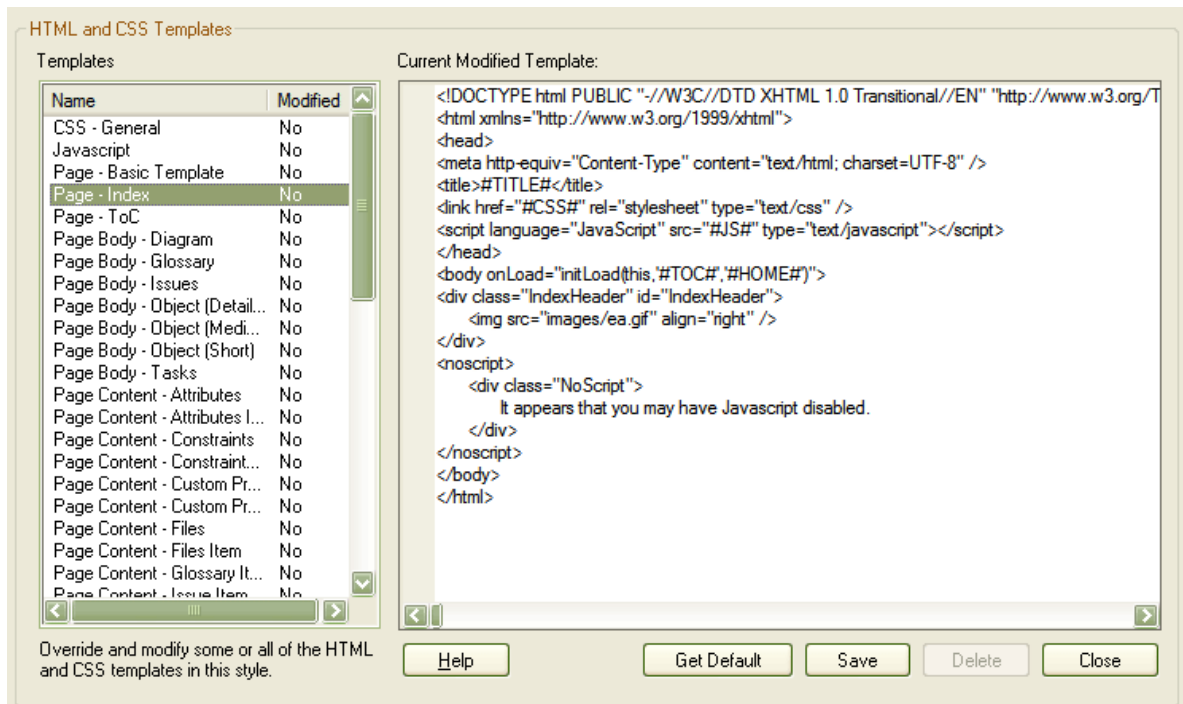
3. To create a new template, right click on *Web Style Templates* and select *Create HTML Template* from the context menu.
4. Enter a name for the new template when prompted to do so. The *HTML and CSS Style Editor* will open. See below for further details.

Tip: To delete a template, right click on it and select *Delete HTML Template* from the context menu.

The *HTML and CSS Style Editor* (shown below) contains a list of all available HTML fragments for modification and customization.

Each fragment typically contains HTML plus one or more special tag names that EA will replace with information during generation. Currently you cannot alter the content within the tag names, but you can omit a complete tag by removing it, or alter its basic display properties in the surrounding HTML.

Special tag names are delimited by "#" characters - eg. #NOTES#



- **Get Default** retrieves the default EA template for the currently selected template item in the left hand list.
- **Save** saves your version of the template for this style only.
- **Delete** removes your modified version of the template - which will cause EA to use the default template during report generation.

To select a template during generation - use the **Style** drop list on the **Generate HTML Report** dialog. Once a style is selected, EA will apply that to the current report. Select <default> for the inbuilt style.

Note: Each time EA generates the web report it overwrites these files, so you will need to back up your modified versions and copy them back in after every update.

14.3 Virtual Documents

You can create "virtual documents" in EA. This allows you to set up document objects and insert any packages you wish into the document. For example, you could create a document which includes a Sequence Diagram package and a Code Engineering packages (these packages are included in the [EA example project file](#)).

To create a virtual document, you first need to [Create a document object](#), [Add Packages to Your Document Object](#), then [Generate the Document](#).

You can include any combination of packages, and [add](#) or [delete](#) packages as required. You can also [Rearrange the Package Order](#) within the documentation.

The document will obtain its contents dynamically - ie. you don't need to update the document if you make a change to one of the packages included in it.

Tip: You can create as many document objects as you wish, for as many combinations of packages as required. You can include the same package in multiple objects.

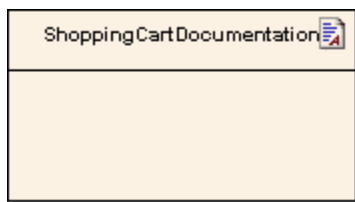
14.3.1 Create a Document Object

To create a "virtual document" you need to create a document object. Follow the steps below:

1. Create a new class diagram.
2. From the UML Toolbox, drag the **Class** icon onto the diagram to create a new class. Name the class something appropriate - in this example we will be including documentation relevant to the shopping cart components of the model, so we will call it "ShoppingCartDocumentation".
3. In the **Stereotype** field of the Class Properties dialog, enter **Model Document**. This is crucial to the virtual documentation process.

Note: The stereotype must be **Model Document** in order for the process to work correctly.

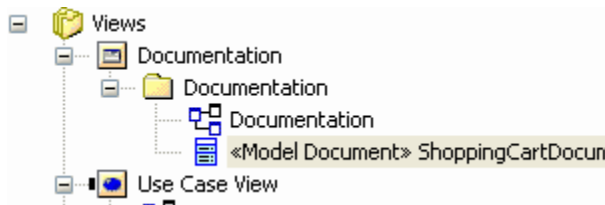
4. Press **OK** - this will create your document class.



Notice the small icon on the top right-hand corner of your class which indicates that this is a document object.

Tip: Resize your *ShoppingCartDocumentation* as required for neatness.

Your document object will appear in the Project Browser as shown below:

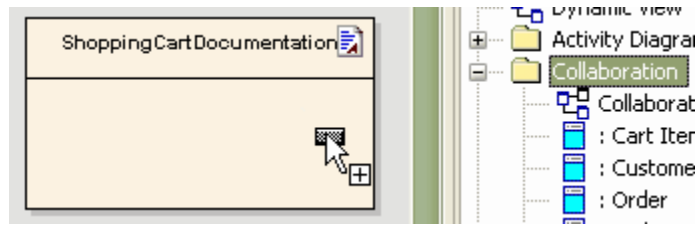


The next step is to [add packages to your document](#).

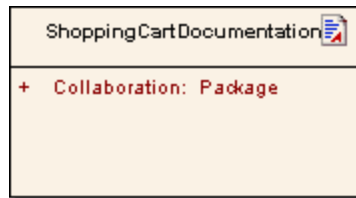
14.3.2 Add Packages to Your Document Object

To add packages to your document object, follow the example below. As this document object is called *ShoppingCartDocumentation*, this example will explain how to add shopping cart related packages to the object.

1. Keeping your Documentation diagram open, find a package you want to add to the documentation in the Project Browser - for example, the Collaboration package in the Dynamic view from the [EA example project file](#).
2. Drag and drop the Collaboration package from the Project Browser onto the document object as shown below:



The title of the package will appear in the document object, as shown below:



This means that the Collaboration package will now be included in the document when you [generate it](#). You can add as many packages from as many different views as desired using the above method.

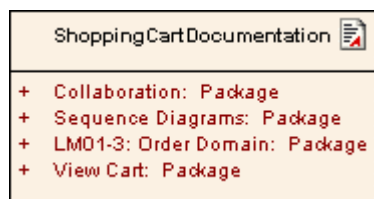
The next step is to [generate your document](#). You can also [rearrange](#) or [delete packages](#) if you wish.

14.3.3 Rearrange the Package Order

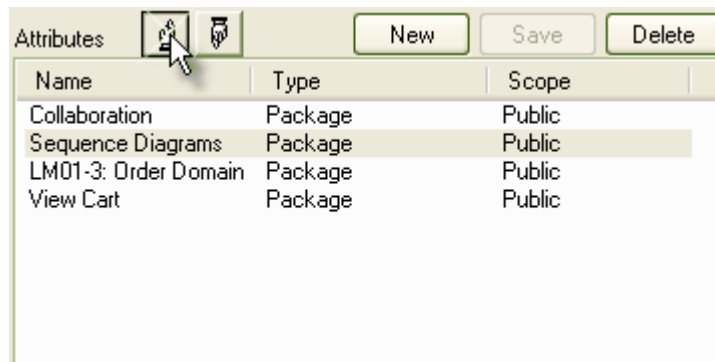
If you have more than one package in your document object, you can rearrange the order by following this example.

Note that the example now has 4 packages included from the [EA example project file](#):

- Collaboration Package (from the Dynamic View)
- Sequence Diagrams Package (from the Dynamic View)
- LM01-3: Order Domain Package (from the Logical View)
- View Cart Package (from the Logical View)



1. Right click on the document object and select **Attributes** from the context menu. This will open the **Attributes** dialog.
2. On the **Attributes** list, use the **Up** and **Down** buttons to change the order the packages will be included in the documentation.



- When you are satisfied with the order of your packages, press **OK**.

See Also

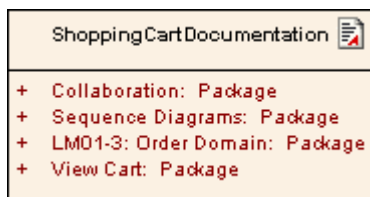
- [Delete a package from your document object](#)
- [Generate the Document.](#)

14.3.4 Delete a Package from Your Document Object

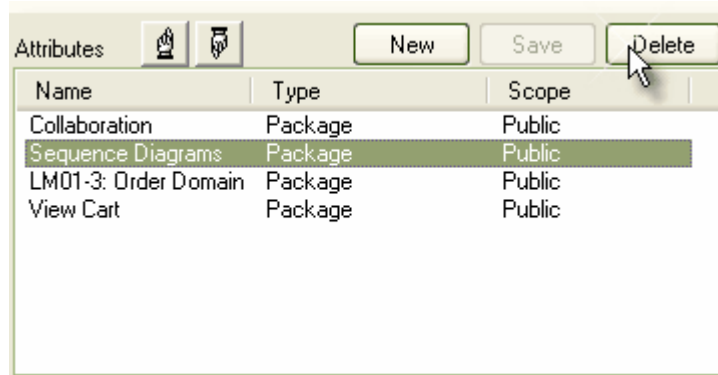
You can delete a package from your document object by following this example:

Note that our example now has 4 packages included from the [EA example project file](#):

- Collaboration Package (from the Dynamic View)
- Sequence Diagrams Package (from the Dynamic View)
- LM01-3: Order Domain Package (from the Logical View)
- View Cart Package (from the Logical View)



- Right click on the document object and select **Attributes** from the context menu. This will open the **Attributes** dialog.
- On the **Attributes** list, select the package you wish to delete.
- Press **Delete** to remove the package from the document.

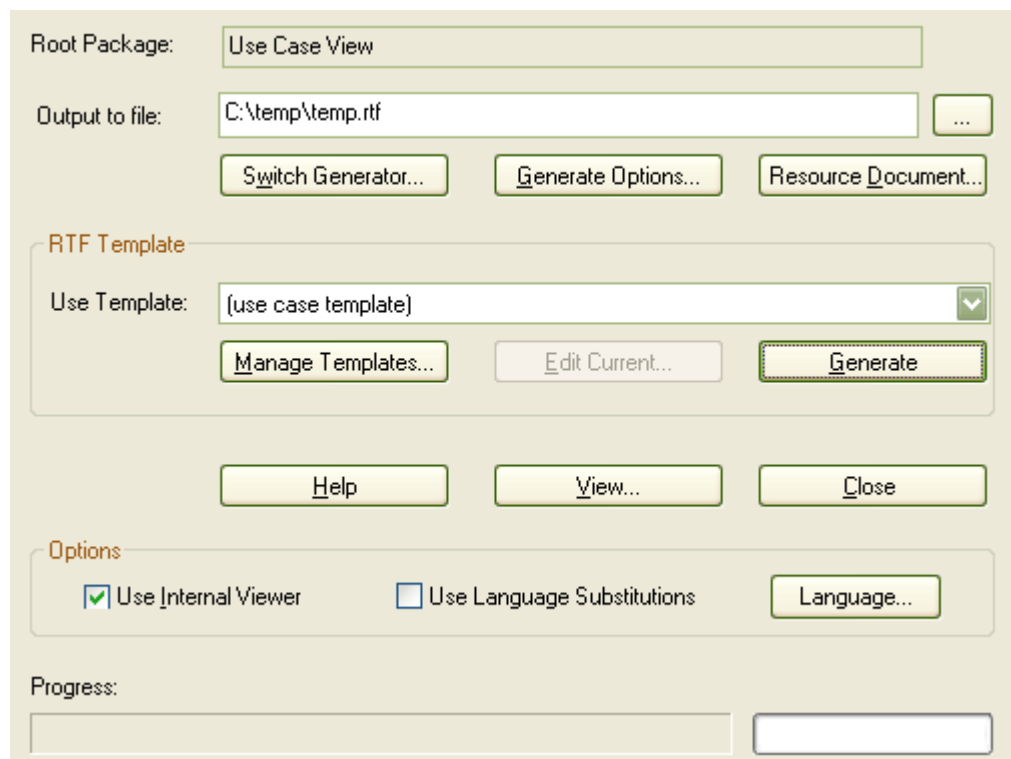
**See Also**

- [Rearrange the Package Order](#)
- [Generate the Document](#)

14.3.5 Generate the Document

To generate the documentation listed in the document object, follow the steps below:

1. Select the element on the diagram.
2. From the *Element* menu, select *Documentation*. This will open the *Generate RTF Documentation* dialog (shown below).



3. Set the options for this as required - see [The RTF Report Dialog](#) and related topics for further information on these settings.
4. Press **Generate** to create the documentation.
5. Press **View** to view the documentation. For the example given in this section, the document will include the following packages from the [EA example project file](#):
 - Collaboration Package (from the Dynamic View)
 - Sequence Diagrams Package (from the Dynamic View)
 - LM01-3: Order Domain Package (from the Logical View)
 - View Cart Package (from the Logical View)

See Also

- [Rearrange the Package Order](#)
- [Delete a Package from Your Document Object](#).

Part

15

15 Automation and Scripting

Automation provides a way for other applications to access the information in an EA model using Windows OLE Automation (ActiveX). Typically this will involve scripting clients such as MS Word or Visual Basic. This section describes how to access the Enterprise Architect automation interface.

For the key methods available see:

- [The Repository Functions](#)
- [The Project Functions](#)

For Details of the main elements within the Automation Interface see:

- The [Repository](#) which represents the model as a whole and provides entry to model packages and collections;
- [Elements](#) which are the basic structural unit (eg. Class, Use Case and Object)
- [Element Features](#) which are attributes and operations defined on an element
- [Diagram](#) - the visible drawings contained in the model
- [Connectors](#) - relationships between elements

See Also

- [The Automation Interface](#)
- [The Project Interface \(XML\)](#)
- [Add-ins](#)

15.1 The Automation Interface

Introduction

The Automation Interface provides a way of accessing the internals of EA models. Here are some examples of things you might want to do using the Automation Interface:

- Perform repetitive tasks, such as update the version number for all elements in a model.
- Generate code from a state machine diagram.
- Produce custom reports.
- Ad hoc queries.

and much more.

Connecting to the Automation Interface

All development environments capable of generating ActiveX Com clients should be able to connect to the Enterprise Architect Automation Interface. This guide provides detailed instructions on [connecting to the interface](#) using Microsoft Visual Basic 6.0, Borland Delphi 7.0 and Microsoft C#. There is also more detailed instruction on how to [set-up Visual Basic](#); the principles should be applicable to other languages.

Examples and Tips

Instruction on how to use the Automation Interface is provided by means of sample code. See [pointers to the samples](#) and other [available resources](#). Also, you will probably want to consult the extensive [Reference Section](#).

Calling Executables from EA

EA can be set up to call an external application. You can pass parameters on the current position selected in the Project View to the application being called. For instructions, go to [Calling Executables from EA](#). A more sophisticated method is to create [Add-Ins](#), which are discussed in a separate section.

15.1.1 Using the Automation Interface

This chapter provides instructions on how to connect to and use the interface.

See Also

- [Connecting to the Interface](#)
- [Set Up VB](#)
- [Examples and Tips](#)

15.1.1.1 Connecting to the Interface

All development environments capable of generating ActiveX Com clients should be able to connect to the Enterprise Architect Automation Interface.

By way of example, the following text describes how to connect using several such tools. The procedure may vary slightly with different versions of these products.

Microsoft Visual Basic 6.0

1. Create a new project.
2. Click on the *Project* menu and select *References*.
3. Select *Enterprise Architect Object Model 2.0* from the list. (If this does not appear, go to the command line and re-register EA using ea.exe /unregister then ea.exe /register).
4. Refer to the general library documentation on the use of classes. The following example creates and opens a repository object:

```
Public Sub ShowRepository()  
    Dim MyRep As New EA.Repository  
    MyRep.OpenFile "c:\eatest.eap"  
End Sub
```

Borland Delphi 7.0

1. Create a new project.
2. Click on the *Project* menu and select *Import Type Library*.
3. Select *Enterprise Architect Object Model 2.0* from the list. (If this does not appear, go to the command line and re-register EA using ea.exe /unregister then ea.exe /register).
4. Click the button *Create Unit*.
5. Include EA_TLB in Project1's Uses clause.
6. Refer to the general library documentation on the use of classes. The following example creates and opens a repository object:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    r: TRepository;  
    b: boolean;  
begin
```

```
r := TRepository.Create(nil);  
b := r.OpenFile('c:\eatest.eap');  
end;
```

Microsoft C#

1. Select *Enterprise Architect Object Model 2.0* from the list. (If this does not appear, go to the command line and re-register EA using `ea.exe /unregister` then `ea.exe /register`).
2. Create a new *Windows Application* project.
3. From the Solution Explorer, right-click on *References* and select *Add References* from the menu.
4. Click the *COM* tab.
5. Click the button *Select* then *OK*.
6. Refer to the general library documentation on the use of classes. The following example creates and opens a repository object:

```
private void button1_Click(object sender, System.EventArgs e)  
{  
    EA.Repository r = new EA.RepositoryClass();  
    r.OpenFile("c:\eatest.eap");  
}
```

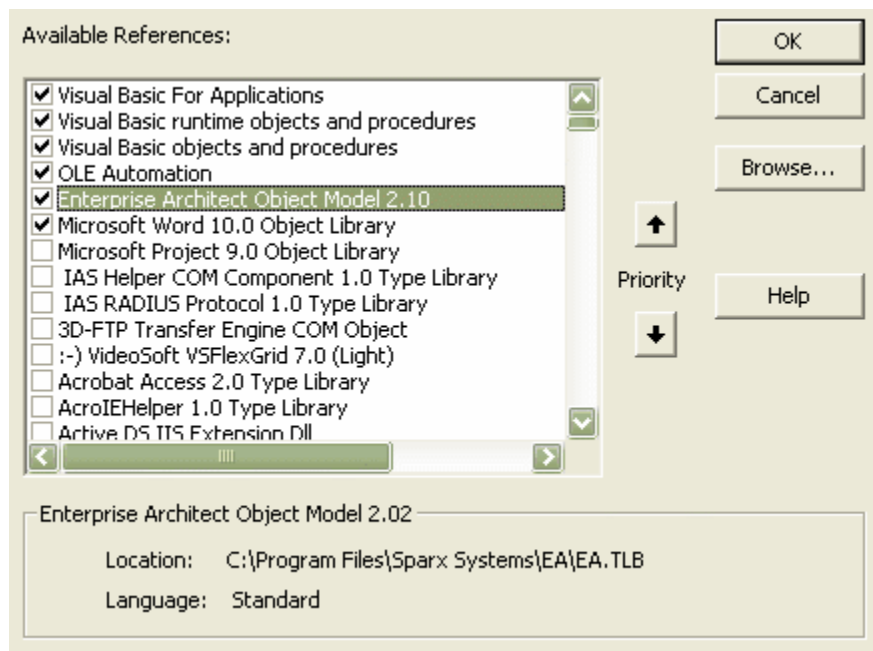
15.1.1.1.1 Set Up VB

Setting References in Visual Basic

The following describes how to use the Enterprise Architect ActiveX interface with Visual Basic. Usage is ensured for version 6. This may vary slightly on versions other than these. It is assumed that VB has been accessed through a Microsoft Application such as VB 6.0, MS Word or MS Access. If the code is not called from within Word, the Word VB reference must also be set.

On creating a new VB project, set a reference to an **Enterprise Architect Type Library** and a **Word Type Library** as follows:

1. Select References on the *Tools* menu.



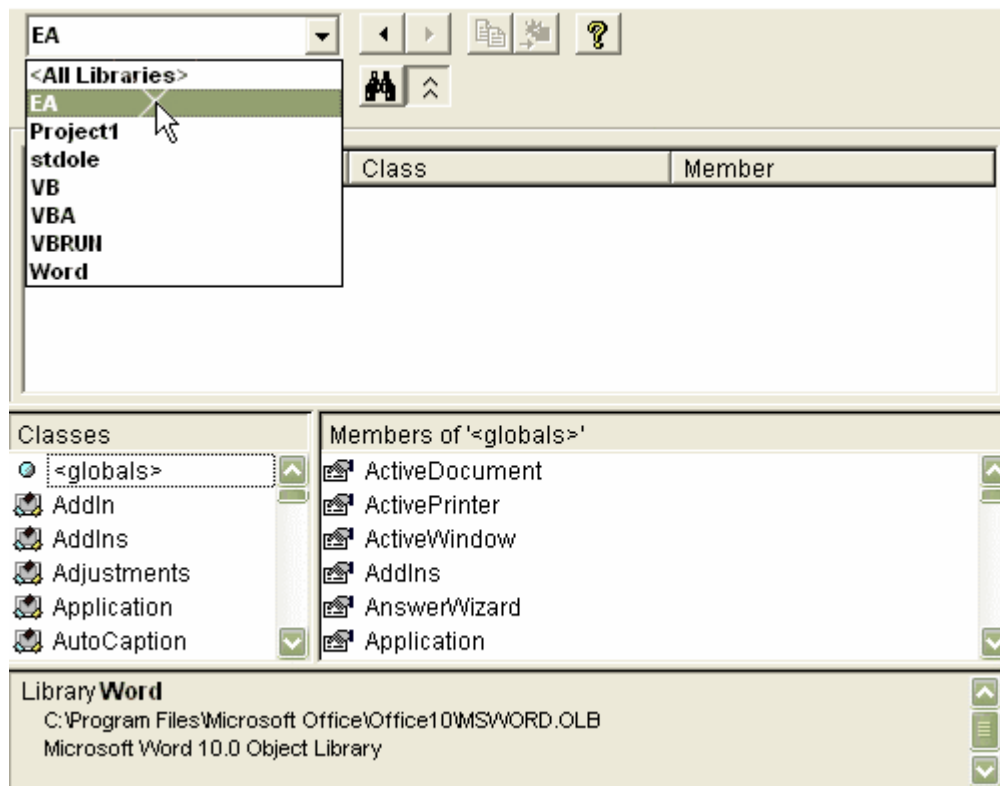
2. Select the check box for *Enterprise Architect Object Model 2.1* from the list.
3. Do the same for **VB** or **VB Word** - select the check box for the *Microsoft Word 10.0 Object Library*.
4. Press **OK**.

Note: If *Enterprise Architect Object Model 2.0* does not appear in the list, go to the command line and manually re-enter EA using the following:

- To unregister EA: **ea.exe /unregister**
- To register EA: **ea.exe /register**

Visual Basic 5/6 users should also note that the version number of the EA interface is stored in the VBP project file in a form similar to the following:
 Reference="\G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\Program Files\Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02

If you experience problems moving from one version of EA to another, open the VBP file in a text editor and remove this line. Then open the project in Visual Basic and use Project-References to create a new reference to the EA Object model.



Reference to objects in EA and Word should now be available in the Object Browser. This can be accessed from the main menu by selecting *View* then *Object Browser*, or by pressing *F2*.

The drop down list on the top-left of the window should now include EA and Word (if MS-Project is installed this will also need to be set up).

15.1.1.2 Examples and Tips

Instructions for using the interface are provided through sample code. There are several sets of examples:

- VB 6 and C# samples are available in the Samples folder of your EA installation – normally "C:\Program Files\Sparx Systems\EA\Code Samples".
- A series of VB.NET examples is provided in the [reference section](#).
- A comprehensive example of using Visual Basic to create MS Word documentation is available from the internet at www.sparxsystems.com.au/AutIntVB.htm.

Additionally, the following list of tricks and traps should be noted:

- The EA application will appear when you access the interface, you will not be able to close it but you can minimize it if it gets in the way.
- The EA ActiveX Interface is a functional interface rather than a data interface. When you load data through the interface you will encounter a noticeable delay as EA user interface elements (eg. Windows, menus) are loaded along with the data.
- Collections are zero-based. Repository.Models(0) represents the first model in the repository.
- You can create multiple Repository objects - but don't do it. They will manipulate the same data. It is not currently possible to open two .EAP files at once.
- Sometimes during the development of your client software your program may terminate unexpectedly. It may be that EA.exe is left running in such a state that it is unable to support further interface calls. If you encounter inexplicable problems ensure that EA is not running (see Windows Task Manager / Process Tab).

EA Not Closing

If your automation controller was written using the .NET framework, EA will not close even after you release all your references to it. To force the release of the COM pointers, call the memory management functions as shown below:

```
GC.Collect();
GC.WaitForPendingFinalizers();
```

There are additional concerns when controlling a running instance of EA which loads Add-ins - see the [Tricks and Traps](#) topic for details.

See Also

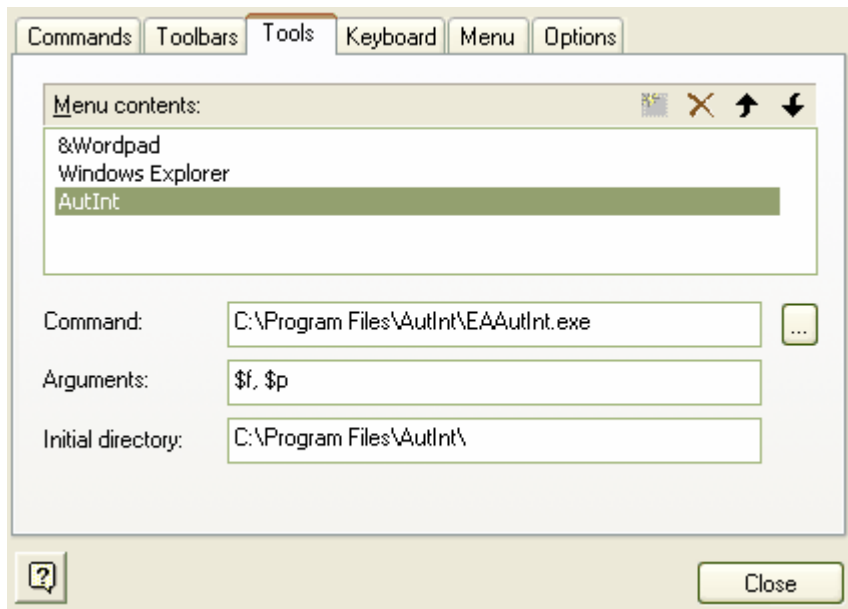
- [Call from EA](#)
- [Available Resources](#)

15.1.1.2.1 Call from EA

Automation Interface Examples: Calling Applications from EA

EA can be set up to call an external application. You can pass parameters on the current position selected in the Project View to the application being called.

To define an application runnable from EA - select from the main menu the *Tools | Customize | Tools* tab option.



With this you can:

- Add a command line for an application
- Define parameters to pass to this application

The parameters required for running the AutInt executable are:

1. The EA file parameter \$f and
2. The current PackageId \$p

Hence the arguments should simply contain: \$f, \$p

The available parameters for passing information to external applications are:

| Parameter | Description | Notes |
|-----------|---------------------------------------|--|
| \$f | Project Name | ie. c:\projects\EAexample.eap |
| \$F | Calling Application (EA) | 'EA' |
| \$p | Current Package Id | ie. 144 |
| \$P | Package GUID | GUID for accessing this package |
| \$d | Diagram ID | Id for accessing associated diagram |
| \$D | Diagram GUID | GUID for accessing associated diagram |
| \$e | Comma separated list of element IDs | All elements selected in the current diagram |
| \$E | Comma separated list of element GUIDs | All elements selected in the current diagram |

Once this has been set up, the application can be called from the main menu in EA using Tools | YourApplication.

15.1.1.2.2 Available Resources

Other available resources include:

| Resource | Download Link |
|---|--|
| Examples - Some PDF copies of documents produced by this application | www.sparxsystems.com.au/resources/developers/autint_vb_examples.html |
| The Application - A brief on the application | www.sparxsystems.com.au/resources/developers/autint_vb_application.html |
| Details - Background on the code | www.sparxsystems.com.au/resources/developers/autint_vb_detail.html |
| Download - Download the code and the executable | www.sparxsystems.com.au/resources/developers/autint_vb_download.html |
| References - Guidelines on the use of the AI | www.sparxsystems.com.au/resources/developers/autint_vb_references.html |

15.1.2 Reference

This chapter provides detailed information on all the objects available in the object model provided by the Automation Interface.

See Also

- [Interface Overview](#)
- [App](#)
- [Enumerations](#)
- [Repository](#)
- [Element](#)

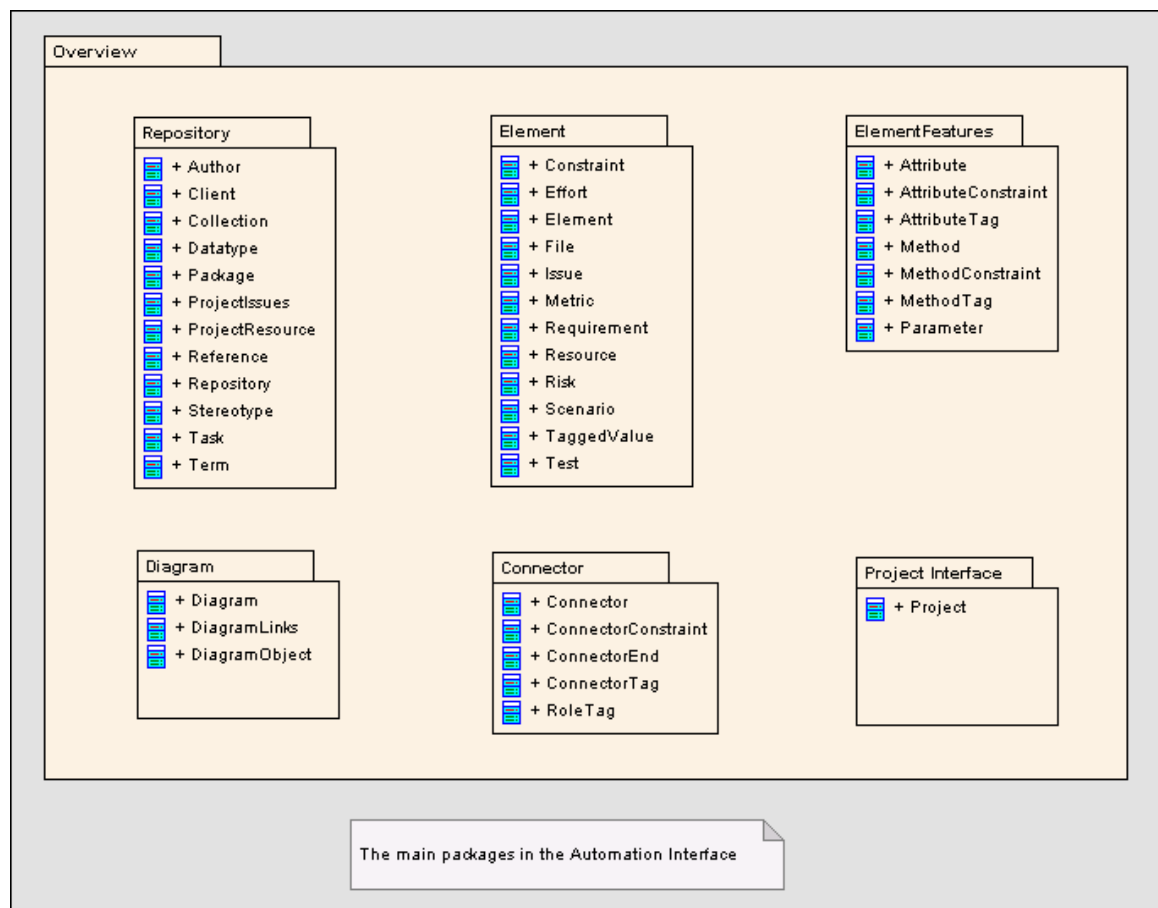
- [Element Features](#)
- [Connector](#)
- [Diagram](#)
- [Project Interface](#)
- [Code Samples](#)

15.1.2.1 Interface Overview

public Package

This diagram illustrates the main interface elements and their associated contents. Each element in this document is creatable by Automation and may be accessed directly in some cases - or through the various collections that exist.

Figure 1 : Automation Interface



Overview

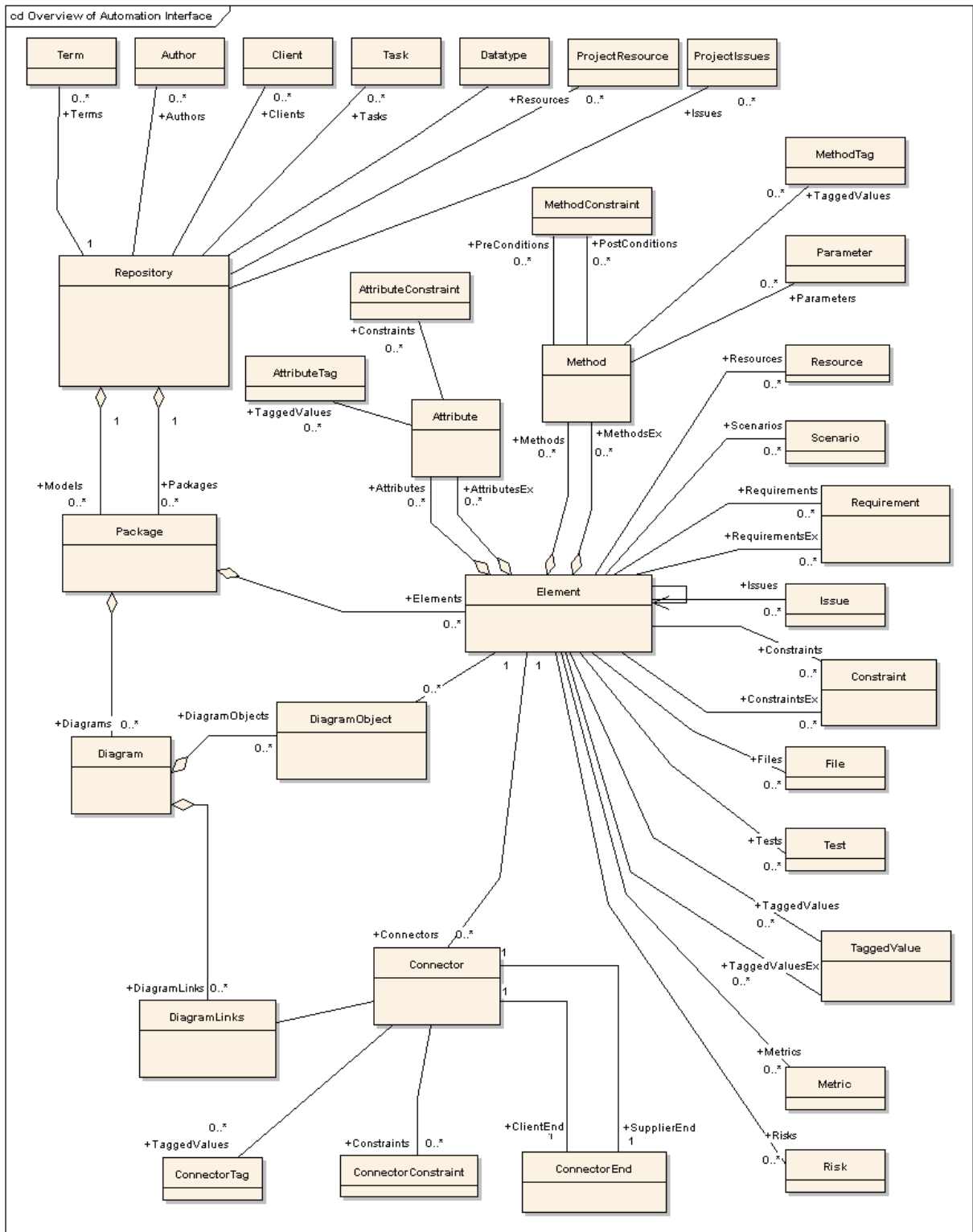
public Package

This package provides an overview of the main elements within the Automation Interface. These are:

- The [Repository](#) which represents the model as a whole and provides entry to model packages and collections;
- [Elements](#) which are the basic structural unit (eg. Class, Use Case and Object)
- [Element Features](#) which are attributes and operations defined on an element
- [Diagram](#) - the visible drawings contained in the model
- [Connectors](#) - relationships between elements

This diagram provides a high level overview of the Automation Interface for accessing, manipulating, modifying and creating EA UML elements. The top level object is the Repository - which contains collections for a variety of system level objects, as well as the main Models collection which provides access to the UML elements, diagrams, packages etc. within the project. In general the Role names applied at the Target end of associations indicate the name of the Collection which is used to access instances of that object.

Figure 2 : Overview of Automation Interface



Internal Links

- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface

- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface

Connectors

| Connector | Source | Target |
|-----------------------------------|---|---------------------------------------|
| <i>Nesting</i> source > target | <i>Connector</i> Contained Element | <i>Overview</i> Containing Element |
| <i>Nesting</i> source > target | <i>Repository</i> Contained Element | <i>Overview</i> Containing Element |
| <i>Nesting</i> source > target | <i>Diagram</i> Contained Element | <i>Overview</i> Containing Element |
| <i>Nesting</i> source > target | <i>Element</i> Contained Element | <i>Overview</i> Containing Element |
| <i>Nesting</i> source > target | <i>Project Interface</i> Contained Element | <i>Overview</i> Containing Element |
| <i>Nesting</i> source > target | <i>ElementFeatures</i> Contained Element | <i>Overview</i> Containing Element |

15.1.2.2 App

The App object represents a running instance of EA. Its object provides access to the Automation Interface.

| Attribute | Type | Notes |
|------------|-------------------|--|
| Repository | <i>Repository</i> | Read Only. Provides a handle to the Repository object. |
| Project | <i>Project</i> | Read Only. Provides a handle to the Project Interface. |
| Visible | <i>Boolean</i> | Read/Write. Whether or not the application is visible. |

GetObject() Support

The App object is creatable and a handle may be obtained by creating one.

In addition, clients may use the equivalent of VB's GetObject() to obtain a reference to a currently running instance of EA.

Use this method to more quickly test changes to add-ins and external clients, since the EA application and data files do not need to be constantly re-loaded.

For example:

```
Dim App as EA.App
Set App = GetObject(, "EA.App")
MsgBox App.Repository.Models.Count
```

Another example which uses the App object without saving it to a variable:

```
Dim Rep as EA.Repository
Set Rep = GetObject(, "EA.App").Repository
MsgBox Rep.ConnectionString
```

15.1.2.3 Enumerations

These enumerations are defined by the Automation Interface:

- [MDGMenus Enum](#)
- [EnumRelationSetType Enum](#)
- [ConstLayoutStyles Enum](#)
- [ObjectType Enum](#)
- [XMIType Enum](#)

15.1.2.3.1 MDGMenus Enum

Use this enumeration when providing HiddenMenus property to [MDG_GetProperty](#).

These options are exclusive of one another and may be read or added to hide more than one menu.

See [MDG_GetProperty](#) for an example of use.

| | |
|----------------|-----------------------------------|
| mgMerge | Hide "Merge" menu option. |
| mgBuildProject | Hide "Build Project" menu option. |
| mgRun | Hide "Run" menu option. |

15.1.2.3.2 EnumRelationSetType Enum

This enumeration represents values returned from the GetRelationSet method of the [Element](#) object.

| Method | Notes |
|-------------------|---|
| rsGeneralizeStart | List of elements which the current element generalizes. |
| rsGeneralizeEnd | List of elements which are generalized by the current element |
| rsRealizeStart | List of elements which the current element realises. |
| rsRealizeEnd | List of elements which are realised by the current element. |
| rsDependStart | List of elements which the current element depends on. |
| rsDependEnd | List of elements which depend on the current element. |
| rsParents | List of all parent elements of the current element. |

15.1.2.3.3 ConstLayoutStyles Enum

The enum values defined here are used exclusively for the [Layout a Diagram](#) method. They allow for the ability to define the layout options as depicted in the Layout a Diagram menu command (refer to [Layout a Diagram](#) for further information):

| Method | Notes |
|--------|-------|
|--------|-------|

| | |
|-----------------------------|---|
| IsDiagramDefault | |
| IsProgramDefault | |
| IsCycleRemoveGreedy | Use the Greedy Cycle Removal algorithm |
| IsCycleRemoveDFS | Use the Depth First Cycle Removal algorithm |
| IsLayeringLongestPathSink | Layer the diagram using the Longest Path Sink algorithm |
| IsLayeringLongestPathSource | Layer the diagram using the Longest Path Source algorithm |
| IsLayeringOptimalLinkLength | Layer the diagram using the Optimal Link Length algorithm |
| IsInitializeNaive | Initialize the layout using the Naive Initialize Indices algorithm |
| IsInitializeDFSOut | Initialize the layout using the Depth First Search Outward algorithm |
| IsInitializeDFSIn | Initialize the layout using the Depth First Search Inward algorithm |
| IsCrossReduceAggressive | Perform aggressive Cross-reduction in the layout process (time consuming) |
| IsLayoutDirectionUp | Direct links to point upwards |
| IsLayoutDirectionDown | Direct links to point downwards |
| IsLayoutDirectionLeft | Direct links to point leftwards |
| IsLayoutDirectionRight | Direct links to point rightwards |

15.1.2.3.4 ObjectType Enum

The ObjectType enumeration identifies EA object types even when referenced through a dispatch interface.

For example:

```
Object ob=Repository.GetElementByID(13);
if ( ob.ObjectType==otElement )
;
else if ( ob.ObjectType==otAuthor )
...

```

15.1.2.3.5 XMIMType Enum

The following enumeration values are used Project.ExportPackageXMI() method. They allow for the specification of the XMI export type :

- xmiEADefault
- xmiRoseDefault
- xmiEA10
- xmiEA11
- xmiEA12
- xmiRose10
- xmiRose11
- xmiRose12
- xmiMOF13
- xmiMOF14
- xmiEA20
- xmiEA21

15.1.2.3.6 PropType Enum

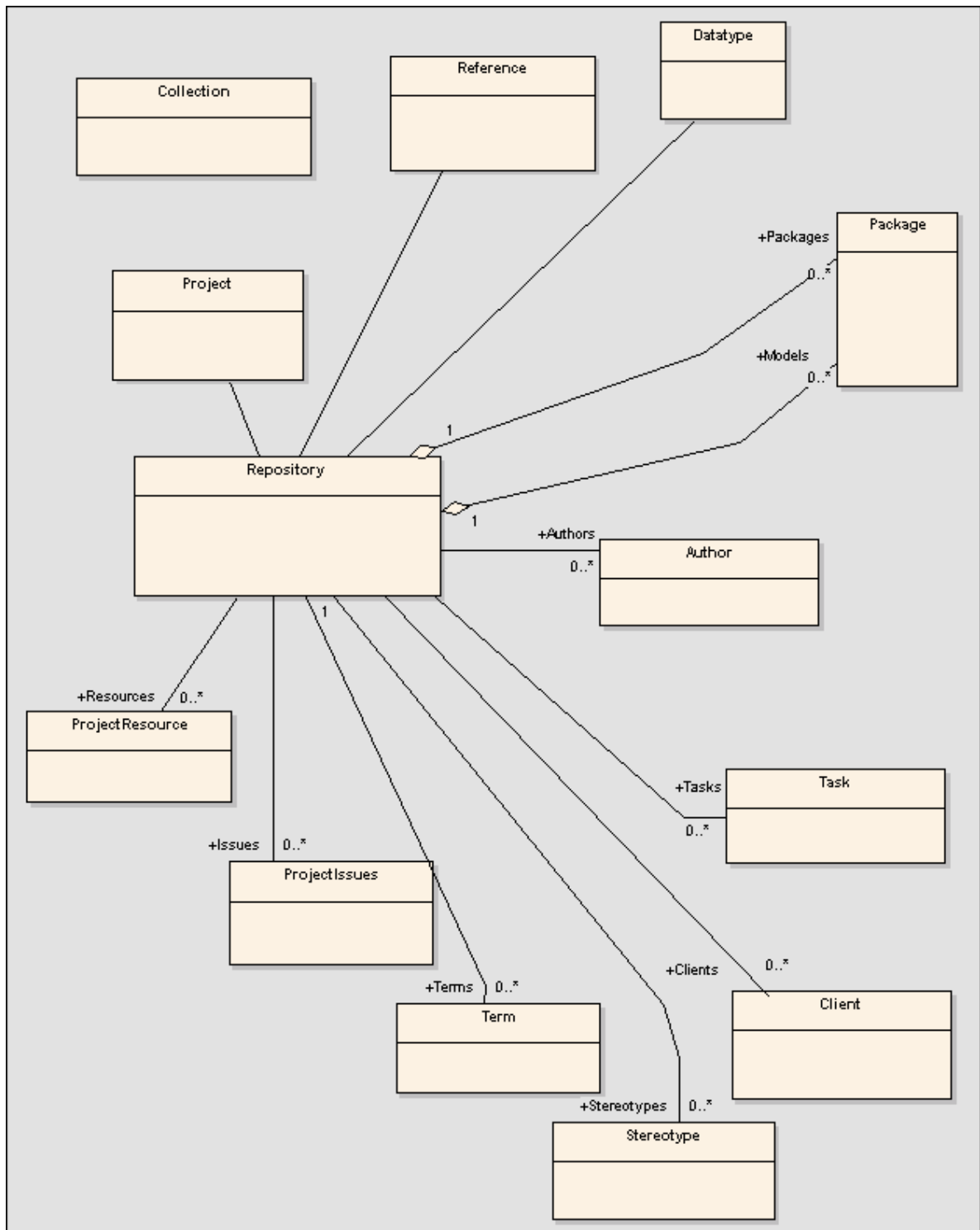
The PropType enumeration gives the automation programmer an indication of what sort of data is going to be stored by this property.

| Element | Meaning |
|-----------------|---|
| ptInteger | 16-bit or 32-bit signed integer. |
| ptBoolean | True or False |
| ptString | Unicode string |
| ptEnum | A string being an entry in the semi-colon separated list specified in the validation field of the Property. |
| ptArray | An array containing values of any type. |
| ptFloatingPoint | 4 or 8 byte floating point value. |

15.1.2.4 Repository

This diagram illustrates the [Repository](#) and its first level functions and collections. From here, the rest of the model may be accessed - using the Models collection and the other system level collections.

Figure 3 : Repository



Repository

public Package

The Repository package contains the high level system objects and entry point into the Model itself using the Model's collection.

15.1.2.4.1 Repository

public Class

The Repository is the main container of all models, packages, elements, etc. You can iteratively begin accessing the model using the Models collection. Also has some convenience methods to directly access elements, packages, etc. without having to locate them in hierarchy first.

Associated table in .EAP file: <none>

Repository Attributes

| Attribute | Type | Notes |
|------------------|----------------------------|---|
| Models | Collection of type Package | Read only. Models of type package and belong to a collection of packages. This is the top level entry point to an EA project file. Each model is a Root Node in the Project Browser and may contain views/packages etc. A Model is a special form of a Package - it has a ParentID of '0'. By iterating through all models, you can access all the elements within the project hierarchy. You may also use the AddNew function to create a new Model. A Model may also be deleted, but remember that everything contained in the Model will be deleted as well! |
| Terms | Collection | Read only. The project Glossary. Each Term object is an entry in the glossary. Add, modify and delete Terms to maintain the Glossary. |
| Issues | Collection | Read only. The System Issues list. Contains Issue objects - each detailing a particular issue as it relates to the project as a whole. |
| Authors | Collection | Read only. The system Authors collection. Contains 0 or more Author objects - each of which can be associated with elements, diagrams etc. as the item author or owner. Use AddNew, Delete and GetAt to manage Authors |
| Clients | Collection | Read only. A list of Clients associated with the Project. You can add new, modify and delete Client objects using this collection. |
| Tasks | Collection | Read only. A list of system tasks (to do list). Each entry is a Task Item - you may add new, modify and delete Tasks. |
| Datatypes | Collection | Read only. The Datatypes collection. Contains a list of Datatype objects - each representing a data type definition for either data modeling or code generation purposes. |
| Resources | Collection | Read only. A list of available resources to assign to work items within the project. Use the add new, modify and delete functions to manage Resources. |
| Stereotypes | Collection | Read only. The stereotypes collection. A list of Stereotype objects which contain information in a stereotype and what elements it may be applied to. |
| PropertyTypes | Collection | Read only. Collection of Property Types available to the Repository. |
| LibraryVersion | Long | Read only. The build number of the EA runtime. |
| LastUpdate | String | Read only. The identifier string identifying the EA runtime session and the timestamp it was set. |
| FlagUpdate | Boolean | Read/Write. Instructs EA to update the Repository with the LastUpdate value. |
| InstanceGUID | String | Read only: The identifier string identifying the EA runtime session. |
| ConnectionString | String | Read only. The filename/connection string of the current Repository |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface |

| | | |
|-------------------|---------|---|
| EnableUIUpdates | Boolean | Read/Write. Set this property to false to improve the performance of changes to the model, eg. Bulk addition of elements to a package. To reveal the changes to the user, call <code>Repository.RefreshModelView()</code> . |
| BatchAppend | Boolean | Read/Write. Set this property to true when your automation client needs to rapidly insert many elements, operations, attributes and/or operation parameters. Set to false when work is complete This can result in 10 to 20 fold improvement in adding new elements in bulk. |
| SuppressEADialogs | Boolean | Read/Write. Set this property in the EA_OnPostNewElement or EA_OnPostNewConnector broadcast events to control whether EA should suppress showing the default properties dialogs to the user when an Element or Connector is newly created. |
| ProjectGUID | String | Read only: Returns a unique ID for the project |
| EnableCache | Boolean | Read/Write: An optimization for pre-loading package objects when dealing with large sets of automation objects |

Repository Methods

| Method | Type | Notes |
|---------------------------------|---------|---|
| OpenFile (String Filename) | Boolean | param: Filename [String - in] The filename of the EA project to open. This is the main point for opening an EA project file from an automation client. Provide the filename of a valid EA project and call this to open it and work with the contained objects. |
| GetElementByID (long ElementID) | Element | param: ElementID [long - in] Get a pointer to an Element using an absolute reference number (local ID). This is usually found using the ElementID property of an element - and stored for later use when you wish to open an Element without using the collection <code>GetAt()</code> function. |
| GetPackageByID (long PackageID) | Package | param: PackageID [long - in] Get a pointer to a Package using an absolute reference number (local ID). This is usually found using the PackageID property of a package- and stored for later use when you wish to open a Package without using the collection <code>GetAt()</code> function. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| GetDiagramByID (long DiagramID) | Diagram | param: DiagramID [long - in] Get a pointer to a Diagram using an absolute reference number (local ID). This is usually found using the DiagramID property of an element - and stored for later use when you wish to open a Diagram without using the collection <code>GetAt()</code> function. |

| | | |
|--------------------------------|-----------|---|
| GetReferenceList (String Type) | Reference | param: Type [String - in] The list type to get Get a pointer to Reference List object. The parameter specifies the list type to get. Valid lists are: Diagram, Element, Constraint, Requirement, Connector, Status, Cardinality, Effort, Metric, Scenario, Status, Test. |
| GetProjectInterface () | Project | Return a pointer to the EA.Project interface - the XML based automation server for EA. Use this interface to work with EA using XML, and also to access utility functions for loading diagrams, running reports etc. |

| | | |
|---|------------|--|
| Exit | | Shut down EA immediately. Used by DotNET programmers where the garbage collector does not immediately release all referenced COM objects. |
| OpenFile2 (string FilePath, string Username, string Password) | Boolean | As for OpenFile() except allows the specification of a password. |
| ShowWindow(long Show) | | param: Show [long - in] Show or hides EA. |
| GetCurrentDiagram() | Diagram | Returns selected diagram. |
| GetConnectorByID(long ConnectorID) | Connector | param: ConnectorID [long - in] Searches the repository for a connector with matching ID. |
| GetTreeSelectedItem(Object SelectedItem) | ObjectType | param: SelectedItem [Object - in] Used to get an object variable and type corresponding to the currently selected item in the tree view. To use this function, create a generic object variable and pass this as the parameter. Depending on the return type, cast it to a more specific type. This object passed back through the parameter can be a package, element, diagram, attribute or operation object. |
| GetTreeSelectedPackage() | Package | Returns the package in which the currently selected tree view object is contained. |
| CloseAddins() | | Called by automation controllers to ensure that Add-ins created in .NET will not linger after all controller references to EA have been cleared. |
| AdviseElementChange (long ObjectID) | Boolean | param: ObjectID [long - in] Provides an Add-In or automation client with the ability to advise the EA user interface that a particular element has changed and if it is visible in any open diagram, to reload and refresh that element for the user. |
| AdviseConnectorChange (long ConnectorID) | Boolean | param: ConnectorID [long - in] Provides an Add-In or automation client with the ability to advise the EA user interface that a particular connector has changed and if it is visible in any open diagram, to reload and refresh that connector for the user. |
| OpenDiagram (long DiagramID) | Boolean | param: DiagramID [long - in] Provides a method for an automation client or Add-In to open a diagram by its ID. The diagram will be added to the tabbed list of open diagrams in the main EA view. |
| CloseDiagram (long DiagramID) | Boolean | param: DiagramID [long - in] Provides a method to close a diagram (if open) by its ID in the current list of diagrams EA has open. |

| | | |
|-------------------------------------|----------------------------|---|
| ActivateDiagram (long DiagramID) | Boolean | param: DiagramID [long - in] Provides the means to activate an already open diagram (ie. make it be the active tab) in the main EA user interface. |
| SaveDiagram(long DiagramID) | Boolean | param: DiagramID [long - in] Call this to save an open diagram by its ID number. Assumes the diagram will be open in the main user interface Tab list. |
| ReloadDiagram(long DiagramID) | Boolean | param: DiagramID [long - in] Reloads the diagram specified by ID value. This would commonly be used to refresh a visible diagram after code import/export or other batch process where the diagram requires complete refreshing. |
| CloseFile() | | Closes any open file |
| GetTreeSelectedItemType() | ObjectType | param: none Returns the type of object currently selected in the tree. One of: <ul style="list-style-type: none"> • otDiagram • otElement • otPackage • otAttribute • otMethod |
| GetTechnologyVersion (string ID) | String | param: ID [string] Returns the version of the MDG Technology resource with the specified technology ID. |
| IsTechnologyLoaded (string ID) | Boolean | param: ID [string] Returns True, if the MDG Technology resource with the specified technology ID is loaded into the repository. Returns False, otherwise. |
| ImportTechnology(string Technology) | Boolean | param: Technology [string] Installs the given MDG Technology resource into the repository. The Technology parameter represents the contents of the Technology resource file. Returns True, if the technology is successfully loaded into the model. Returns False, otherwise. |
| GetCounts() | | Returns a set of counts from a number of tables within the base EA repository. These can be used to determine whether records have been added or deleted from the tables for which information is retrieved. |
| GetElementSet() | Collection | Returns a set of elements as a collection based on an input of element ID numbers in comma separated form. eg. GetElementSet("34,56,21,5") |

| | | |
|---|------------------------------|--|
| DeleteTechnology(string ID) | Boolean | <p>param: ID [string]</p> <p>Removes the MDG Technology resource with the specified technology ID from the repository.</p> <p>Returns True, if the technology is successfully removed from the model. Returns False otherwise.</p> |
| AddTab(string TabName, string ControlID) | activex custom control | <p>param: TabName [String - in] param: ControlID [String - in]</p> <p>Allows an ActiveX custom control to be added as a tabbed window.</p> <p>TabName is used as the tab caption. ControlID is the ProgID of the control. eg. Project1.UserControl1</p> <p>EA creates a control and if successful, returns its IUnknown pointer which may be used by the caller to manipulate the control.</p> |
| CreateOutputTab (string Name) | | <p>param: Name [String in] Creates a tab in the Output View with the specified name.</p> |
| RemoveOutputTab (string Name) | | <p>param: Name [String in] Removes the tab in the Output View with the specified name.</p> |
| WriteOutput (string Name, string String, long ID) | | <p>param: Name [String in] param: String [String in] param: ID [long in]</p> <p>Writes the text in String to the tab in the Output View with the specified name. It also associates the text in the Output View with the specified ID.</p> |
| ClearOutput(string Name) | | <p>param: Name [String in]</p> <p>Removes all the text in the tab in the Output View with the specified name.</p> |
| EnsureOutputVisible(string Name) | | <p>param: Name [String in]</p> <p>Ensures that the tab in the Output View with the specified name is visible to the user. The Output View will be made visible if it is hidden.</p> |
| RefreshModelView(long PackageID) | | <p>param: PackageID [Long in]</p> <p>Reloads entire model, updating the user interface. If PackageID is 0, the entire model is reloaded. If PackageID represents a valid PackageID, only that package will be reloaded.</p> |
| GetElementByGuid(string Guid) | Element | <p>param: Guid [String - in]</p> <p>Returns a pointer to an Element in the repository using the Element's GUID reference number (global ID). This is usually found using the ElementGUID property of an element - and stored for later use when you wish to open an Element without using the collection GetAt() function.</p> |
| GetConnectorByGuid(string Guid) | Connector | <p>param: Guid [String - in]</p> <p>Returns a pointer to a Connector in the repository with matching Guid. This is usually found using the ConnectorGUID property of a Connector.</p> |

| | | |
|---|------------|---|
| ShowDynamicHelp(string Topic) | | <p>param: Topic [String - in]</p> <p>Shows the help topic, specified by the Topic parameter, as a view tab</p> |
| GetContextItemType() | ObjectType | <p>Returns the ObjectType of the item in context within EA. A Context Item is defined as the item selected anywhere within the EA gui including:</p> <ul style="list-style-type: none"> • An Item selected in the Project View Tree • An Item selected in an open diagram • An Item selected in certain dialogs, such as the Attributes properties dialog <p>The supported ObjectTypes can be any one of the following values:</p> <ul style="list-style-type: none"> • otElement • otPackage • otDiagram • otAttribute • otMethod • otConnector |
| GetContextItem(Object Item) | ObjectType | <p>param: Item [Object - out]</p> <p>Sets Item as a pointer to the item in context within EA. as well as its corresponding ObjectType as a return value.</p> <p>For additional information about Context Items and the supported ObjectTypes refer to the GetContextItemType method.</p> |
| ActivateTab(string Name) | | <p>param: Name [String - in]</p> <p>Activates the open EA tabbed view with the matching Name.</p> |
| GetDiagramByGuid(string Guid) | Diagram | <p>param: Guid [String - in]</p> <p>Returns a pointer to a Diagram using the global reference ID (global ID). This is usually found using the Diagram GUID property of an element - and stored for later use when you wish to open a Diagram without using the collection GetAt() function.</p> |
| GetCurrentLoginUser(boolean GetGuid = false) | String | <p>If security is not enabled in the repository, an error is generated.</p> <p>If GetGuid is True, a Guid generated by EA representing the user is returned, otherwise the text as entered in System Users/User Details/Login is returned.</p> |
| ChangeLoginUser(string Name, string Password) | Boolean | <p>param: Name [String - in]</p> <p>param: Password [String - in]</p> <p>Set the currently logged on user to be that specified by the Name parameter, with the provided Password value. This logs the user into the repository when security is enabled. If security is not enabled an exception ("Security not enabled") is thrown.</p> |
| GetTreeSelectedObject() | Object | <p>The related method GetTreeSelectedItem() has an out parameter which is inaccessible by some scripting languages. As an alternative, this method provides the selected item through the return value.</p> |

| | | |
|---|---------|---|
| ShowProfileToolbox(string Technology, string Profile, boolean Show) | | <p>param: Technology [String - in] param: Profile [String - in] param: Show [Boolean - in]</p> <p>Shows/Hides the contents of the specified Technology or Profile in EA's Toolbox. To show/hide a profile in the Toolbox, specify the Profile's ID value in the Profile parameter and set the Technology parameter to a null string. To show/hide a technology in the Toolbox, specify the Technology's ID in the Technology parameter and set the Profile parameter to a null string.</p> |
| ActivatePerspective(string Perspective, long Options) | Boolean | <p>param: Perspective [String - in] param: Options [long - in]</p> <p>Activates the perspective, specified by its name in the Perspective parameter. The Options parameter is reserved for future use.</p> |
| AddPerspective(string Perspective, long Options) | Boolean | <p>param: Perspective [String - in] param: Options [long - in]</p> <p>Adds the perspective, specified by its name in the Perspective parameter, to the repository. The Options parameter is reserved for future use.</p> |
| DeletePerspective(string Perspective, long Options) | Boolean | <p>param: Perspective [String - in] param: Options [long - in]</p> <p>Deletes the perspective, specified by its name in the Perspective parameter. The Options parameter is reserved for future use.</p> |
| GetActivePerspective() | String | Returns the name of the currently active perspective. |
| HasPerspective(string Perspective) | String | <p>param: Perspective [String - in]</p> <p>Returns "1" or "0" depending on whether or not a perspective, specified by its name in the Perspective parameter, is available in the repository.</p> |
| IsTabOpen() | String | <p>param: TabName [String - in]</p> <p>Returns 2 to indicate that a tab is open and active (top-most), 1 indicates that it is open but not top-most, 0 to indicate that it is not visible at all.</p> <p>Note that TabName is case-sensitive.</p> |

| | | |
|---------------------------------|-----------|--|
| GetAttributeByGuid(string Guid) | Attribute | <p>param: Guid [String - in]</p> <p>Returns a pointer to an Attribute in the repository with matching Guid. This is usually found using the AttributeGUID property of a Attribute.</p> |
| GetMethodByGuid(string Guid) | Method | <p>param: Guid [String - in]</p> <p>Returns a pointer to a Method in the repository with matching Guid. This is usually found using the MethodGUID property of a Method.</p> |

| | | |
|--|---------|---|
| ShowInProjectView(Object Item) | | <p>param: Item [Object - in]</p> <p>Selects the object referenced by the supplied parameter in the Project View window. Accepted object types are Package, Element, Diagram, Attribute, and Method. An exception will be thrown if object is of an invalid type.</p> |
| AddDefinedSearches(string sXML) | | <p>param: sXML [string - in]</p> <p>Allows you to enter a set of defined searches that will last in EA for the life of the application. When EA loads again they will need to be inserted again by your Add-In. You can get this xml by exporting the searches from the model search dialog in EA - (Ctrl F then hit the advanced button).</p> |
| GetElementsByQuery(string QueryName, string SearchTerm) | | <p>param: QueryName[string - in] param: SearchTerm[string -in]</p> <p>Allows you to run a search in EA returning the result as a collection. QueryName will be the name of the search to run, for example 'Simple'. SearchTerm will be what you are looking for, For exmaple GetElementsByQuery('Simple','Class1'). Where results will contain elements with 'Class1' in the Name and Notes field.</p> |
| RunModelSearch(string 'QueryName', string 'SearchTerm', string 'SearchOptions', string 'SearchData') | | <p>param: sQueryName[string - in] param: sSearchTerm[string - in] param: sSearchOptions[string - in] param: sSearchData[string - in]</p> <p>Runs a search displaying the results in EA's search window. QueryName will be the name of the search to run, for example 'Simple'. SearchTerm will be what you are searching for. SearchOptions is currently not being used. The SearchData param allows you to supply a list of results in the form of xml which will be appended onto the result list in EA. See XML Format, this param is not required so pass in an empty string to run search as per normal.</p> |
| ExecutePackageBuildScript(long ScriptOptions, string PacakgeGuid) | | <p>param: ScriptOptions, [long - in] param: PacakgeGuid[string - in]</p> <p>Allows you to run the active package build script based on your current selection in the project browser. You can also run a script by passing in the package GUID. ScriptOptions can be any one of these values 1,2,3,4,5; 1 = Build; 2 = Test; 3 = Run; 4 = Create Workbench Instance; 5 = Debug</p> |
| ActivateToolbox(string Toolbox, long Options) | Boolean | <p>param: Toolbox [String - in] param: Options [long - in]</p> <p>Activates the toolbox page in the GUI, specified by its name in the Toolbox parameter. Returns true if the specified toolbox was successfully activated, false otherwise. The Options parameter is reserved for future use.</p> |

| | | |
|--|--|--|
| ImportPackageBuildScripts(string PackageGuid, string BuildScriptXML) | | param: PackageGuid [string - in] param: BuildScriptXML[string - in] Allows you to import build scripts into a package in EA. You can import your package build scripts by supplying a package GUID and the xml data which you can be exported from in EA (insert link here to probably that page above). |
| RefreshOpenDiagrams(Boolean FullReload) | | param: FullReload [Boolean - in] Refreshes the diagram contents for all diagrams open in EA. If FullReload is false the displayed contents of elements and connectors are refreshed in each diagram. If FullReload is true each of the diagrams are completely reloaded from the repository. |

15.1.2.4.2 Author

Author

public Class

An Author object represents a named model author. Accessed using the Repository Authors collection.

Associated table in .EAP file: t_authors

Author Attributes

| Attribute | Type | Notes |
|------------|-----------------------------|--|
| Name | String | Read/Write. Author name |
| Roles | String | Read/Write. Roles the author may play in this project |
| Notes | String | Read/Write. Notes about the author |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface |

Author Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current Author object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.3 Client

Client

public Class

A Client represents one or more people or organizations related to the project. Accessed using the Repository Clients collection.

Associated table in .EAP file: t_clients

Client Attributes

| Attribute | Type | Notes |
|------------------|-----------------------------|---|
| Name | <i>String</i> | Read/Write. Client name |
| Organization | <i>String</i> | Read/Write. Associated organization |
| Phone1 | <i>String</i> | Read/Write. Main phone number |
| Phone2 | <i>String</i> | Read/Write. Second phone number |
| Mobile | <i>String</i> | Read/Write. Mobile phone if available |
| Fax | <i>String</i> | Read/Write. Fax number |
| Email | <i>String</i> | Read/Write. Email address |
| Roles | <i>String</i> | Read/Write. Roles this client may play in project |
| Notes | <i>String</i> | Read/Write. Notes about client |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Client Methods

| Method | Type | Notes |
|-----------------|----------------|--|
| Update () | <i>Boolean</i> | Update the current Client object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.4 Collection

Collection

public Class

This is the main collection class used by all elements within the automation interface. It contains methods to iterate through the collection, refresh the collection and delete an item from the collection. It is important to realize that when AddNew is called, the item is not automatically added to the current collection. The typical steps are:

1. Call AddNew to add a new item
2. Modify the item as required
3. Call Update on the item to save it to the database
4. Call Refresh on the collection to include it in the current set

Delete is much the same - until Refresh is called, the collection will still contain a reference to the deleted

item - which should not be called.

For each - may be used to iterate through the collection for languages that support this type of construct.

Collection Attributes

| Attribute | Type | Notes |
|------------|-----------------------------|---|
| Count | <i>short</i> | Read only. The number of objects referenced by this list. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Collection Methods

| Method | Type | Notes |
|---------------------------|---------------|---|
| GetAt (short) | <i>Object</i> | param: index [short - in] Retrieves the array object using a numerical index. If the index is out of bounds, an error will occur. |
| Delete (short) | <i>void</i> | param: index [short - in] Delete the item at the selected reference |
| DeleteAt (Boolean, short) | <i>void</i> | param: refresh [Boolean - in] param: index [short - in] Delete the item at the selected index. Also provides an option to refresh the collection after the deletion. You should not refresh a collection while looping through it, as the count and item positions may alter. You should not access the item at the selected index once it is deleted or an exception will be thrown. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| GetByName (String) | <i>Object</i> | param: Name [String - in] Get an item in the current collection by Name. Only applies to the following collections: Models, Packages, Elements, Diagrams, TaggedValues. |
| Refresh () | <i>void</i> | Refresh the collection by re-querying the model and reloading the collection. Should be called after adding a new item or after deleting an item. |
| AddNew (String, String) | <i>Object</i> | param: Type [String - in] param: Name [String - in] Add a new item to the current collection. Note that the interface is the same for all collections - you must provide a Name and Type argument. What these are used for depend on the actual collection member. Also note that you must call Update() on the returned object to complete the AddNew. If Update() is not called the object is left in an indeterminate state. |

15.1.2.4.5 Datatype

Datatype

public Class

A Datatype is a named type that may be associated with attribute or method types. It typically is related to either code engineering or database modeling. Datatypes also indicate which language or database system they relate to. Accessed using the Repository Datatypes collection.

Associated table in .EAP file: t_datatypes

Datatype Attributes

| Attribute | Type | Notes |
|--------------|-----------------------------|--|
| Name | String | Read/Write. The datatype name (eg. "integer"). This will appear in the related drop down datatype lists where appropriate |
| Type | String | Read/Write. The type: may be "DDL" for database datatype or "Code" for language datatypes |
| Product | String | Read/Write. The datatype product - eg. Java, C++, Oracle. |
| Size | Long | Read/Write. The datatype size |
| MaxLen | Long | Read/Write. Maximum length (DDL only) |
| MaxPrec | Long | Read/Write. Maximum precision(DDL only) |
| MaxScale | Long | Read/Write. Maximum scale(DDL only) |
| DefaultLen | Long | Read/Write. Default length (DDL only) |
| DefaultPrec | Long | Read/Write. Default precision(DDL only) |
| DefaultScale | Long | Read/Write. Default scale(DDL only) |
| UserDefined | Long | Read/Write. Indicates if datatype is a user defined type or system generated. Datatypes distributed with EA are all system. Datatypes created in the Datatype dialog are marked 1 (true) |
| HasLength | String | Read/Write. Indicates datatype has a length component |
| GenericType | String | Read/Write. The associated generic type for this data type |
| DatatypeID | Long | Read/Write. Instance ID for this datatype within the current model. System maintained |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Datatype Methods

| Method | Type | Notes |
|-----------------|---------|--|
| Update () | Boolean | Update the current Datatype object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs |

15.1.2.4.6 EventProperties

EventProperties

An EventProperties object is passed to BroadcastFunctions to facilitate parameter passing.

EventProperties Attributes

| Attribute | Type | Notes |
|------------|----------------------------|---|
| Count | Long | Read only. Number of parameters being passed to this broadcast event. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

EventProperties Methods

| Method | Type | Notes |
|--------|-------------------------------|--|
| Get | EventProperty | <p>Read only. Parns: Index (Variant) Returns an EventProperty in the list, raising an error if Index is out of range.</p> <p>Index can either be a number representing a zero-based index into the array, or a string representing the name of the EventProperty.</p> <p>eg. Props.Get(3) or Props.Get("ObjectID")</p> |

15.1.2.4.7 EventProperty

EventProperty

EventProperty objects are always part of an [EventProperties](#) collection, and are passed to add-in methods responding to [broadcast events](#).

EventProperty Attributes

| Attribute | Type | Notes |
|-------------|----------------------------|---|
| Name | String | A string distinguishing this property from others in the list. |
| Value | Variant | A string, number or object reference representing the property value. |
| Description | String | Explanation of what this property represents. |
| ObjectType | ObjectType | Distinguishes objects referenced through Dispatch interface. |

See Also

- [Event Properties](#)

15.1.2.4.8 Package

Package

public Class

A Package object corresponds to a package element in the EA Project Browser. It is accessed either through the Repository Models collection (a Model is a special form of Package) or through the Package

Packages collection. Note that a Package has an Element object as an attribute - this corresponds to an EA Package element in the t_object table and is used to associate additional information (such as scenarios and constraints) with the logical package. To set additional information for a package, reference the Element object directly. Also note that if you add a Package to a Diagram, you should add an instance of the Element - not the Package itself - to the DiagramObjects collection for a Diagram.

Associated table in .EAP file: t_package

Package Attributes

| Attribute | Type | Notes |
|--------------|-------------------|---|
| Name | <i>String</i> | Read/Write. The Name of the package. |
| Packages | <i>Collection</i> | Read only. A collection of contained packages which may be walked through. |
| Elements | <i>Collection</i> | Read only. A collection of Elements which belong to this package. |
| Diagrams | <i>Collection</i> | Read only. A collection of Diagrams contained in this package. |
| Notes | <i>String</i> | Read/Write. Notes about this package. |
| IsNamespace | <i>Boolean</i> | Read/Write. True is package is a Namespace root. Use 0 and 1 to set False and True. |
| PackageID | <i>Long</i> | Read only. The local Package ID number. Valid only in this model file. |
| PackageGUID | <i>Variant</i> | Read only. The global Package ID. Valid across models. |
| ParentID | <i>Long</i> | Read/Write. The ID of the Package that is the Parent of this one. 0 indicates this package is a Model (ie. it has no parent). |
| Created | <i>Date</i> | Read/Write. Date package created. |
| Modified | <i>Date</i> | Read/Write. Date package last modified. |
| IsControlled | <i>Boolean</i> | Read/Write. Indicates if the package has been marked as 'Controlled'. |
| IsProtected | <i>Boolean</i> | Read/Write. Indicates if the package has been marked as Protected. |
| UseDTD | <i>Boolean</i> | Read/Write. Indicates if a DTD is to be used when exporting XML. |
| LogXML | <i>Boolean</i> | Read/Write. Indicates if XML export information is to be logged. |
| XMLPath | <i>String</i> | Read/Write. The path to the where XML will be saved when using controlled packages. |
| Version | <i>String</i> | Read/Write. The Version of the package. |
| LastLoadDate | <i>Date</i> | Read/Write. Date XML was last loaded for package. |
| LastSaveDate | <i>Date</i> | Read/Write. Date XML was last saved from package. |

| | | |
|------------|----------------|--|
| Flags | <i>String</i> | Read/Write. Extended information about package. |
| Owner | <i>String</i> | Read/Write. The package owner when using controlled packages. |
| CodePath | <i>String</i> | Read/Write. Path to where associated source code is found. |
| UMLVersion | <i>String</i> | Read/Write. UML Version for XML export purposes. |
| TreePos | <i>Long</i> | Read/Write. The relative position in the tree compared to other packages (use to sort packages). |
| IsModel | <i>Boolean</i> | Read only. Indicates if package is a Model or Package. |

| | | |
|---------------------|-----------------------------|--|
| Element | <i>Object</i> | Read only. The associated Element object. Use to set Element type information for a package ... including Stereotype, Complexity, Alias, Author, Constraints, Scenarios ... etc. |
| BatchSave | <i>long</i> | Read/Write. Boolean value to indicate whether package will be included in the batch XMI export list or not. |
| BatchLoad | <i>long</i> | Read/Write. Flag to indicate package will be batch loaded during batch import from controlled packages. Not yet implemented. |
| Connectors | <i>Collection</i> | Read only. Collection of connectors. |
| Alias | <i>string</i> | Read only. Alias. |
| IsVersionControlled | <i>boolean</i> | Read/Write. Whether or not this package is under version control. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Package Methods

| | | |
|-----------------------------|-------------------|--|
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | <i>Boolean</i> | Update the current Package object after modification or appending a new item. If false is returned, check the GetLastError function for more information. Note that a Package object also has an 'Element' component which must be taken into account. The Package object contains information about the hierarchy, contents etc. The Element attribute contains information about Stereotype, Constraints, Files & etc. - all the attributes of a typical Element. |
| FindObject(String DottedID) | <i>LPDISPATCH</i> | Param: DottedID [String in] Returns a Package, Element, Attribute or Operation matching the parameter DottedID. The string DottedID is in the form object.object.object where object is replaced by the name of a package, element attribute or operation. Examples of DottedIDs include MyNamespace.Class1, CStudent.m_Name, MathClass.DoubleIt(int) If the DottedID is not found, an error is returned: "Can't find matching object" |

| | | |
|---|-------------|---|
| VersionControlAdd (String ConfigGuid, String XMLFile, Bool KeepCheckedOut) | <i>void</i> | <p>param: ConfigGuid [String - in] Name corresponding to the Unique ID of the Version Control configuration to use.</p> <p>param: XMLFile [String - in] Name of the XML file to use for this package. This filename is relative to the Working Copy folder specified for the Config.</p> <p>param: KeepCheckedOut [Boolean - in] Specify True to add to version control and keep package checked-out.</p> <p>Places the package under version control, using the specified Version Control Configuration and the specified XML filename. Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</p> |
| VersionControlRemove () | <i>void</i> | Removes version control from the package. Throws an exception if the operation fails. Use GetLastError() to retrieve error information. |
| VersionControlCheckout (String Comment) | <i>void</i> | <p>param: Comment [String - in] Log message that is added to the version controlled file's history (where applicable).</p> <p>Perform checkout of the version controlled package. Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</p> |
| VersionControlCheckin (String Comment) | <i>void</i> | <p>param: Comment [String - in] Log message that is added to the version controlled file's history (where applicable).</p> <p>Perform checkin of the version controlled package. Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</p> |

| | | |
|--------------------------------|-------------|--|
| VersionControlGetStatus () | <i>long</i> | Returns the version control status of the package. Return value maps to the following enumerated type; enum EnumCheckOutStatus { csUncontrolled, csCheckedIn, csCheckedOutToThisUser, csReadOnlyVersion, csCheckedOutToAnotherUser, csOfflineCheckedOutToThisUser, csOfflineNotCheckedOutToThisUser, csDeleted } csUncontrolled - Either unable to communicate with the version control provider associated with the package or the package file is unknown to the provider. csReadOnlyVersion - Package is marked as read-only. An earlier revision of the package has been retrieved from Version Control. csOfflineCheckedOutToThisUser - Indicates that the package was "checked out" by this user whilst disconnected from version control. csOfflineNotCheckedOutToThisUser - Indicates that EA can not currently connect to the VC config and the package was not previously checked out to this user. csDeleted - The package file has been deleted from version control. Throws an exception if the operation fails. Use GetLastError() to retrieve error information. |
|--------------------------------|-------------|--|

15.1.2.4.9 ProjectIssues

ProjectIssues

public Class

A system level Issue. Indicates a problem or risk associated with the system as a whole. Accessed using the Repository Issues collection.

Associated table in .EAP file: t_issues

ProjectIssues Attributes

| Attribute | Type | Notes |
|-----------|------|-------|
|-----------|------|-------|

| | | |
|--------------|----------------------------|--|
| Category | String | Read/Write. The category this issue belongs to. |
| Name | String | Read/Write. Issue name (ie. the Issue itself). |
| Date | Date | Read/Write. Date created. |
| Owner | String | Read/Write. Owner of issue. |
| Status | String | Read/Write. Current issue status. |
| Notes | String | Read/Write. Associated description of issue. |
| Resolver | String | Read/Write. Person resolving issue. |
| DateResolved | Date | Read/Write. Date issue resolved. |
| Resolution | String | Read/Write. Description of resolution. |
| Priority | String | Read/Write. Issue priority ... generally should use Low, Medium or High. |
| Severity | String | Read/Write. Issue severity. Should be marked as Low, Medium or High. |
| IssueID | Long | Read only. The ID of this issue. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

ProjectIssues Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current Issue object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.10 ProjectResource

ProjectResource

public Class

A Project Resource is a named person who is available to work on the current project in any capacity. Accessed using the Repository Resources collection.

Associated table in .EAP file: t_resources

ProjectResource Attributes

| Attribute | Type | Notes |
|--------------|--------------------|--|
| Name | String | Name of resource. |
| Organization | package: String | Organization resource associated with. |
| Phone1 | Variant | Main phone. |
| Phone2 | Variant | Alternate phone. |
| Mobile | Variant | Mobile number if available. |
| Fax | String | Fax number. |
| Email | String | Email address. |
| Roles | String | The roles this resource can play in the current project. |
| Notes | String | A description if appropriate. |

| | | |
|------------|----------------------------|---|
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |
|------------|----------------------------|---|

ProjectResource Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Resource object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.11 Property Type

PropertyType

public Class

A PropertyType object represents a defined property that may be applied to UML elements as a tagged value. Accessed using the Repository PropertyTypes collection. Each PropertyType corresponds to one of the predefined tagged values defined for the model.

Associated table in .EAP file: t_propertytypes

Author Attributes

| Attribute | Type | Notes |
|-------------|----------------------------|--|
| Tag | <i>String</i> | Read/Write. Name of the property (Tag Name) |
| Description | <i>String</i> | Read/Write. Short description for the property |
| Detail | <i>String</i> | Read/Write. Configuration information for the property |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface |

Author Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current PropertyType object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.12 Reference

Reference

public Class

This Interface provides access to the various lookup tables within EA. Use the Repository GetReferenceList() method to get a handle to a list. Valid lists are:

- Diagram
- Element
- Constraint
- Requirement
- Connector
- Status
- Cardinality
- Effort
- Metric
- Scenario
- Status
- Test

Reference Attributes

| Attribute | Type | Notes |
|------------|--|---|
| Count | Short | Count of items in the list. |
| Type | String | The list type (eg. Diagram Types). |
| ObjectType | ObjectTyp e | Read only. Distinguishes objects referenced through Dispatch interface. |

Reference Methods

| Method | Type | Notes |
|-----------------|--------|---|
| GetAt (Short) | String | param: index [Short - in] The index of the item to retrieve from the list. Get the item at the specified index. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Refresh () | short | Refresh the current list and return the count of items. |

15.1.2.4.13 Stereotype

Stereotype

public Class

The Stereotype element corresponds to a UML Stereotype, which is an extension mechanism for varying the behavior and type of a model element. Use the repository Stereotypes collection to add new elements and delete existing ones.

Associated table in .EAP file: t_stereotypes

Stereotype Attributes

| Attribute | Type | Notes |
|------------------|----------------------------|--|
| Name | <i>String</i> | Read/Write. The Stereotype. Appears in the Stereotype drop list for elements that match the AppliesTo attribute. |
| AppliesTo | <i>String</i> | Read/Write. A reference to the Stereotype Base Class ... ie. which element it applies to. |
| Notes | <i>String</i> | Read/Write. Notes about the Stereotype. |
| MetafileLoadPath | <i>String</i> | Read/Write. Path to an associated metafile. The automation interface does not yet support loading metafiles - to do this you must use the Stereotype dialog in EA. |
| Style | <i>String</i> | Read/Write. Additional style specifier for stereotype. |
| StereotypeGUID | <i>String</i> | Read/Write. Unique identifier for stereotype, generally set and maintained by EA. |
| VisualType | <i>String</i> | Read/Write. Indicates an inbuilt visual style associated with a stereotype. Not currently implemented. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

Stereotype Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Stereotype object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.14 Task

Task

public Class

A Task is an entry in the System ToDo list. Accessed using the Repository Tasks collection.

Associated table in .EAP file: t_tasks

Task Attributes

| Attribute | Type | Notes |
|-----------|----------------|---|
| TaskID | <i>Long</i> | Read only. Local ID of task. |
| Name | <i>Variant</i> | Read/Write. Task name. |
| Notes | <i>Variant</i> | Read/Write. Description of the task. |
| Priority | <i>String</i> | Read/Write. Priority associated with this task. |
| Status | <i>Variant</i> | Read/Write. Current task status. |
| Owner | <i>String</i> | Read/Write. The task owner. |
| StartDate | <i>Date</i> | Read/Write. Date task is to start. |
| EndDate | <i>Date</i> | Read/Write. Date task scheduled to finish. |
| Phase | <i>String</i> | Read/Write. The phase of the project the task relates to. |

| | | |
|------------|---------------------------|---|
| History | <i>String</i> | Read/Write. Memo field to hold task history, notes etc. |
| Percent | <i>Long</i> | Read/Write. Percent the task is complete. |
| TotalTime | <i>Long</i> | Read/Write. The total expected time the task will run - may be in hours or days or some other unit. |
| ActualTime | <i>Long</i> | Read/Write. Time already expended on task. May be hours or days or other units. |
| AssignedTo | <i>String</i> | Read/Write. Person this task is assigned to - ie. the responsible resource. |
| Type | <i>String</i> | Read/Write. Sets or returns string representing the type. |
| ObjectType | ObjectTye | Read only. Distinguishes objects referenced through Dispatch interface |

Task Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Task object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.4.15 Term

Term

public Class

A Term object represents one entry in the system glossary. Accessed using the Terms collection of the repository.

Associated table in .EAP file: t_glossary

Term Attributes

| Attribute | Type | Notes |
|------------|---------------------------|---|
| Term | <i>String</i> | Read/Write. The glossary item name. |
| Type | <i>String</i> | Read/Write. The type this term applies to (eg. business or technical). |
| Meaning | <i>String</i> | Read/Write. The description of the term - its meaning. |
| TermID | <i>Long</i> | Read only. A local ID number to identify the term in the model. |
| ObjectType | ObjectTye | Read only. Distinguishes objects referenced through Dispatch interface. |

Term Methods

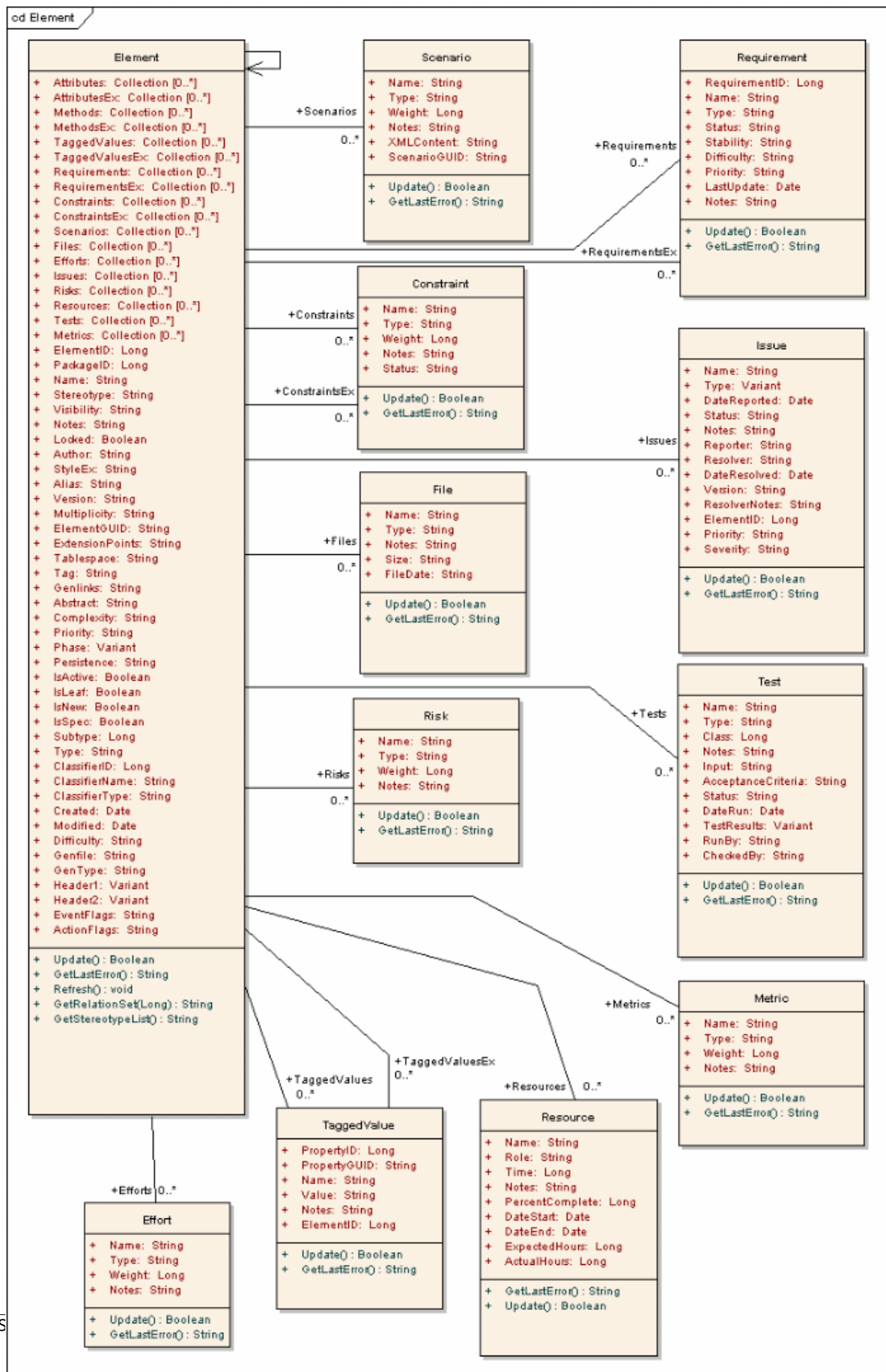
| Method | Type | Notes |
|--------|------|-------|
|--------|------|-------|

| | | |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Term object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5 Element

This diagram illustrates the relationships between an Element and its associated extended information. The related information is accessed through the collections owned by the Element (eg. Scenarios and Tests). It also includes a full description of the Element object - the basic model structural unit.

Figure 4 : Element



Element**public Package**

The Element package contains information about an Element and its associated extended properties such as testing and project management information. An Element is the basic item in an EA model. Classes, Use Cases, Components & etc. are all different types of UML Elements.

15.1.2.5.1 Constraint**Constraint****public Class**

A Constraint is a condition imposed on an Element. Constraints are accessed through the Element Constraints collection

Associated table in .EAP file: t_objectconstraints

Constraint Attributes

| Attribute | Type | Notes |
|------------|----------------------------|---|
| Name | String | Read/Write. The name of the constraint (i.e. the constraint). |
| Type | String | Read/Write. Constraint type. |
| Weight | Long | Read/Write. A weighting factor. |
| Notes | String | Read/Write. Notes about the constraint. |
| Status | String | Read/Write. Current status. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |
| ParentID | Long | Read only. The ElementID of the element to which this constraint applies. |

Constraint Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current Constraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.2 Effort**Effort****public Class**

Effort Attributes

| Attribute | Type | Notes |
|------------|-----------------------------|---|
| Name | <i>String</i> | Read/Write. The name of the effort (i.e. the effort). |
| Type | <i>String</i> | Read/Write. Effort type. |
| Weight | <i>Long</i> | Read/Write. A weighting factor. |
| Notes | <i>String</i> | Read/Write. Notes about the constraint. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Effort Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Saves the Effort to the Model |
| GetLastError () | <i>String</i> | This function is rarely used since an exception will be thrown when an error occurs. Returns a string value describing the most recent error that occurred in relation to this object. |

15.1.2.5.3 ElementElementpublic Class

An Element is the main modeling unit. It corresponds to Class, Use Case, Node, Component & etc. You create new Elements by adding to the Package Elements collection. Once you have created an element, you can add it to the DiagramObjects collection of a diagram to include it in a diagram.

Elements also have a collection of Connectors. Each entry in this collection indicates a relationship to another Element.

There are also some extended collections for managing additional information about the Element - including TaggedValues, Issues, Constraints, Requirement & etc.

Associated table in .EAP file: t_object

Element Attributes

| Attribute | Type | Notes |
|----------------|-------------------|--|
| Attributes | <i>Collection</i> | Read only. Collection of Attribute objects for current element. Use the AddNew and Delete function to manage attributes. |
| AttributesEx | <i>Collection</i> | Read only. Collection of Attribute objects belonging to the current element and its parent elements. |
| Methods | <i>Collection</i> | Read only. Collection of Method objects for current element. |
| MethodsEx | <i>Collection</i> | Read only. Collection of Method objects belonging to the current element and its parent elements. |
| TaggedValues | <i>Collection</i> | Read only. Collection of TaggedValue objects. |
| TaggedValuesEx | <i>Collection</i> | Read only. Collection of TaggedValue objects belonging to the current element and its parent elements. |
| Requirements | <i>Collection</i> | Read only. Collection of Requirement objects. |

| | | |
|----------------|-------------------|--|
| RequirementsEx | <i>Collection</i> | Read only. Collection of Requirement objects belonging to the current element and its parent elements. |
| Constraints | <i>Collection</i> | Read only. Collection of Constraint objects. |
| ConstraintsEx | <i>Collection</i> | Read only. Collection of Constraint objects belonging to the current element and its parent elements. |
| Scenarios | <i>Collection</i> | Read only. Collection of Scenario objects for current element |
| Files | <i>Collection</i> | Read only. Collection of File objects. |
| Efforts | <i>Collection</i> | Read only. Collection of Effort objects. |
| Issues | <i>Collection</i> | Read only. Collection of Issue objects. |
| Risks | <i>Collection</i> | Read only. Collection of Risk objects. |
| Resources | <i>Collection</i> | Read only. Collection of Resource objects for current element. |
| Tests | <i>Collection</i> | Read only. Collection of Test objects for current element. |
| Metrics | <i>Collection</i> | Read only. Collection of Metric elements for current element. |
| ElementID | <i>long</i> | Read only. The local ID of the Element. Valid for this file only. |
| PackageID | <i>long</i> | Read/Write. A local ID for the package containing this element. |
| Name | <i>String</i> | Read/Write. The element name - should be unique within the current package. |
| Stereotype | <i>String</i> | Read/Write. The element name - should be unique within the current package. |
| Visibility | <i>String</i> | Read/Write. The Scope of this element within the current package. Valid values are: Public, Private, Protected or Package. |
| Notes | <i>String</i> | Read/Write. Further descriptive text about Element. |
| Locked | <i>Boolean</i> | Read/Write. Indicates if Element has been locked against further change. |
| Author | <i>String</i> | Read/Write. The Element Author (see the Repository: Authors list for more details). |
| StyleEx | <i>String</i> | Read/Write: Advanced style settings. Not currently used. |

| | | |
|-----------------|---------------|--|
| Alias | <i>String</i> | Read/Write. An optional Alias for this element. |
| Version | <i>String</i> | Read/Write. The Version of the Element. |
| Multiplicity | <i>String</i> | Read/Write. Multiplicity value for this Element. |
| ElementGUID | <i>String</i> | Read only. A globally unique ID for this Element - unique across all model files. If you need to set this value manually, you should only do so when the element is first created - and make sure you format the GUID exactly as EA expects. |
| ExtensionPoints | <i>String</i> | Read/Write. Optional extension points for a Use Case as a comma-separated list. |
| Tablespace | <i>String</i> | Read/Write. Associated tablespace for a Table element. |
| Tag | <i>String</i> | Read/Write. Optional Tag value for additional user defined information and searching. |
| Genlinks | <i>String</i> | Read/Write. Links to other classes discovered at code reversing time - Parents and Implements links only |
| Abstract | <i>String</i> | Read/Write. Indicates if Element is Abstract (1) or Concrete (0) |
| Complexity | <i>String</i> | Read/Write. A Complexity value indicating how difficult the Element is. May be used for metric reporting and estimation. Valid values are: 1 for Easy, 2 for Medium, 3 for Difficult |

| | | |
|-------------|----------------|--|
| Priority | <i>String</i> | Read/Write. The priority of this element as compared to other project elements. Only applies to Requirement, Change and Issue types, otherwise ignored. Valid values are: "Low", "Medium" and "High" |
| Phase | <i>String</i> | Read/Write. Phase this element scheduled to be constructed in. Any string value |
| Persistence | <i>String</i> | Read/Write. The persistence associated with this element... may be "Persistent" or "Transient" |
| IsActive | <i>Boolean</i> | Read/Write. Boolean value indicating element is active or not. 1 = true, 0 = false |
| IsLeaf | <i>Boolean</i> | Read/Write. Boolean value indicating element is leaf node or not. 1 = true, 0 = false |
| IsNew | <i>Boolean</i> | Read/Write. Boolean value indicating element is new or not. 1 = true, 0 = false |
| IsSpec | <i>Boolean</i> | Read/Write. Boolean value indicating element is specification or not. 1 = true, 0 = false |
| Type | <i>String</i> | Read/Write. The Element type (eg. Class, Component etc) - Note that type is case sensitive inside EA and should be provided with an initial capital (proper case). Valid types are: Action, Activity, ActivityPartition, ActivityRegion, Actor, Artifact, Association, Boundary, Change, Class, Collaboration, Component, Constraint, Decision, DeploymentSpecification, DiagramFrame, EmbeddedElement, Entity, EntryPoint, Event, ExceptionHandler, ExitPoint, ExpansionNode, ExpansionRegion, GUIElement, InteractionFragment, InteractionOccurrence, InteractionState, Interface, InterruptibleActivityRegion, Issue, Node, Note, Object, Package, Parameter, Part, Port, ProvidedInterface, Report, RequiredInterface, Requirement, Screen, Sequence, State, StateNode, Synchronization, Text, TimeLine, UMLDiagram, UseCase |

| | | |
|----------------|---------------|---|
| Subtype | <i>Long</i> | Read/Write. A numeric subtype which varies the type of the main element: <ul style="list-style-type: none"> • For Event, 0 = Receiver, 1 = Sender • For Class, 1 = Parameterised, 2 = Instantiated, 3 = Both, 0 = Neither • For Note, 1 = Note linked to connector, 2 = Constraint linked to connector • For StateNode, 100 = ActivityInitial, 101 = ActivityFinal • For Activity, 0 = Activity, 8 = StructuredActivity • For Synchronization, 0 = Horizontal, 1 = Vertical |
| ClassifierID | <i>Long</i> | Read/Write. Local ID of a Classifier associated with this Element - that is the base type. Only valid for instance type elements (eg. Object) |
| ClassifierName | <i>String</i> | Read/Write. Name of associated Classifier (if any) |
| ClassifierType | <i>String</i> | Read only. Type of associated classifier. |
| Created | <i>Date</i> | Read/Write. Date element created |
| Modified | <i>Date</i> | Read/Write. Date element last Modified |
| Difficulty | <i>String</i> | Read/Write. A difficulty level associated with this element for estimation/metrics - only useable for Requirement, Change and Issue element types, otherwise ignored. Valid values are: "Low", "Medium", "High" |
| Genfile | <i>String</i> | Read/Write. The file associated with this element for code generation and synchronization purposes. May include macro expansion tags for local conversion to full path. |

| | | |
|--------------|-------------------|--|
| GenType | <i>String</i> | Read/Write. The code generation type - eg. Java, C++, C#, VBNet, Visual Basic, Delphi. |
| Header1 | <i>Variant</i> | Read/Write. A user defined string for inclusion as header in source files generated. |
| Header2 | <i>Variant</i> | Read/Write. Same as for Header1 - but used in CPP source file |
| EventFlags | <i>String</i> | Read/Write. A structure to hold a variety of flags to do with signals, events etc. |
| ActionFlags | <i>String</i> | Read/Write. A structure to hold flags concerned with Action semantics. |
| Elements | <i>Collection</i> | The child elements of this element. |
| Diagrams | <i>Collection</i> | The child diagrams of this element. |
| ParentID | <i>Long</i> | Read/Write. Can be used to set or retrieve the If this element is a child of another, the ElementID of the other element. If not, returns 0. |
| Connectors | <i>Collection</i> | Read only. Returns a collection containing the connectors to other elements. |
| ClassifierID | <i>Long</i> | Read/Write. Sets or gets the ElementID of the Classifier. |
| Status | <i>String</i> | Read/Write. Sets or gets the status eg. "Proposed", "Approved" etc. |

| | | |
|------------------|-----------------------------------|---|
| TreePos | <i>Long</i> | Read/Write. Sets or gets the tree position. |
| Elements | <i>Collection</i> | Read only. Returns a collection of sub-elements attached to this element as seen in the tree view. |
| Diagrams | <i>Collection</i> | Read only. Returns a collection of sub-diagrams attached to this element as seen in the tree view. |
| ObjectType | <i>ObjectType</i> | Read only. Distinguishes objects referenced through Dispatch interface. |
| Partitions | <i>Collection</i> | Read Only. List of logical partitions into which an element may be divided. Only valid for elements which support partitions, such as Activities and States. |
| CustomProperties | <i>Collection</i> | Read only. List of "Advanced Properties" for an element. The collection of advanced properties will change depending on element type – for example an Action and an Activity have different Advanced Properties. Currently only editable from the user interface. |
| StateTransitions | <i>Collection</i> | Read only. List of State Transitions that an element may support. Applies in particular to Timing Elements. |
| EmbeddedElements | <i>Collection</i> | Read only. List of elements which are embedded into this element. Includes Ports, Parts, Pins, Parameter Sets & etc. |
| BaseClasses | <i>Collection</i> | Read only. List of Base Classes for this element presented as a collection for convenience. |
| Realizes | <i>Collection</i> | Read only. List of Interfaces realized by this element for convenience. |
| MiscData | <i>String</i> | Read only. This low-level property provides information about the contents of the PDATAx fields. These database fields are not documented and developers will need to gain understanding of these fields through their own endeavors to use this property. MiscData is zero based so MiscData(0) corresponds to PDATA1, MiscData(1) to PDATA2 etc. |
| StereotypeEx | <i>String</i> | Read/Write. Returns all the applied stereotypes of the element in a comma-separated list |
| PropertyType | <i>Long</i> | Read/Write. The GUID of the type which defines either a Port or a Part. |

| | | |
|------------|----------------------------|--|
| Properties | Properties | Returns a list of specialized properties which apply to the element that may not be available using the automation model. The properties are purposely undocumented because of their obscure nature and because they are subject to change as progressive enhancements are made to them. |
| MetaType | <i>String</i> | Read only: The Element's domain-specific meta type, as defined by an applied stereotype from an MDG Technology. |

Element Methods

| Method | Type | Notes |
|---|----------------|--|
| Update () | <i>Boolean</i> | Update the current Element object after modification or appending a new item. If false is returned, check the <code>GetLastError</code> function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Refresh () | <i>void</i> | Refresh the Element features in the Project Browser tree. Usually called after adding or deleting attributes or methods - when the user interface is required to be updated as well. |
| GetRelationSet (EnumRelationSetType) | <i>String</i> | Returns a string containing a comma-separated list of ElementIDs of related elements based on the given type. See EnumRelationSetType . |
| GetStereotypeList () | <i>String</i> | Returns a comma-separated list of stereotypes allied to this element. |
| SetAppearance (long Scope, long Item, long Value) | <i>Long</i> | Sets the visual appearance of Element. Scope: Scope of appearance set to modify 0 – Local (Diagram-local appearance) 1 – Base (Default appearance across entire model) Item: Appearance item to modify 0 – Background color 1 – Font Color 2 – Border Color 3 – Border Width Value: Value to set appearance to. |

15.1.2.5.4 File

File

public Class

A File represents an associated File for an Element. It is accessed through the Element Files collection

Associated table in .EAP file: `t_objectfiles`

File Attributes

| Attribute | Type | Notes |
|-----------|---------------|---|
| Name | <i>String</i> | Read/Write. The file name - may be a logical file or a reference to a web address (using <code>http://</code>) |
| Type | <i>String</i> | Read/Write. File type. |
| Notes | <i>String</i> | Read/Write. Notes about the file. |

| | | |
|------------|-----------------------------|---|
| Size | <i>String</i> | Read/Write. The file size. |
| FileDate | <i>String</i> | Read/Write. The file date when entry is created. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

File Methods

| Method | Type | Notes |
|-----------------|----------------|--|
| Update () | <i>Boolean</i> | Update the current File object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.5 Issue (Maintenance)

Issue

public Class

An Issue is either a "Change" or a "Defect" and is associated with the containing Element - and accessed through the Issues collection of an Element.

Associated table in .EAP file: t_objectproblems

Issue Attributes

| Attribute | Type | Notes |
|---------------|-----------------------------|---|
| Name | <i>String</i> | Read/Write. The Issue name - ie. the Issue itself. |
| Type | <i>Variant</i> | Read/Write. Issue type - may be "Defect" or "Change", "Issue" and "ToDo". |
| DateReported | <i>Date</i> | Read/Write. Date issue reported. |
| Status | <i>String</i> | Read/Write. The current status of the issue. |
| Notes | <i>String</i> | Read/Write. Issue Description. |
| Reporter | <i>String</i> | Read/Write. Person reporting issue. |
| Resolver | <i>String</i> | Read/Write. Person resolving issue. |
| DateResolved | <i>Date</i> | Read/Write. Date issue resolved. |
| Version | <i>String</i> | Read/Write. Version associated with issue. Note that this method is only available through the dispatch interface. eg. Object ob = Issue; Print ob.Version; |
| ResolverNotes | <i>String</i> | Read/Write. Notes entered by resolver about resolution. |
| ElementID | <i>Long</i> | Read/Write. ID of element associated with this issue. |
| Priority | <i>String</i> | Read/Write. Issue priority ... generally should use Low, Medium and High. |
| Severity | <i>String</i> | Read/Write. Issue severity. Should be marked as Low, Medium or High. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Issue Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Issue object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.6 Metric

Metric

public Class

A Metric is a named item with a weighting which may be associated with an element for purposes of building metrics about the model. Accessed through the Element Metrics collection

Associated table in .EAP file: t_objectmetrics

Metric Attributes

| Attribute | Type | Notes |
|------------|-----------------------------------|---|
| Name | <i>String</i> | Read/Write. The name of the Metric. |
| Type | <i>String</i> | Read/Write. The Metric type. |
| Weight | <i>Long</i> | Read/Write. A user defined weighting for estimation or metric purposes. |
| Notes | <i>String</i> | Read/Write. Notes about this metric. |
| ObjectType | <i>ObjectType</i> | Read only. Distinguishes objects referenced through Dispatch interface. |

Metric Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Metric object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.7 Requirement

Requirement

public Class

An Element Requirement object holds information about the responsibilities of an element in the context of the model. Accessed using the Element Requirements collection

Associated table in .EAP file: t_objectrequires

Requirement Attributes

| Attribute | Type | Notes |
|---------------|-----------------------------|--|
| RequirementID | Long | Read only. A local ID for this requirement. |
| Name | String | Read/Write. The requirement itself. |
| Type | String | Read/Write. Requirement type. |
| Status | String | Read/Write. Current status of the requirement. |
| Stability | String | Read/Write. Estimated stability of the requirement. |
| Difficulty | String | Read/Write. Estimated difficulty to implement. |
| Priority | String | Read/Write. Assigned priority of the requirement. |
| LastUpdate | Date | Read/Write. Date requirement last updated. |
| Notes | String | Read/Write. Further notes about requirement. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |
| ParentID | Long | Read only. The ElementID of the element to which this requirement applies. |

Requirement Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current Requirement object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.8 ResourceResourcepublic Class

An element resource is a named person/task pair with timing constraints and percent complete indicators. Use this to manage the work associated with delivering an Element.

Associated table in .EAP file: t_objectresources

Resource Attributes

| Attribute | Type | Notes |
|-----------------|--------|---|
| Name | String | Read/Write. Name of resource (eg. person's name). |
| Role | String | Read/Write. Role they will play in implementing Element. |
| Time | Long | Read/Write. Time expected - numeric indicating number of days. |
| Notes | String | Read/Write. Descriptive notes. |
| PercentComplete | Long | Read/Write. Current percent complete figure. |
| DateStart | Date | Read/Write. Date to start work. |
| DateEnd | Date | Read/Write. Expected end date. |
| ExpectedHours | Long | Read/Write. The total expected time the task will run - may be in hours or days or some other unit. |

| | | |
|-------------|--|---|
| ActualHours | <i>Long</i> | Read/Write. Time already expended on task. May be hours or days or other units. |
| History | <i>String</i> | Read/Write. Gets or sets history text. |
| ObjectType | ObjectTyp e | Read only. Distinguishes objects referenced through Dispatch interface. |

Resource Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | <i>Boolean</i> | Update the current Resource object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |

15.1.2.5.9 Risk

Risk

public Class

A Risk object represents a named risk associated with an element and may be used for project management purposes. Accessed through the Element Risks collection.

Associated table in .EAP file: t_objectrisks

Risk Attributes

| Attribute | Type | Notes |
|------------|--|---|
| Name | <i>String</i> | Read/Write. The risk. |
| Type | <i>String</i> | Read/Write. The risk type associated with this element. |
| Weight | <i>Long</i> | Read/Write. A weighting for estimation or metric purposes. |
| Notes | <i>String</i> | Read/Write. Further notes describing the risk. |
| ObjectType | ObjectTyp e | Read only. Distinguishes objects referenced through Dispatch interface. |

Risk Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Risk object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.10 Scenario**Scenario****public Class**

A Scenario corresponds to a Collaboration or Use Case instance. Each scenario is a path of execution through the logic of a Use Case. Scenarios may be added to using the Element Scenarios collection.

Associated table in .EAP file: t_objectscenarios

Scenario Attributes

| Attribute | Type | Notes |
|--------------|----------------------------|---|
| Name | String | Read/Write. The Scenario name. |
| Type | String | Read/Write. The Scenario type (eg. "Basic Path"). |
| Weight | Long | Read/Write. Currently used to position scenarios in the scenario list (ie. List Position). |
| Notes | String | Read/Write. Description of the Scenario. Usually contains the steps to execute the scenario. |
| XMLContent | String | Read/Write. A structured field which may contain scenario details in XML format. To be implemented in 2003. |
| ScenarioGUID | String | Read/Write. A unique ID for the scenario. Used to identify the scenario unambiguously within a model. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

Scenario Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current Scenario object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.11 TaggedValue**TaggedValue****public Class**

A TaggedValue is a named property and value associated with an element and is accessed through the TaggedValues collection

Associated table in .EAP file: t_objectproperties

TaggedValue Attributes

| Attribute | Type | Notes |
|--------------|--------|---|
| PropertyID | Long | Read only. A local ID for the property. |
| PropertyGUID | String | Read/Write. A global ID for the property. |

| | | |
|------------|-----------------------------|---|
| Name | <i>String</i> | Read/Write. Name of the property (Tag). |
| Value | <i>String</i> | Read/Write. The value assigned in this instance. |
| Notes | <i>String</i> | Read/Write. Further descriptive notes. |
| ElementID | <i>Long</i> | Read/Write. The local ID of the associated element. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

TaggedValue Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current TaggedValue object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.5.12 Test

Test

public Class

A Test is a single Test Case applied to an Element. Tests are added and accessed through the Element Tests collection.

Associated table in .EAP file: t_objecttests

Test Attributes

| Attribute | Type | Notes |
|--------------------|-----------------------------|---|
| Name | <i>String</i> | Read/Write. The Test Name |
| Type | <i>String</i> | Read/Write. The Test Type - eg. Load, Regression etc. |
| Class | <i>Long</i> | Read/Write. The Test Class: 1 = Unit Test, 2 = Integration Test, 3 = System test, 4 = Acceptance Test, 5 = Scenario Test. |
| Notes | <i>String</i> | Read/Write. Detailed notes about test to be carried out. |
| Input | <i>String</i> | Read/Write. Input data. |
| AcceptanceCriteria | <i>String</i> | Read/Write. The acceptance criteria for successful execution. |
| Status | <i>String</i> | Read/Write. Current status of test. |
| DateRun | <i>Date</i> | Read/Write. Date last run. |
| TestResults | <i>Variant</i> | Read/Write. Results of test. |
| RunBy | <i>String</i> | Read/Write. Person conducting test. |
| CheckedBy | <i>String</i> | Read/Write. Results confirmed by. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Test Methods

| Method | Type | Notes |
|--------|------|-------|
|--------|------|-------|

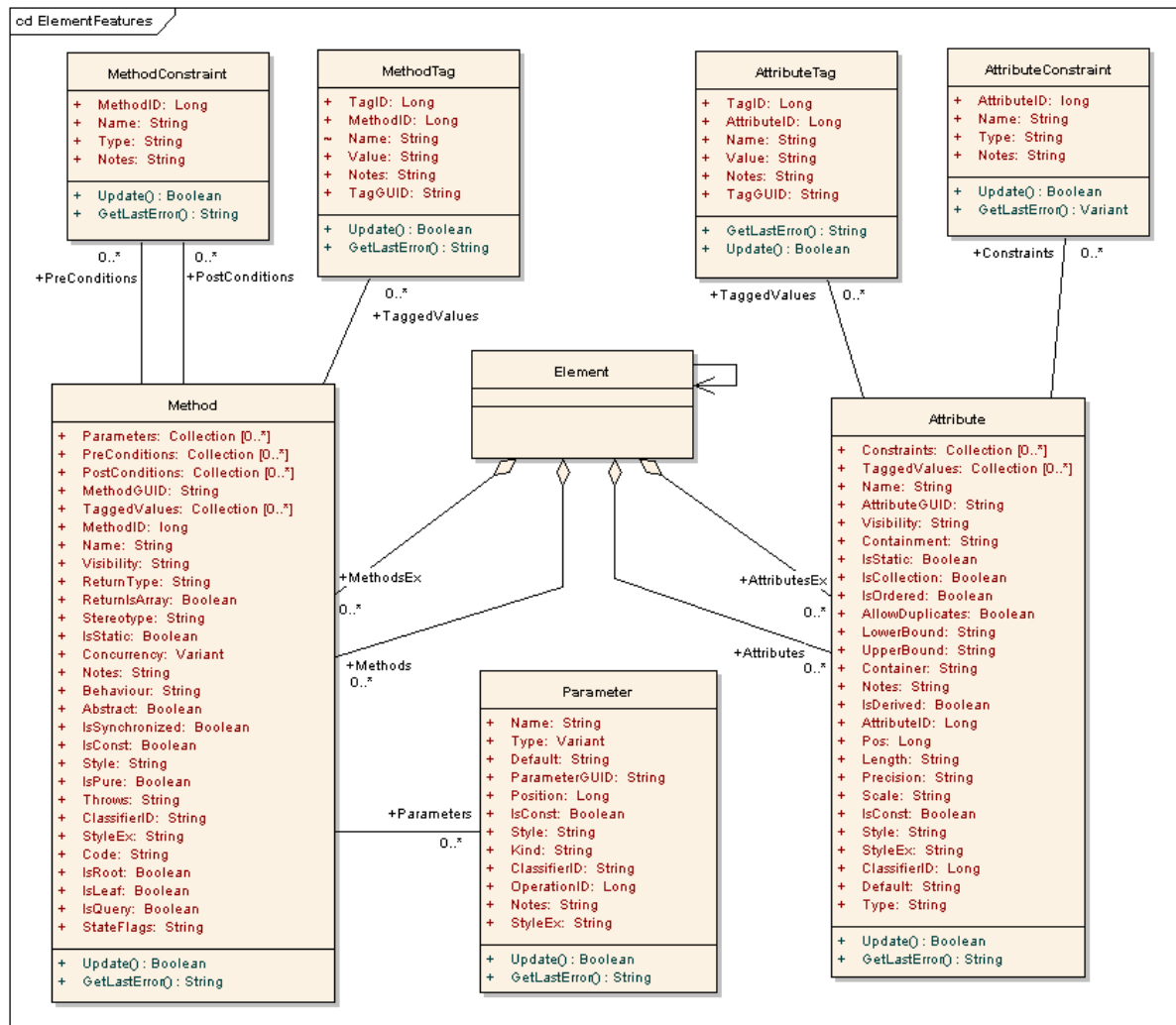
| | | |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Test object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6 Element Features

This diagram illustrates the components associated with element features. These include Attributes and Methods - and the associated constraints and tagged values related to them.

It also includes the Parameter object which define the arguments associated with an operation (method).

Figure 5 : ElementFeatures



ElementFeatures

public Package

The Element Features package contains descriptions of the model interfaces that allow access to Operations and Attributes - and their associated tagged values and constraints.

15.1.2.6.1 Attribute**Attribute**public Class

An Attribute corresponds to a UML Attribute. It contains further collections for constraints and tagged values. Attributes are accessed from the Element Attributes collection.

Associated table in .EAP file: t_attribute

Attribute Attributes

| Attribute | Type | Notes |
|-----------------|-------------------|---|
| Constraints | <i>Collection</i> | Read only. A collection of AttributeConstraints. Used to access and manage constraints associated with this Attribute. |
| TaggedValues | <i>Collection</i> | Read only. A collection of AttributeTags. Use to access and manage tagged values associated with this Attribute. |
| Name | <i>String</i> | Read/Write. The attribute name. |
| AttributeGUID | <i>String</i> | Read/Write. A globally unique ID for the current Attribute. System generated. |
| Visibility | <i>String</i> | Read/Write. The Scope of the Attribute. May be Private, Protected, Public or Package. |
| Containment | <i>String</i> | Read/Write. Type of Containment. May be "Not Specified", "By Reference" or "By Value". |
| IsStatic | <i>Boolean</i> | Read/Write. Indicates if the current attribute is a static feature or not. If attribute represents a database column, this when set represents the "Unique" option. |
| IsCollection | <i>Boolean</i> | Read/Write. Indicates if the current feature is a collection or not. If attribute represents a database column, this when set represents a Foreign Key. |
| IsOrdered | <i>Boolean</i> | Read/Write. Indicates if a collection is ordered or not. If attribute represents a database column, this when set represents a Primary Key. |
| AllowDuplicates | <i>Boolean</i> | Read/Write. Indicates if duplicates are allowed in the collection. If attribute represents a database column, this when set represents the "Not Null" option. |
| LowerBound | <i>String</i> | Read/Write. A value for the collection lower bound. |
| UpperBound | <i>String</i> | Read/Write. A value for the collection upper bound. |
| Container | <i>String</i> | Read/Write. The container type. |
| Notes | <i>String</i> | Read/Write. Further notes about this attribute. |
| IsDerived | <i>Boolean</i> | Read/Write. Indicates if attribute is derived (eg. a calculated value). |
| AttributeID | <i>Long</i> | Read only. Local ID number of Attribute. |
| Pos | <i>Long</i> | Read/Write. Position of Attribute in class attribute list. |
| Length | <i>String</i> | Read/Write. Attribute length where applicable. |
| Precision | <i>String</i> | Read/Write. Precision value. |
| Scale | <i>String</i> | Read/Write. Scale value. |
| IsConst | <i>Boolean</i> | Read/Write. Flag indicating if Attribute is Const or not. |

| | | |
|--------------|---------------|---|
| Style | <i>String</i> | Read/Write. Further style information. |
| StyleEx | <i>String</i> | Read/Write. Advanced style settings. Use with care. |
| ClassifierID | <i>Long</i> | Read/Write. Classifier ID if appropriate - indicates base type associated with attribute if not a primitive type. |
| Default | <i>String</i> | Read/Write. Default value associated with attribute. |
| Type | <i>String</i> | Read/Write. The Attribute type (by name - also see ClassifierID). |

| | | |
|----------------|-----------------------------|--|
| Stereotype | <i>String</i> | Read/Write. Sets or gets the Stereotype for this attribute. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |
| ParentID | <i>Long</i> | Read only. Returns the ElementID of the Element that this attribute is a part of. |
| TaggedValuesEx | <i>Collection</i> | Read only. Collection of TaggedValue objects belonging to the current Attribute and its parent attributes. |

Attribute Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Attribute object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6.2 Attribute Constraint

AttributeConstraint

public Class

An AttributeConstraint is a constraint associated with the current Attribute.

Associated table in .EAP file: t_attributeconstraints

AttributeConstraint Attributes

| Attribute | Type | Notes |
|-------------|-----------------------------|---|
| AttributeID | <i>Long</i> | Read/Write. ID of Attribute this constraint applies to. |
| Name | <i>String</i> | Read/Write. The Constraint. |
| Type | <i>String</i> | Read/Write. Type of Constraint. |
| Notes | <i>String</i> | Read/Write. Descriptive notes about constraint. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

AttributeConstraint Methods

| Method | Type | Notes |
|--------|------|-------|
|--------|------|-------|

| | | |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current AttributeConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6.3 Attribute Tag

AttributeTag

public Class

An AttributeTag represents a Tagged Value associated with an Attribute

Associated table in .EAP file: t_attributetag

AttributeTag Attributes

| Attribute | Type | Notes |
|------------------|---------------------------------|---|
| TagID | <i>Long</i> | Read only. Local ID to identify tagged value. |
| AttributeID | <i>Long</i> | Read/Write. Local ID of Attribute associated with this tagged value. |
| Name | <i>String</i> | Read/Write. Name or Tag. |
| Value | <i>String</i> | Read/Write. Value associated with this tag. |
| Notes | <i>String</i> | Read/Write. Descriptive notes. |
| TagGUID | <i>String</i> | Read/Write. A globally unique ID for this tagged value. |
| ObjectType | ObjectTyp e | Read only. Distinguishes objects referenced through Dispatch interface. |

AttributeTag Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | <i>Boolean</i> | Update the current AttributeTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |

15.1.2.6.4 Method

Method

public Class

A Method represents a UML operation. It is accessed from the Element Methods collection and includes collections for parameters, constraints and tagged values

Associated table in .EAP file: t_operation

Method Attributes

| Attribute | Type | Notes |
|----------------|-------------------|--|
| Parameters | <i>Collection</i> | Read only. The Parameters collection for the current Method. Use to add and access Parameter objects for the current Method. |
| PreConditions | <i>Collection</i> | Read only. PreConditions (constraints) as they apply to this method. Returns a MethodConstraint object of type "pre". |
| PostConditions | <i>Collection</i> | Read only. PostConditions (constraints) as they apply to this method. Returns a MethodConstraint object of type "post". |
| MethodGUID | <i>String</i> | Read/Write. A globally unique ID for the current Method. System generated. |
| TaggedValues | <i>Collection</i> | Read only. TaggedValues collection for the current method. Access a list of MethodTag objects. |
| MethodID | <i>Long</i> | Read only. A local ID for the current method - only valid within this .EAP file. |
| Name | <i>String</i> | Read/Write. The Method name. |
| Visibility | <i>String</i> | Read/Write. The method scope - Public, Protected, Private or Package. |
| ReturnType | <i>String</i> | Read/Write. Return type for the method - may be a primitive data type or a class or interface type. |
| ReturnsArray | <i>Boolean</i> | Read/Write. Flag to indicate return value is an array. |
| Stereotype | <i>String</i> | Read/Write. The method stereotype (optional). |
| IsStatic | <i>Boolean</i> | Read/Write. Flag to indicate a static method. |
| Concurrency | <i>Variant</i> | Read/Write. Concurrency type of method. |
| Notes | <i>String</i> | Read/Write. Descriptive notes about the Method. |
| Behavior | <i>String</i> | Read/Write. Some further explanatory behavior notes (eg. pseudocode). |
| Abstract | <i>Boolean</i> | Read/Write. Flag indicating if Method is abstract (1) or not (0). |
| IsSynchronized | <i>Boolean</i> | Read/Write. Flag indicating a Synchronized method call. |
| IsConst | <i>Boolean</i> | Read/Write. Flag indicating method is Const. |
| Style | <i>String</i> | Read/Write. Extended style information about method. |
| IsPure | <i>Boolean</i> | Read/Write. Flag indicating method is defined as Pure in C++. |
| Throws | <i>String</i> | Read/Write. Exception information. |
| ClassifierID | <i>String</i> | Read/Write. ClassifierID that applies to the ReturnType. |
| StyleEx | <i>String</i> | Read/Write. Advanced style settings. Not currently used. |
| Code | <i>String</i> | Read/Write. Optional field to hold Method Code (currently not used in EA). |
| IsRoot | <i>Boolean</i> | Read/Write. Flag to indicate if Method is Root. |
| IsLeaf | <i>Boolean</i> | Read/Write. Flag to indicate Method is Leaf (cannot be overridden). |
| IsQuery | <i>Boolean</i> | Read/Write. Flag to indicate method is a query (ie. does not alter class variables). |
| StateFlags | <i>String</i> | Read/Write. Some flags as applied to methods in State elements. |

| | | |
|----------|-------------|--|
| Pos | <i>Long</i> | Read/Write. Specifies the position of the method within the set of operations defined for a class. |
| ParentID | <i>Long</i> | Read only. An optional ID of an element that 'owns' this diagram - eg. a Sequence diagram owned by a Use Case. |

| | | |
|------------|---------------------------------|---|
| ObjectType | ObjectTyp e | Read only. Distinguishes objects referenced through Dispatch interface. |
|------------|---------------------------------|---|

Method Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Method object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6.5 Method Constraint

MethodConstraint

public Class

A Method constraint is a condition imposed on a Method. It is accessed through either the Method PreConditions or Method PostConditions collection.

Associated table in .EAP file: t_operationpres and t_operationposts

MethodConstraint Attributes

| Attribute | Type | Notes |
|------------|---------------------------------|---|
| MethodID | <i>Long</i> | Read/Write. The local ID of the associated method. |
| Name | <i>String</i> | Read/Write. The Name of the constraint. |
| Type | <i>String</i> | Read/Write. The constraint type. |
| Notes | <i>String</i> | Read/Write. Descriptive notes about this constraint. |
| ObjectType | ObjectTyp e | Read only. Distinguishes objects referenced through Dispatch interface. |

MethodConstraint Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current MethodConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6.6 Method Tag

MethodTag

public Class

A MethodTag is a tagged value associated with a method

Associated table in .EAP file: t_operationtag

MethodTag Attributes

| Attribute | Type | Notes |
|------------|----------------------------|---|
| TagID | Long | Read only. A unique ID for this tagged value. |
| MethodID | Long | Read/Write. The ID of the associated Method. |
| Name | String | Read/Write. The Tag or name of the property. |
| Value | String | Read/Write. A value to apply to this tag. |
| Notes | String | Read/Write. Descriptive notes about this item. |
| TagGUID | String | Read/Write. A uniqueID for this tagged value. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

MethodTag Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current MethodTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6.7 Parameter

Parameter

public Class

A Parameter object represents a method argument and is accessed through the Method Parameters collection.

Associated table in .EAP file: t_operationparams

Parameter Attributes

| Attribute | Type | Notes |
|---------------|---------|---|
| Name | String | Read/Write. The Parameter name - must be unique for a single Method. |
| Type | Variant | Read/Write. The Parameter type - may be a primitive type or defined classifier. |
| Default | String | Read/Write. A default value for this Parameter. |
| ParameterGUID | String | Read/Write. A globally unique ID for the current Parameter. System generated. |
| Position | Long | Read/Write. The position in the argument list. |
| IsConst | Boolean | Read/Write. Flag indicating the parameter is Const (cannot be altered). |
| Style | String | Read/Write. Some style information. |
| Kind | String | Read/Write. The parameter Kind - in, inout, out, return. |

| | | |
|--------------|-----------------------------|---|
| ClassifierID | <i>String</i> | Read/Write. A ClassifierID for the Parameter if known. |
| OperationID | <i>Long</i> | Read only. ID of the Method associated with this parameter. |
| Notes | <i>String</i> | Read/Write. Descriptive Notes. |
| StyleEx | <i>String</i> | Read/Write. Advanced style settings. Not currently used. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

Parameter Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Parameter object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.6.8 Partitions

Partitions

public Collection

A collection of internal element partitions (regions). This is commonly seen in Activities, States, Boundary, Diagram Frame and similar elements. Not all elements support partitions.

This collection contains a set of Partition elements. The set is read/write: information is not saved until the host element is saved, so ensure that you call the Element.Save method after making changes to a Partition.

Partition Attributes

| Attribute | Type | Notes |
|------------|-----------------------------|---|
| Name | <i>String</i> | Read/Write. The Partition name - may represent a condition or constraint in some cases. |
| Note | <i>String</i> | Read/Write. A free text note associated with this Partition. |
| Operator | <i>String</i> | Read/Write. An optional operator value that specifies the Partition type. |
| Size | <i>String</i> | Read/Write. Vertical or horizontal width of partition in pixels. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

15.1.2.6.9 Embedded Elements

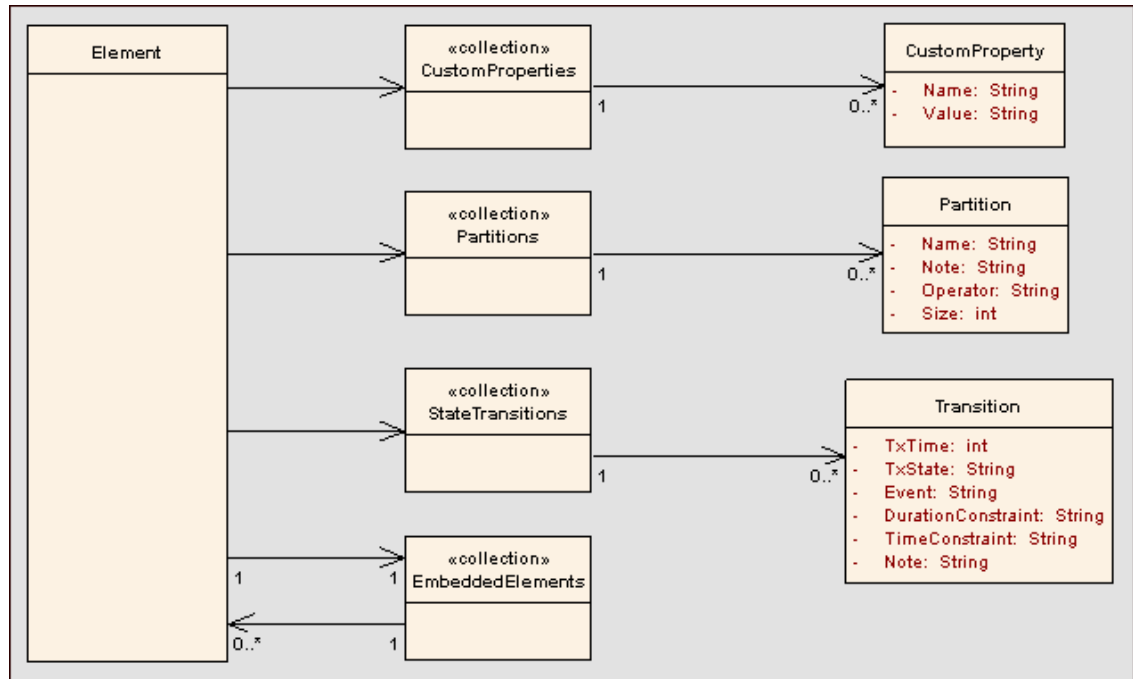
Embedded Elements

public Collection

In UML 2.0 an element may have one or more embedded elements such as Ports, Pins, Parameters, ObjectNodes etc. These are attached to the boundary of the host element and may not be moved off the

element. They are owned by their host element. This collection gives easy access to the set of elements embedded on the surface of an element. Note that some embedded elements may have their own embedded element collection (eg. Ports may have interfaces embedded on them).

The Embedded Elements collection contains Element objects.



15.1.2.6.10 Transitions

Transitions

public Collection

Applies only to Timeline elements. A Timeline element displays 0 or more state transitions at set times on its extent. This collection lets you access the transition set. You can also access additional information by referring to the Connectors associated with the Timeline - and referencing messages passed between timelines. Note that any changes made to elements in this collection are only saved when the main Element is saved.

Transition Attributes

| Attribute | Type | Notes |
|--------------------|--------|---|
| TxTime | String | Read/Write. The time that the transition occurs. Value depends on range set in diagram. |
| TxState | String | Read/Write. The state to transition to. Defined in the Timeline Properties dialog. |
| Event | String | Read/Write. Event (optional) that initiated transition. |
| DurationConstraint | String | Read/Write. A constraint on the time duration that the transition will take. |
| TimeConstraint | String | Read/Write. A constraint on when the transition has to be complete by. |

| | | |
|------------|---------------------------------|---|
| Note | <i>String</i> | Read/Write. A free text note. |
| ObjectType | ObjectT ype | Read only. Distinguishes objects referenced through Dispatch interface. |

15.1.2.6.11 Custom Properties

Custom Properties

public Collection

The CustomProperties collection contains 0 or more Cust Properties associated with the current element. These properties provide advanced UML configuration options, and may not be added to or deleted. The value of each property may be set.

Note that the number and type of property will vary depending on the actual element.

CustomProperty

| Attribute | Type | Notes |
|------------|---------------------------------|--|
| Name | <i>String</i> | Read-only. The Custom Property name. |
| Value | <i>String</i> | Read/Write. The value associated with this custom property. May be a string, the boolean values 'true' or 'false' or an enumeration value from a defined list. The UML 2.0 specification in general provides information on enumeration kinds relevant here. |
| ObjectType | ObjectT ype | Read only. Distinguishes objects referenced through Dispatch interface. |

15.1.2.6.12 Properties

Properties

Properties Attributes

| Attribute | Type | Notes |
|-----------|------|---|
| Count | Long | The number properties that are available for this object. |

Properties Methods

| Method | Type | Notes |
|----------------------|----------|--|
| Item(Variant Index) | Property | Returns a property either by name or by zero-based integer offset into the list of properties. |

Property

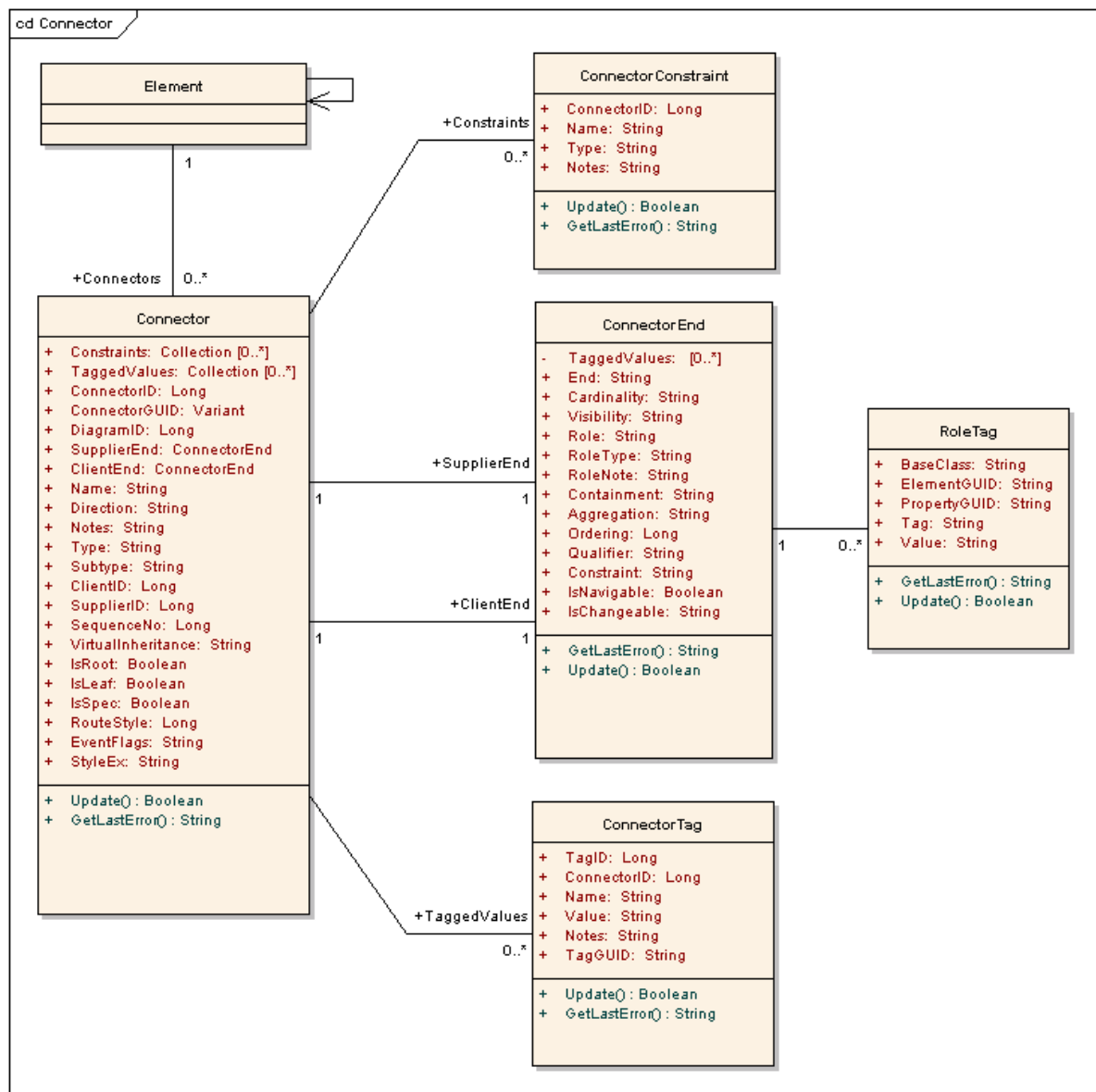
Property Attributes

| Attribute | Type | Notes |
|-----------|-------------------------------|--|
| Name | String | Read only. Identifies the property. The object to which the properties list applies may have an automation property with the same name, in which case the data accessed through Value is identical to that obtained through the automation property. |
| Type | PropTyp e | Read only. Provides an indication of what sort of data is going to be stored by this property. This restriction may be further defined by the Validation attribute. |

| | | |
|------------|---------|--|
| Validation | String | Read only. Optional string that is used to validate any data that is passed to the Value attribute. This string is used by the programmer at run time to provide an indication of what's expected, and by EA to ensure that the submitted data is appropriate. |
| Value | Variant | Read/write. The value of the property as defined in the other fields. |

15.1.2.7 Connector

Figure 6 : Connector



Connector

public Package

The Connector package details how connectors between elements are accessed and managed.

15.1.2.7.1 Connector

Connector

public Class

A Connector object represents the various kinds of connectors between UML Elements. It is accessed from either the Client or Supplier Element - using the Connectors collection of that element. When creating a new connector you must assign it a valid type. These are:

- Aggregation
- Association
- Collaboration
- Dependency
- Generalization
- Instantiation
- Nesting
- NoteLink
- Realisation
- Sequence
- StateFlow
- UseCase

Associated table in .EAP file: t_connector

Connector Attributes

| Attribute | Type | Notes |
|---------------|---------------------|---|
| Constraints | <i>Collection</i> | Read only. Collection of Constraint objects. |
| TaggedValues | <i>Collection</i> | Read only. Collection of TaggedValue objects. |
| ConnectorID | <i>Long</i> | Read only. Local identifier for the current connector. System generated. |
| ConnectorGUID | <i>Variant</i> | Read only. A globally unique ID for the current connector. System generated. |
| Color | <i>Long</i> | Read/Write. Sets the color of the connector. |
| DiagramID | <i>Long</i> | Read/Write. The DiagramID of the connector. |
| SupplierEnd | <i>ConnectorEnd</i> | Read Only. A pointer to the ConnectorEnd object representing the Target end of the relationship. |
| ClientEnd | <i>ConnectorEnd</i> | Read Only. A pointer to the ConnectorEnd object representing the Source end of the relationship. |
| Name | <i>String</i> | Read/Write. The connector name. |
| Direction | <i>String</i> | Read/Write. Connector Direction. May be set to one of the following: 1. Unspecified, 2. Bi-Directional, 3. Source -> Destination, 4. Destination -> Source. |
| Notes | <i>String</i> | Read/Write. Descriptive notes about connector. |
| Type | <i>String</i> | Read/Write. Connector type. Valid types are held in the t_connectortypes table in the .EAP file. |
| Subtype | <i>String</i> | Read/Write. A possible subtype to refine the meaning of the connector. |
| ClientID | <i>Long</i> | Read/Write. ElementID of the Element at the Source end of this Connector. |

| | | |
|--------------------|----------------------------|--|
| SupplierID | <i>Long</i> | Read/Write. ElementID of the Element at the Target end of this Connector. |
| SequenceNo | <i>Long</i> | Read/Write. The SequenceNo of the connector. |
| VirtualInheritance | <i>String</i> | Read/Write. For Generalization indicates if inheritance is virtual. |
| IsRoot | <i>Boolean</i> | Read/Write. Flag indicating connector is a root. |
| IsLeaf | <i>Boolean</i> | Read/Write. Flag indicating connector is a leaf. |
| IsSpec | <i>Boolean</i> | Read/Write. Flag indicating connector is a Specification. |
| RouteStyle | <i>Long</i> | Read/Write. The route style. |
| EventFlags | <i>String</i> | Read/Write. Structure to hold a variety of flags concerned with event signaling on messages. |
| StyleEx | <i>String</i> | Read/Write. Advanced style settings. Not currently used. |
| Stereotype | <i>String</i> | Read/Write. Sets or gets the Stereotype for this connector end. |
| TransitionEvent | <i>String</i> | Read/Write. See the Transition topic in this document for appropriate values. |
| TransitionGuard | <i>String</i> | Read/Write. See the Transition topic in this document for appropriate values. |
| TransitionAction | <i>String</i> | Read/Write. See the Transition topic in this document for appropriate values. |
| Width | <i>Long</i> | Read/Write. Specifies the width of the connector. |
| CustomProperties | <i>Collection</i> | Read only. Returns a collection of Advanced properties associated with an element in the form of CustomProperty objects. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |
| Properties | Properties | Returns a list of specialized properties which apply to the connector that may not be available using the automation model. The properties are purposely undocumented because of their obscure nature and because they are subject to change as progressive enhancements are made to them. |
| MetaType | <i>String</i> | Read only: The Connector's domain-specific meta type, as defined by an applied stereotype from an MDG Technology. |

Connector Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Connector object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |

15.1.2.7.2 Connector Constraints

ConnectorConstraint

public Class

A ConnectorConstraint holds information about special conditions that apply to a Connector. It is accessed through the Connector Constraints collection

Associated table in .EAP file: t_connectorconstraints

ConnectorConstraint Attributes

| Attribute | Type | Notes |
|-------------|-----------------------------|---|
| ConnectorID | Long | Read/Write. A local ID value (long) - system generated. |
| Name | String | Read/Write. The constraint name. |
| Type | String | Read/Write. The constraint type. |
| Notes | String | Read/Write. Notes about this constraint. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

ConnectorConstraint Methods

| Method | Type | Notes |
|-----------------|---------|---|
| Update () | Boolean | Update the current ConnectorConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs |

15.1.2.7.3 Connector End**ConnectorEnd**public Class

A ConnectorEnd contains information about a single end of a Connector. It may be a Supplier or a Client. A ConnectorEnd is accessed from the Connector as either the ClientEnd or SupplierEnd

Associated table in .EAP file: derived from t_connector

ConnectorEnd Attributes

| Attribute | Type | Notes |
|--------------|----------|--|
| TaggedValues | private: | Read only collection. |
| End | String | Read only. The End this connectorEnd object applies to - Client or Supplier. |
| Cardinality | String | Read/Write. Cardinality associated with this end. |
| Visibility | String | Read/Write. Scope associated with this connector end. Valid types are: Public, Private, Protected and Package. |
| Role | String | Read/Write. The connector end role. |
| RoleType | String | Read/Write. The Role type applied to this end of the connector. |
| RoleNote | String | Read/Write. Notes associated with the role of this connector end. |
| Containment | String | Read/Write. Containment type applied to this connector end. |
| Aggregation | Long | Read/Write. Aggregation as it applies to this end. Valid values are: 0 = None, 1 = Shared, 2 = Composite. |
| Ordering | Long | Read/Write. Ordering for this connector end. |
| Qualifier | String | Read/Write. A qualifier that may apply to connector end. |

| | | |
|--------------|----------------------------|---|
| Constraint | <i>String</i> | Read/Write. A constraint that may be applied to this connector end. |
| IsNavigable | <i>Boolean</i> | Read/Write. Flag indicating this end is navigable from the other. |
| IsChangeable | <i>String</i> | Read/Write. Flag indicating this end is changeable or not. |
| Stereotype | <i>String</i> | Read/Write. Sets or gets the Stereotype for this connector end. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

ConnectorEnd Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | <i>Boolean</i> | Update the current ConnectorEnd object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |

15.1.2.7.4 Connector Tag

ConnectorTag

public Class

A ConnectorTag is a tagged value for a connector and is accessed through the Connector TaggedValues collection

Associated table in .EAP file: t_connectortag

ConnectorTag Attributes

| Attribute | Type | Notes |
|-------------|----------------------------|---|
| TagID | <i>Long</i> | Read only. A local ID to identify the tagged value. |
| ConnectorID | <i>Long</i> | Read/Write. The local ID of the associated connector. |
| Name | <i>String</i> | Read/Write. The tag or name. |
| Value | <i>String</i> | Read/Write. A value associated with the tag. |
| Notes | <i>String</i> | Read/Write. Descriptive notes associated with this tagged value. |
| TagGUID | <i>String</i> | Read/Write. A globally unique ID for this tagged value. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

ConnectorTag Methods

| Method | Type | Notes |
|-----------|----------------|--|
| Update () | <i>Boolean</i> | Update the current ConnectorTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |

| | | |
|-----------------|--------|--|
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
|-----------------|--------|--|

15.1.2.7.5 Role Tag

RoleTag

public Class

This interface provides access to the association role tagged values. Each connector end has a RoleTag collection that can be accessed to add, delete and access the RoleTags.

In code you will do something like (where con is a connector object):

Code fragment for accessing a RoleTag in VB.NET:

```

client = con.ClientEnd
client.Role = "m_client"
client.Update()
tag = client.TaggedValues.AddNew("tag", "value")
tag.Update()
tag = client.TaggedValues.AddNew("tag2", "value2")
tag.Update()
client.TaggedValues.Refresh()
For idx = 0 To client.TaggedValues.Count - 1
    tag = client.TaggedValues.GetAt(idx)
    Console.WriteLine(tag.Tag)
    client.TaggedValues.DeleteAt(idx, False)
Next
tag = Nothing

```

RoleTag Attributes

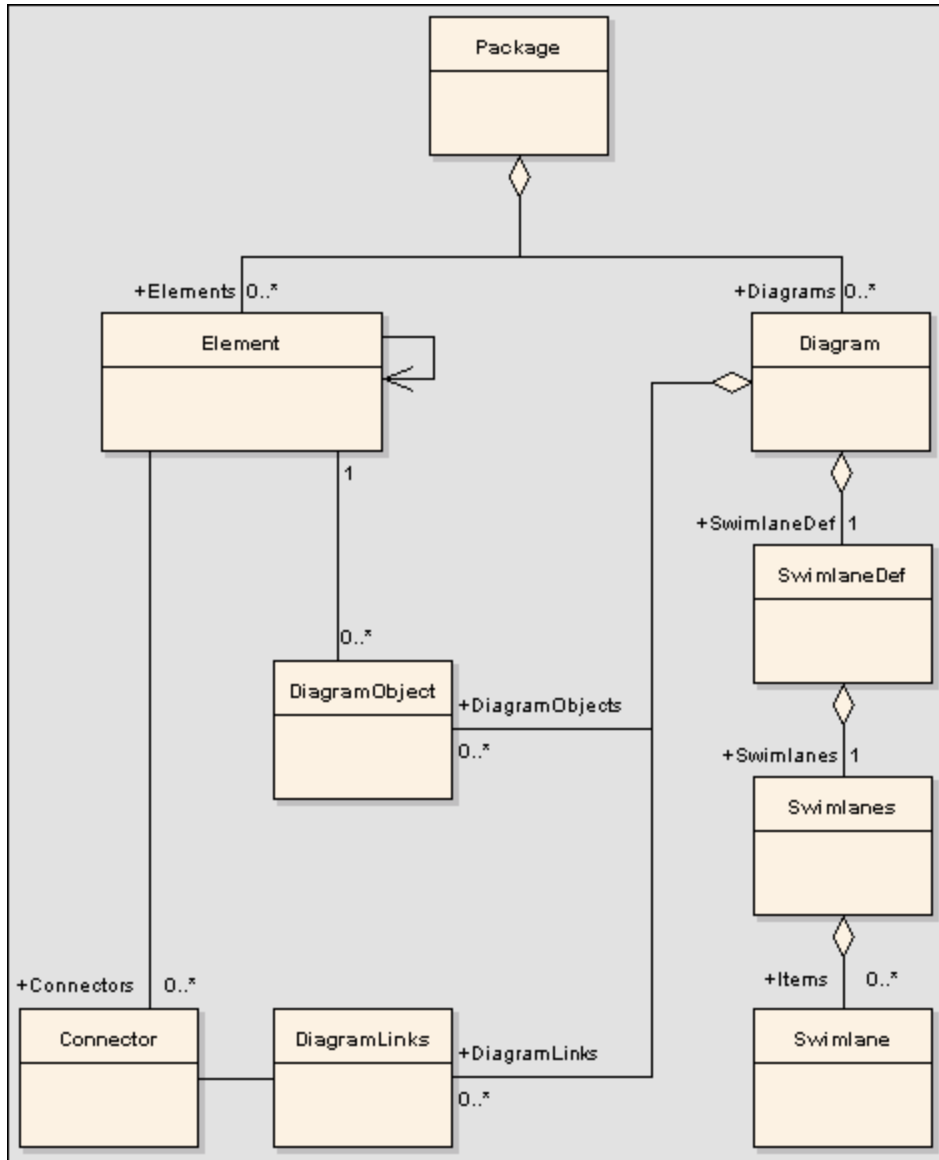
| Attribute | Type | Notes |
|--------------|-----------------------------|---|
| BaseClass | String | Read/Write. Indicates the role end - set to ASSOCIATION_SOURCE or ASSOCIATION_TARGET. |
| ElementGUID | String | Read/Write. GUID of the Connector with which this role tag is associated. |
| PropertyGUID | String | Read/Write. A system generated GUID to identify the tagged value. |
| Tag | String | Read/Write. The actual tag name. |
| Value | String | Read/Write. The value associated with this tag. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

RoleTag Methods

| Method | Type | Notes |
|-----------------|---------|--|
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | Boolean | Update the RoleTag after changes or on initial creation. |

15.1.2.8 Diagram

Figure 7 : Diagram



Diagram

public Package

The Diagram package has information on a Diagram and DiagramObjects and DiagramLinks - which are the instances of Elements within a diagram.

15.1.2.8.1 Diagram

Diagram

public Class

A Diagram corresponds to a single EA diagram. It is accessed through the Package Diagrams collection and in turn contains a collection of diagram objects and diagram links. Adding to the DiagramObjects collection will add an element to the Diagram (the Element must already exist). When adding a new diagram, you must set the diagram type to a valid type - these are:

- Activity
- Analysis
- Component
- Custom
- Deployment
- Logical
- Sequence
- Statechart
- Use Case

Note: Use the Analysis type for a Collaboration Diagram.

Associated table in .EAP file: t_diagram

Diagram Attributes

| Attribute | Type | Notes |
|----------------|----------------|---|
| DiagramObjects | Collecti on | Read only. A collection of references to DiagramObjects. A DiagramObject is an instance of an Element in a Diagram - and includes size and display characteristics. |
| DiagramLinks | Collecti on | Read only. A list of DiagramLink objects - each containing information about the display characteristics of a connector in a diagram. Note: A Diagram link is only created once a user modifies a connector in a diagram in some way. Until this condition has been met default values are used and the diagram link is not in use. |
| DiagramID | Long | Read only. A local ID for the diagram. |
| PackageID | Long | Read/Write. An ID of the package that this diagram belongs to. |
| ParentID | Long | Read/Write. An optional ID of an element that 'owns' this diagram - eg. a Sequence diagram owned by a Use Case. |
| Type | String | Read only. The Diagram Type. See the t_diagramtypes table in the .EAP file for more information. |
| Name | String | Read/Write. The diagram name. |
| Version | String | Read/Write. The version of the diagram. |
| Author | String | Read/Write. The author. |
| ShowDetails | String | Read/Write. Flag to indicate Diagram Details text should be shown. |
| ShowPublic | Boolea n | Read/Write. Flag to show or hide Public features. |
| ShowPrivate | Boolea n | Read/Write. Flag to show or hide Private features. |
| ShowProtected | Boolea n | Read/Write. Flag to show or hide Protected features. |
| Orientation | String | Read/Write. Page orientation - use "P" or "L" for Portrait or Landscape respectively. |
| cx | Long | Read/Write. The X dimension of the diagram (800 is default). |
| cy | Long | Read/Write. The Y dimension of diagram (1100 is default). |
| Scale | Long | Read/Write. The zoom scale - 100 is default. |
| CreatedDate | Date | Read/Write. The date diagram created. |

| | | |
|---------------------|-----------------------------|--|
| ModifiedDate | <i>Variant</i> | Read/Write. The date the diagram was last modified. |
| HighlightImports | <i>Boolean</i> | Read/Write. Flag to indicate elements from other packages should be highlighted. |
| ShowPackageContents | <i>Boolean</i> | Read/Write. Flag to indicate package contents should be shown in the current diagram. |
| StyleEx | <i>Long</i> | Read/Write. Advanced style settings. Not currently used. |
| ExtendedStyle | <i>String</i> | Read/Write. An extended style attribute. |
| IsLocked | <i>Boolean</i> | Read/Write. Flag indicating this diagram is locked or not. |
| DiagramGUID | <i>Variant</i> | Read/Write. A globally unique ID for this diagram. |
| Swimlanes | <i>String</i> | Read/Write. Information on swimlanes contained in the diagram. Please note that this property is superceded by SwimlaneDef . |
| Notes | <i>String</i> | Read/Write. Set/retrieve notes for this diagram. |
| Stereotype | <i>String</i> | Read/Write. Sets or gets the stereotype for this diagram. |
| SelectedObjects | <i>Collection</i> | Read only. Gets a collection representing the currently selected elements on the diagram. |
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |
| SelectedConnector | <i>Connector</i> | Read only. Returns the currently selected diagram connector. Returns Null if there is no currently selected diagram. |
| MetaType | <i>String</i> | Read only: The Diagram's domain-specific meta type, as defined by an MDG Technology |
| SwimlaneDef | SwimlaneDef | Read/Write. Information on swimlanes contained in the diagram. |

Diagram Methods

| Method | Type | Notes |
|--------------------|----------------|---|
| Update () | <i>Boolean</i> | Update the current Diagram object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| ReorderMessages () | void | Resets the display order of sequence and collaboration messages. Typically used after inserting or deleting messages in the diagram. |

15.1.2.8.2 Diagram Links

DiagramLinks

public Class

A DiagramLink is an object which holds display information about a connector between two elements in a specific diagram. It includes the custom points, display appearance & etc. Accessed from the Diagram DiagramLinks collection.

Associated table in .EAP file: t_diagramlinks

DiagramLinks Attributes

| Attribute | Type | Notes |
|-------------|-------------|--|
| DiagramID | <i>Long</i> | Read/Write. The local ID for the associated diagram. |
| ConnectorID | <i>Long</i> | Read/Write. The ID of the associated connector. |

| | | |
|------------|-----------------------------|---|
| Geometry | <i>String</i> | Read/Write. The geometry associated with the current connector in this diagram. |
| IsHidden | <i>Boolean</i> | Read/Write. Flag to indicate if this item is hidden or not. |
| Path | <i>String</i> | Read/Write. The path of the connector in this diagram. |
| Style | <i>String</i> | Read/Write. Additional style information - eg. color, thickness. |
| InstanceID | <i>Long</i> | Read only attribute. Holds the link identifier for the current model. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

DiagramLinks Methods

| Method | Type | Notes |
|-----------------|----------------|---|
| GetLastError () | <i>String</i> | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | <i>Boolean</i> | Update the current DiagramLink object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |

15.1.2.8.3 Diagram Objects

DiagramObjects

public Class

The DiagramObjects collection holds a list of element IDs and presentation information that indicates what will be displayed in a diagram and how it will be shown

Associated table in .EAP file: t_diagramobjects

DiagramObjects Attributes

| Attribute | Type | Notes |
|------------|-----------------------------|--|
| DiagramID | <i>Long</i> | Read/Write. The ID of the associated diagram (long). |
| ElementID | <i>Long</i> | Read/Write. The ElementID of the object instance in this diagram. |
| left | <i>Long</i> | Read/Write. The left position of the element. |
| right | <i>Long</i> | Read/Write. The right position of the element. |
| top | <i>Long</i> | Read/Write. The top position of the element. |
| bottom | <i>Long</i> | Read/Write. The bottom position of the element. |
| InstanceID | <i>Long</i> | Read/Write. Read only attribute. Holds the link identifier for the current model. |
| Sequence | <i>Long</i> | Read/Write. The sequence position when loading into diagram (affects Z order). |
| Style | <i>Variant</i> | Write only (reading this value will give undefined results). Style information for this object. See Setting the Style below for more information. |
| ObjectType | Object Type | Read only. Distinguishes objects referenced through Dispatch interface. |

DiagramObjects Methods

| Method | Type | Notes |
|-----------------|---------|---|
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| Update () | Boolean | Update the current DiagramObject object after modification or appending a new item. If false is returned, check the GetLastError function for more information. |

Setting The Style

The Style attribute is used for setting the appearance of a diagram object. It is set with a string value in the format "BCol=n;BFol=n;LCol=n;LWth=n;" where:

BCol = Background Color
BFol = Font Color
LCol = Line Color
LWth = Line Width

The color value is a decimal representation of the hex RGB value, where red=FF, green=FF00 and blue=FF0000. eg.

```
DiagObj.Style = "BCol=35723;BFol=9342520;LCol=9342520;LWth=1;"
```

The following code snippet shows how you might change the style settings for all of the objects in the current diagram, in this case changing everything to red.

```
For Each aDiagObj In aDiag.DiagramObjects
    aDiagObj.Style = "BCol=255;BFol=9342520;LCol=9342520;LWth=1;"
    aDiagObj.Update
    aRepos.ReloadDiagram aDiagObj.DiagramID
Next
```

15.1.2.8.4 SwimlaneDef

A swimlane object makes available attributes relating to a single row or column in a list of swimlanes.

| Attribute | Type | Notes |
|----------------|---------------------------|--|
| Swimlanes | Swimlanes | Read/Write. A list of individual swimlanes. |
| Orientation | String | Read/Write. Indication of whether the swimlanes are vertical or horizontal. |
| Locked | Boolean | Read/Write. If set to true, disables user modification of the swimlanes via the diagram. |
| HideNames | Boolean | Read/Write. Set to true to hide the swimlane titles. |
| ShowInTitleBar | Boolean | Read/Write. Allows vertical swimlane titles to be shown in title bar. |
| HideClassifier | Boolean | Read/Write. Removes any classifier from title display. |
| LineWidth | Long | Read/Write. Width of line, in pixels, used to draw swimlanes. Valid values: 1, 2 or 3. |
| LineColor | Long | Read/Write. RGB color used to draw swimlane borders. |
| Bold | Boolean | Read/Write. Show the title text in bold. |
| FontColor | Long | Read/Write. RGB color used to draw the titles. |

15.1.2.8.5 Swimlanes

A swimlanes object is attached to a Diagram's [SwimlaneDef](#) object and provides a mechanism to access individual swimlanes.

| Attribute | Type | Notes |
|-----------|------|-------|
|-----------|------|-------|

| | | |
|-------|-------------------------------------|---|
| Items | Swimlane collection | Read/Write. Access an individual swimlane. param: Index [Object] Either a string representing the title text or an integer representing the zero-based index of the swimlane to delete. If the string matches more than one swimlane title, the first matching swimlane is returned. |
|-------|-------------------------------------|---|

| Method | Type | Notes |
|-----------------------------|--------------------------|--|
| Count () | Long | Gives the number of swimlanes. |
| Add (String, Long) | Swimlane | Adds a new swimlane to the end of the list. param: Title [String] The title text which appears at the top of the swimlane. May be the same as an existing swimlane title. param: Width [Long] The width of the swimlane in pixels. returns: A swimlane object representing the newly added entry. |
| Delete (Object) | void | param: Index [Object] Either a string representing the title text or an integer representing the zero-based index of the swimlane to delete. If the string matches more than one entry, only the first entry is deleted. |
| DeleteAll () | void | Removes all swimlanes. |
| Insert (Long, String, Long) | Swimlane | Inserts a swimlane at a specific position. param: Index [Long] The zero-based index of the existing Swimlane before which this new entry will be inserted. param: Title [String] The title text which appears at the top of the swimlane. May be the same as an existing swimlane title. param: Width [Long] The width of the swimlane in pixels. returns: A swimlane object representing the newly added entry. |

15.1.2.8.6 Swimlane

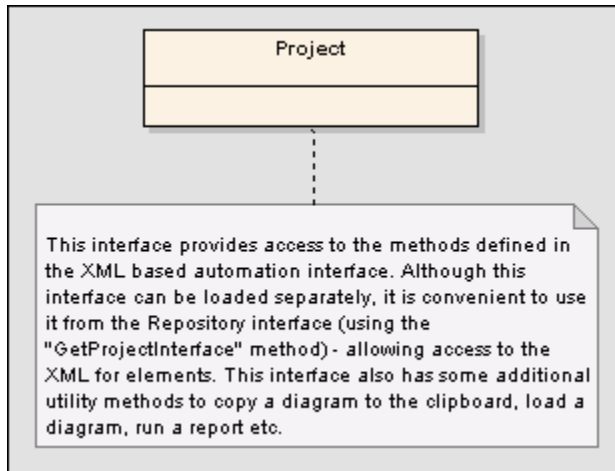
A swimlane object makes available attributes relating to a single row or column in a list of swimlanes.

| Attribute | Type | Notes |
|----------------|--------|--|
| Title | String | Read/Write. Text at the head of the swimlane. |
| ClassifiedGuid | String | Read/Write. The guid of the classifier class. This may be obtained from the corresponding Element object via the ElementGUID property. |
| Width | Long | Read/Write. The width of the swimlane in pixels. |
| BackColor | Long | Read/Write. The swimlane is filled with this RGB color. |

15.1.2.9 Project Interface

The EA.Project interface. This is the XML based interface to EA elements - and also includes some utility functions. You can get a pointer to this interface using the Repository.GetProjectInterface method.

Figure 8 : Project Interface



Project Interface

public Package

This package contains details on the XML based automation interface and its use.

15.1.2.9.1 Project

Project

public Class

Implements: CCmdTarget. The Project interface can be accessed from the Repository using GetProjectInterface(). The returned interface provides access to the XML based automation interface present in EA. Use this interface to get XML for the various internal elements and to run some utility functions to load diagrams, run reports & etc.

Project Attributes

| Attribute | Type | Notes |
|------------|----------------------------|---|
| ObjectType | ObjectType | Read only. Distinguishes objects referenced through Dispatch interface. |

Project Methods

| Method | Type | Notes |
|----------------------|---------------------------------------|---|
| LoadProject (String) | protected abstract: <i>Boolean</i> | param: FileName [String - in] Load an EA project file. Do not use this method if you have accessed the Project interface from the Repository, which will already have loaded a file. |
| ReloadProject () | protected abstract: <i>Boolean</i> | Reload the current project. Convenience method to refresh the current loaded project (in case of outside changes to the .EAP file). |

| | | |
|--|---------------------------------------|--|
| LoadDiagram (<i>String</i>) | protected abstract: <i>Boolean</i> | param: DiagramGUID [<i>String</i> - in] Load a diagram by its GUID. Note that EA expects this GUID in XML format ... if you retrieve the GUID using the Diagram interface, you will need to convert to XML format - use the GUIDtoXML and XMLtoGUID functions to do this. |
| SaveDiagramImageToFile (<i>String</i>) | protected abstract: <i>String</i> | param: FileName [<i>String</i> - in] The filename of the image to save. Save a diagram image of the current diagram to file. |
| GetElement (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] The GUID of the element to retrieve XML for. GUID must be in XML format (use GUIDtoXML to change an EA GUID to an XML GUID). Get XML for the specified element. |
| EnumViews () | protected abstract: <i>String</i> | Enumerate the views for a project. Returned as an XML document. |
| EnumPackages (<i>String</i>) | protected abstract: <i>String</i> | param: PackageGUID [<i>String</i> - in] Get a list of packages inside another. Returned as XML. Supply the GUID of the parent package - in XML format. |
| EnumElements (<i>String</i>) | protected abstract: <i>String</i> | param: PackageGUID [<i>String</i> - in] GUID of package to get list of elements for. Must be in XML format. List elements inside a package. Returned in XML format. |
| EnumLinks (<i>String</i>) | protected abstract: <i>String</i> | param: PackageID [<i>String</i> - in] The package to get all associated links for. |
| EnumDiagrams (<i>String</i>) | protected abstract: <i>String</i> | param: PackageGUID [<i>String</i> - in] The GUID of the package to list diagrams for. Must be in XML format. Get an XML list of all diagrams in a specified package. |
| EnumDiagramElements (<i>String</i>) | protected abstract: <i>String</i> | param: DiagramGUID [<i>String</i> - in] The diagram GUID (in XML format) of the diagram to get elements for Get a list of all elements contained in a diagram - in XML format. |

| | | |
|------------------------------------|--------------------------------------|--|
| EnumDiagramLinks (<i>String</i>) | protected abstract: <i>String</i> | param: DiagramID [<i>String</i> - in] The Diagram ID to get links for. Get a list of links appearing in a diagram (in XML). |
| GetLink (<i>String</i>) | protected abstract: <i>String</i> | param: LinkGUID [<i>String</i> - in] The GUID (in XML format) to get details of. Get connector details in XML format. |
| GetDiagram (<i>String</i>) | protected abstract: <i>String</i> | param: DiagramGUID [<i>String</i> - in] The diagram ID (in XML format) Get diagram details in XML format. |

| | | |
|--|-----------------------------------|--|
| GetElementConstraints (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get constraints (in XML) for an element. Supply the element ID in XML format. |
| GetElementEffort (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get effort (in XML format) for an element. |
| GetElementMetrics (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get metrics in XML format for an element. |
| GetElementFiles (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get files for an element in XML format. |
| GetElementRequirements (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get a list of requirements for an element in XML format. |
| GetElementProblems (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get a list of Issues (problems) associated with an element. |
| GetElementResources (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get a resource list (in XML) for an element. |
| GetElementRisks (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get a list of risks associated with an element - in XML format. |
| GetElementScenarios (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get a list of scenarios for an element - in XML format. |
| GetElementTests (<i>String</i>) | protected abstract: <i>String</i> | param: ElementGUID [<i>String</i> - in] Get a list of tests for an element (in XML format). |
| ShowWindow (<i>Long</i>) | protected abstract: <i>void</i> | param: Show [<i>Long</i> - in] Show or Hide the EA User Interface. |

| | | |
|--|------------------------------------|---|
| Exit () | protected abstract: <i>void</i> | Exit the current instance of EA - this function is maintained for backward compatibility and should never be called. EA will automatically disappear when you are no longer using any of the provided objects. |
| PutDiagramImageOnClipboard (<i>String, Long</i>) | protected abstract: <i>Boolean</i> | param: DiagramGUID [<i>String</i> - in] param: Type [<i>Long</i> - in] Place an image of the current diagram on the clipboard. |

| | | |
|---|--------------------------------|---|
| PutDiagramImageToFile (String, String, Long) | protected abstract: Boolean | param: DiagramGUID [String - in] param: Filename [String - in] param: Type [Long - in] If type = 0 then it will be metafile If type = 1 then it will use the file type from the name extension (ie. .bmp, .jpg, .gif, .png, .tga) Place an image of the current diagram to file. |
| ExportPackageXML (String, XMIMType, Long, Long, Long, Long, String) | protected abstract: String | param: PackageGUID [String - in] The GUID of the package to be exported param: XMIMType [EnumXMIMType - in] Specifies the XML type and version information. See XMIMType Enum for allowable values. param: DiagramXML [Long - in] True if XML for diagrams is required param: DiagramImage [Long - in] Format for diagram images to be created at the same time. Accepted Values [-1=NONE,0=EMF,1=BMP,2=GIF,3=PNG,4=JPG] param: FormatXML [Long - in] True if XML output should be formatted prior to saving param: UseDTD [Long - in] True if a DTD should be used param: FileName [String - in] The filename to output to. Export XML for a specified package. |
| EnumProjects () | protected abstract: String | Get a list of projects in the current file - corresponds to Model in Repository. |

| | | |
|------------------------------------|-------------------------------|--|
| EnumViewEx (String) | protected abstract: String | param: ProjectGUID [String - in] Get a list of Views in the current project. |
| RunReport (String, String, String) | protected abstract: void | param: PackageGUID [String - in] param: TemplateName [String - in] param: FileName [String - in] Run a named report - RTF. |
| GetLastError () | protected abstract: String | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs. |
| GetElementProperties (String) | protected abstract: String | param: ElementGUID [String - in] Get tagged values for a specified element. |
| GUIDtoXML (String) | String | param: GUID [String - in] The EA style GUID to convert to XML format Change an internal GUID to the form used in XML |
| XMLtoGUID (String) | String | param: GUID [String - in] The XML style GUID to convert to EA internal format. Change a GUID in XML format to the form used inside EA. |

| | | |
|--|----------------|---|
| RunHTMLReport (<i>String, String, String, String, String</i>) | <i>String</i> | <p>param: PackageGUID [String - in] param: ExportPath [String - in] param: ImageFormat [String - in] param: Style [String - in] param: Extension [String - in]</p> <p>Run a HTML report (same as <i>Documentation HTML Documentation</i> on right click of Package in the Project Browser).</p> |
| ImportPackageXMI (<i>String, String, Long, Long</i>) | <i>String</i> | <p>param: PackageGUID [String - in] PackageGUID is the filename to import into (or overwrite). param: Filename or XMLText [String - in] The name of the XMI file.</p> <p>Note: <i>If the String is of type filename it will be interpreted as a source file, otherwise the String will be imported as XML text.</i></p> <p>param: ImportDiagrams [Long - in] param: StripGUID [Long - in] Boolean value to indicate whether you want to replace the element UniqueIDs on import. If stripped, then a package could be imported twice into EA - as two different versions.</p> <p>Provides the ability to import an XMI file at a point in the tree.</p> |
| SaveControlledPackage (<i>String</i>) | <i>String</i> | <p>param: PackageGUID [String - in]</p> <p>Saves a package that has been configured as a controlled package - to XMI. The package GUID is only required, EA picks the rest up from the package control info.</p> |
| LoadControlledPackage (<i>String</i>) | <i>String</i> | <p>param: PackageGUID [String - in]</p> <p>Loads a package that has been marked and configured as controlled. The filename details & etc. are stored in the package control data.</p> |
| LayoutDiagram (<i>String, Long</i>) (Deprecated) | <i>Boolean</i> | <p>param: DiagramGUID [String - in] param: LayoutStyle [Long - in] this parameter is always ignored - it is recommended that LayoutDiagramEx is used instead</p> <p>Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for class and object diagrams.</p> |

| | | |
|---|----------------|--|
| <p>LayoutDiagramEx (String, Long, Long, Long, Long, Boolean)</p> | <p>Boolean</p> | <p>param: DiagramGUID [String - in] param: LayoutStyle [Long - in] param: Iterations [Long - in] The number of layout iterations the Layout process should take to perform cross reduction (Default value = 4). param: LayerSpacing [Long - in] The per-element layer spacing the Layout process shall use (Default value = 20). param: ColumnSpacing [Long - in] The per-element column spacing the Layout process shall use (Default value = 20). param: SaveToDiagram [Boolean - in] Specifies whether or not EA should save the supplied layout options as default to the diagram in question.</p> <p>Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for class and object diagrams.</p> <p>LayoutStyle accepts the following options (refer to Layout a Diagram for further info):</p> <ul style="list-style-type: none"> • Default Options: IsDiagramDefault IsProgramDefault • Cycle Removal Options: IsCycleRemoveGreedy IsCycleRemoveDFS • Layering Options: IsLayeringLongestPathSink IsLayeringLongestPathSource IsLayeringOptimalLinkLength • Initialize Options: IsInitializeNaive IsInitializeDFSOut IsInitializeDFSIn • Crossing Reduction Options: IsCrossReduceAggressive • Layout Options - Direction IsLayoutDirectionUp IsLayoutDirectionDown IsLayoutDirectionLeft IsLayoutDirectionRight |
|---|----------------|--|

| | | |
|--|----------------|---|
| <p>GenerateXSD(String, String, String, String)</p> | <p>Boolean</p> | <p>Create a XML schema for this GenerateXSD. Returns True on success. Parameters: PackageGUID: String, Identifies the package Filename: String, Target filepath Encoding: String, The XML encoding for the code page instruction. Options: String, Unused.</p> |
|--|----------------|---|

| | | |
|--|----------------|---|
| GenerateClass (String) | <i>Boolean</i> | Generates the code for a single class. Parameters: ElementGUID: String, Identifies the element to generate. ExtraOptions: String, Allows for extra options to be given to the command. Currently unused. |
| GeneratePackage (String, String) | <i>Boolean</i> | Generates the code for all classes within a package. Parameters: PackageGUID: String, Identifiers the package to generate. ExtraOptions: String, Allows for extra options to be given to the command. Currently allows generation of all subpackages (recurse), force overwrite of all files (overwrite) and specification to auto generate all paths (dir). For example: "recurse=1;overwrite=1;dir=C:\\" |
| TransformElement (String, String, String, String) | <i>Boolean</i> | Parameters: TransformName: String, Specifies the transformation that should be executed. ElementGUID: String, Identifies the element to transform. TargetPackageGUID: String, Identifiers the package to transform into. ExtraOptions: String, Allows for extra options to be given to the command. Currently unused. |
| TransformPackage (String, String, String, String) | <i>Boolean</i> | Runs a transformation on the contents of a package. Parameters: TransformName: String, Specifies the transformation that should be executed. SourcePackageGUID: String, Identifies the package to transform. TargetPackageGUID: String, Identifiers the package to transform into. ExtraOptions: String, Allows for extra options to be given to the command. Currently unused. |
| SynchronizeClass (String, String) | <i>Boolean</i> | Synchronizes a class with latest source code. Parameters: ElementGUID: String, Identifies the element to update from code. ExtraOptions: String, Allows for extra options to be given to the command. Currently unused. |
| SynchronizePackage (String, String) | <i>Boolean</i> | Synchronizes each class in a package with latest source code. Parameters: PackageGUID: String, Identifies the package containing the elements to update from code. ExtraOptions: String, Allows for extra options to be given to the command. Currently allows synchronisation of all child package (children). For example "children=1;" |

| | | |
|---|----------------|---|
| ImportDirectory (String, String, String, String) | <i>Boolean</i> | Imports source code directory into the model. Parameters: PackageGUID: String, Identifies the package to reverse engineer code into. DirectoryPath: String, Specifies the path where the code is found on the computer. Language: String, Specifies the language of the code to be imported. ExtraOptions: String, Allows for extra options to be given to the command. Currently allows import of source from all child directories (recurse). For example "recurse=1;" |
|---|----------------|---|

15.1.2.10 Code Samples

This section contains various code examples indicating how to use the Automation Interface, written in VB Dot Net:

- [Open the Repository](#)
- [Iterate Through an EAP File](#)
- [Add and Manage Packages](#)
- [Add and Manage Elements](#)
- [Add a Connector](#)
- [Add and Manage Diagrams](#)
- [Adding and Deleting Attributes and Methods](#)
- [Element Extras](#)
- [RepositoryExtras](#)
- [Stereotypes](#)
- [Working with Attributes](#)
- [Working with Methods](#)

15.1.2.10.1 Open the Repository

public Object

```
'an example of how to open an EA repository
'inVB.Net

Public Class AutomationExample

    'class level variable for Repository
    Public m_Repository As Object

    Public Sub Run()
    try
        'create the repository object
        m_Repository = CreateObject("EA.Repository")

        'open an EAP file
        m_Repository.OpenFile("F:\Test\EAAuto.EAP")
        'use the Repository in anyway required
        'DumpModel

        'close the repository and tidy up
        m_Repository.Exit()
        m_Repository = Nothing

    ...catcheasexception
        Console.WriteLine(e)
    End try
    End Sub
```

```
end Class
```

15.1.2.10.2 Iterate Through an EAP File

public Object

```
'assume repository has already been opened

' start at the model level
Sub DumpModel()
    Dim idx as Integer
    For idx=0 to m_Repository.Models.Count-1
        DumpPackage(" ",m_Repository.Models.GetAt(idx))
    Next
End Sub

'output package name, then element contents, then process child packages
Sub DumpPackage(Indent as String, Package as Object)
    Dim idx as Integer
    Console.WriteLine(Indent + Package.Name)
    DumpElements(Indent + " ", Package)

    For idx = 0 to Package.Packages.Count-1
        DumpPackage(Indent + " ", Package.Packages.GetAt(idx))
    Next
End Sub

' dump element name
Sub DumpElements(Indent as String, Package as Object)
    Dim idx as Integer
    For idx= 0 to Package.Elements.Count-1
        Console.WriteLine(Indent + ":::" + Package.Elements.GetAt(idx).Name)
    Next
End Sub
```

15.1.2.10.3 Add and Manage Packages

public Object

Example illustrating how to add a Model or a Package

```
Sub TestPackageLifecycle

    Dim idx as integer
    Dim idx2 as integer
    Dim package as object
    Dim model as object
    Dim o as object

    'first add a new Model

    model = m_Repository.Models.AddNew("AdvancedModel", "")
    If not model.Update() Then
        Console.WriteLine(model.GetLastError())
    End If

    'refresh the models collection
    m_Repository.Models.Refresh

    'now work through models collection and add a package

    For idx = 0 to m_Repository.Models.Count -1
        o = m_Repository.Models.GetAt(idx)
        Console.WriteLine(o.Name)
        If o.Name = "AdvancedModel" Then
```

```

        package = o.Packages.Addnew( "Subpackage", "Nothing" )
        If not package.Update() Then
            Console.WriteLine(package.GetLastError())
        EndIf

        package.Element.Stereotype = "system"
        package.Update

        '' for testing purposes just delete the
        '' newly created Model and its contents
        m_Repository.Models.Delete(idx)

    EndIf
Next

End Sub

```

15.1.2.10.4 Add and Manage Elements

public Object

```

'' Add and delete elements in a package

Sub ElementLifecycle

    Dim package as Object
    Dim element as Object

    package = m_Repository.GetPackageByID(2)
    element = package.elements.AddNew( "Login to Website", "UseCase" )
    element.Stereotype = "testcase"
    element.Update
    package.elements.Refresh()

    Dim idx as integer

    '' note the repeated calls to "package.elements.GetAt"
    '' in general you should make this call once and assign to a local
    '' variable - in the example below, EA will load the element required
    '' everytime a call is made - rather than loading once and keeping
    '' a local reference

    For idx = 0 to package.elements.count-1
        Console.WriteLine(package.elements.GetAt(idx).Name)
        If (package.elements.GetAt(idx).Name = "Login to Website" and_
            package.elements.GetAt(idx).Type = "UseCase") Then
            package.elements.deleteat(idx, false)
        EndIf
    Next
End Sub

```

15.1.2.10.5 Add a Connector

public Object

```

'' Add a connector and set values

Sub ConnectorTest

    Dim source as object
    Dim target as object
    Dim con as object
    Dim o as object

```

```

Dim client as object
Dim supplier as object

' 'use ElementID's to quickly load an element in this example
' '... you will need to find suitable ID's in your model

source = m_Repository.GetElementByID(129)
target = m_Repository.GetElementByID(169)

con = source.Connectors.AddNew("test link 2", "Association")

' 'again- replace ID with a suitable one from your model
con.SupplierID = 169

If not con.Update Then
    Console.WriteLine(con.GetLastError)
End If
source.Connectors.Refresh

Console.WriteLine("Connector Created")

o = con.Constraints.AddNew("constraint2", "type")
If not o.Update Then
    Console.WriteLine(o.GetLastError)
End If

o = con.TaggedValues.AddNew("Tag", "Value")
If not o.Update Then
    Console.WriteLine(o.GetLastError)
End If

' 'use the client and supplier ends to set
' 'additional information

client = con.ClientEnd
client.Visibility = "Private"
client.Role = "m_client"
client.Update
supplier = con.SupplierEnd
supplier.Visibility = "Protected"
supplier.Role = "m_supplier"
supplier.Update

Console.WriteLine("Client and Supplier set")

Console.WriteLine(client.Role)
Console.WriteLine(supplier.Role)

End Sub

```

15.1.2.10.6 Add and Manage Diagrams

public Object

```

' 'an example of how to create a diagram and add an element to it
' 'note the optional use of element rectangle setting using
' 'left, right, top and bottom dimensions in AddNew call

Sub DiagramLifeCycle

    Dim diagram as object
    Dim v as object
    Dim o as object
    Dim package as object

    Dim idx as Integer
    Dim idx2 as integer

```

```

package = m_Repository.GetPackageByID(5)

diagram = package.Diagrams.AddNew("Logical Diagram", "Logical")
If not diagram.Update Then
    Console.WriteLine(diagram.GetLastError)
Endif

diagram.Notes = "Hello there this is a test"
diagram.update()

o = package.Elements.AddNew("ReferenceType", "Class")
o.Update

' add element to diagram - supply optional rectangle co-ordinates
v = diagram.DiagramObjects.AddNew("l=200;r=400;t=200;b=600;", "")
v.ElementID = o.ElementID
v.Update

Console.WriteLine(diagram.DiagramID)

End Sub

```

15.1.2.10.7 Adding and Deleting Attributes and Methods

public Object

```

Dim element as object
Dim idx as integer
Dim attribute as object
Dim method as object

' just load an element by ID - you will need to
' substitute a valid ID from your model
element = m_Repository.GetElementByID(246)

' create a new method
method = element.Methods.AddNew("newMethod", "int")
method.Update
element.Methods.Refresh

' now loop through methods for Element - and delete our addition
For idx = 0 to element.Methods.Count - 1
    method = element.Methods.GetAt(idx)
    Console.WriteLine(method.Name)
    If (method.Name = "newMethod") Then
        element.Methods.Delete(idx)
    Endif
Next

' create an attribute
attribute = element.attributes.AddNew("NewAttribute", "int")
attribute.Update
element.attributes.Refresh

' loop through and delete our new attribute
For idx = 0 to element.attributes.Count - 1
    attribute = element.attributes.GetAt(idx)
    Console.WriteLine(attribute.Name)
    If (attribute.Name = "NewAttribute") Then
        element.attributes.Delete(idx)
    Endif
Next

```

15.1.2.10.8 Element Extras*public Object*

```

'' examples of how to access and use element extras - such as
'' scenarios, constraints and requirements

Sub ElementExtras

  Dim element as object
  Dim o as object
  Dim idx as Integer
  Dim bDel as boolean
  bDel = true

  try
    element = m_Repository.GetElementByID(129)

    'manage constraints for an element
    'demonstrate addnew and delete
    o = element.Constraints.AddNew("Appended", "Type")
    If not o.Update Then
      Console.WriteLine("Constraint error:" + o.GetLastError())
    Endif
    element.Constraints.Refresh
    For idx=0 to element.Constraints.Count -1
      o = element.Constraints.GetAt(idx)
      Console.WriteLine(o.Name)
      If (o.Name="Appended") Then
        If bDel Then element.Constraints.Delete(idx)
      Endif
    Next

    'efforts
    o = element.Efforts.AddNew("Appended", "Type")
    If not o.Update Then
      Console.WriteLine("Efforts error:" + o.GetLastError())
    Endif
    element.Efforts.Refresh
    For idx=0 to element.Efforts.Count -1
      o = element.Efforts.GetAt(idx)
      Console.WriteLine(o.Name)
      If (o.Name="Appended") Then
        If bDel Then element.Efforts.Delete(idx)
      Endif
    Next

    'Risks
    o = element.Risks.AddNew("Appended", "Type")
    If not o.Update Then
      Console.WriteLine("Risks error:" + o.GetLastError())
    Endif
    element.Risks.Refresh
    For idx=0 to element.Risks.Count -1
      o = element.Risks.GetAt(idx)
      Console.WriteLine(o.Name)
      If (o.Name="Appended") Then
        If bDel Then element.Risks.Delete(idx)
      Endif
    Next

    'Metrics
    o = element.Metrics.AddNew("Appended", "Change")
    If not o.Update Then
      Console.WriteLine("Metrics error:" + o.GetLastError())
    Endif
    element.Metrics.Refresh
    For idx=0 to element.Metrics.Count -1
      o = element.Metrics.GetAt(idx)

```

```
    Console.WriteLine(o.Name)
    If (o.Name="Appended") Then
        If bDel Then element.Metrics.Delete (idx)
    Endif
Next

'TaggedValues
o = element.TaggedValues.AddNew("Appended", "Change")
If not o.Update Then
    Console.WriteLine("TaggedValues error:" + o.GetLastError())
Endif
element.TaggedValues.Refresh
For idx = 0 to element.TaggedValues.Count -1
    o = element.TaggedValues.GetAt (idx)
    Console.WriteLine(o.Name)
    If (o.Name="Appended") Then
        If bDel Then element.TaggedValues.Delete (idx)
    Endif
Next

'Scenarios
o = element.Scenarios.AddNew("Appended", "Change")
If not o.Update Then
    Console.WriteLine("Scenarios error:" + o.GetLastError())
Endif
element.Scenarios.Refresh
For idx = 0 to element.Scenarios.Count -1
    o = element.Scenarios.GetAt (idx)
    Console.WriteLine(o.Name)
    If (o.Name="Appended") Then
        If bDel Then element.Scenarios.Delete (idx)
    Endif
Next

'Files
o = element.Files.AddNew("MyFile", "doc")
If not o.Update Then
    Console.WriteLine("Files error:" + o.GetLastError())
Endif
element.Files.Refresh
For idx = 0 to element.Files.Count -1
    o = element.Files.GetAt (idx)
    Console.WriteLine(o.Name)
    If (o.Name="MyFile") Then
        If bDel Then element.Files.Delete (idx)
    Endif
Next

'Tests
o = element.Tests.AddNew("TestPlan", "Load")
If not o.Update Then
    Console.WriteLine("Tests error:" + o.GetLastError())
Endif
element.Tests.Refresh
For idx = 0 to element.Tests.Count -1
    o = element.Tests.GetAt (idx)
    Console.WriteLine(o.Name)
    If (o.Name="TestPlan") Then
        If bDel Then element.Tests.Delete (idx)
    Endif
Next

'Defect
o = element.Issues.AddNew("Broken", "Defect")
If not o.Update Then
    Console.WriteLine("Issues error:" + o.GetLastError())
Endif
element.Issues.Refresh
For idx = 0 to element.Issues.Count -1
```

```

    o=element.Issues.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Broken") Then
        If bDel Then element.Issues.Delete(idx)
    Endif
Next

'Change
o=element.Issues.AddNew("Change","Change")
If not o.Update Then
    Console.WriteLine("Issues error:" + o.GetLastError())
Endif
element.Issues.Refresh
For idx=0 to element.Issues.Count-1
    o=element.Issues.GetAt(idx)
    Console.WriteLine(o.Name)
    If(o.Name="Change") Then
        If bDel Then element.Issues.Delete(idx)
    Endif
Next

catche as exception
    Console.WriteLine(element.Methods.GetLastError())
    Console.WriteLine(e)
End try

End Sub

```

15.1.2.10.9 RepositoryExtras

public Object

```

'' Examples of how to access repository
'' collections for system level information

Sub RepositoryExtras

    Dim o as object
    Dim idx as integer

    'issues
    o=m_Repository.Issues.AddNew("Problem","Type")
    If(o.Update=false) Then
        Console.WriteLine(o.GetLastError())
    Endif
    o=nothing
    m_Repository.Issues.Refresh
    For idx=0 to m_Repository.Issues.Count-1
        Console.WriteLine(m_Repository.Issues.GetAt(idx).Name)
        If(m_Repository.Issues.GetAt(idx).Name="Problem") then
            m_Repository.Issues.DeleteAt(idx,false)
            Console.WriteLine("Delete Issues")
        Endif
    Next

    ''tasks
    o=m_Repository.Tasks.AddNew("Task 1","Task type")
    If(o.Update=false) Then
        Console.WriteLine("error - " + o.GetLastError())
    Endif
    o=nothing
    m_Repository.Tasks.Refresh
    For idx=0 to m_Repository.Tasks.Count-1
        Console.WriteLine(m_Repository.Tasks.GetAt(idx).Name)
        If(m_Repository.Tasks.GetAt(idx).Name="Task 1") then
            m_Repository.Tasks.DeleteAt(idx,false)
            Console.WriteLine("Delete Tasks")
        Endif
    Next
End Sub

```



```
Endif
Next

    'glossary
o=m_Repository.Terms.AddNew("Term1","business")
If(o.Update=false) Then
    Console.WriteLine("error - "+o.GetLastError())
Endif
o=nothing
m_Repository.Terms.Refresh
For idx = 0 to m_Repository.Terms.Count-1
    Console.WriteLine(m_Repository.Terms.GetAt(idx).Term)
    If(m_Repository.Terms.GetAt(idx).Term = "Term1") then
        m_Repository.Terms.DeleteAt(idx,false)
        Console.WriteLine("Delete Terms")
    Endif
Next

'authors
o=m_Repository.Authors.AddNew("JoeB","Writer")
If(o.Update=false) Then
    Console.WriteLine(o.GetLastError())
Endif
o=nothing
m_Repository.Authors.Refresh
For idx = 0 to m_Repository.authors.Count-1
    Console.WriteLine(m_Repository.Authors.GetAt(idx).Name)
    If(m_Repository.authors.GetAt(idx).Name = "JoeB") then
        m_Repository.authors.DeleteAt(idx,false)
        Console.WriteLine("Delete Authors")
    Endif
Next

o=m_Repository.Clients.AddNew("Joe Sphere","Client")
If(o.Update=false) Then
    Console.WriteLine(o.GetLastError())
Endif
o=nothing
m_Repository.Clients.Refresh
For idx = 0 to m_Repository.Clients.Count-1
    Console.WriteLine(m_Repository.Clients.GetAt(idx).Name)
    If(m_Repository.Clients.GetAt(idx).Name = "Joe Sphere") then
        m_Repository.Clients.DeleteAt(idx,false)
        Console.WriteLine("Delete Clients")
    Endif
Next

o=m_Repository.Resources.AddNew("Joe Worker","Resource")
If(o.Update=false) Then
    Console.WriteLine(o.GetLastError())
Endif
o=nothing
m_Repository.Resources.Refresh
For idx = 0 to m_Repository.Resources.Count-1
    Console.WriteLine(m_Repository.Resources.GetAt(idx).Name)
    If(m_Repository.Resources.GetAt(idx).Name = "Joe Worker") then
        m_Repository.Resources.DeleteAt(idx,false)
        Console.WriteLine("Delete Resources")
    Endif
Next

End Sub
```

15.1.2.10.10 Stereotypes**public Object**

```

Sub TestStereotypes

    Dim o as object
    Dim idx as integer

    'add a new stereotype to the Stereotypes collection
    o = m_Repository.Stereotypes.AddNew("funky", "class")
    If (o.Update=false) Then
        Console.WriteLine(o.GetLastError())
    End if
    o = nothing

    'make sure we refresh
    m_Repository.Stereotypes.Refresh

    'then iterate through - deleting our new entry in the process
    For idx = 0 to m_Repository.Stereotypes.Count-1
        Console.WriteLine(m_Repository.Stereotypes.GetAt(idx).Name)
        If (m_Repository.Stereotypes.GetAt(idx).Name = "funky") then
            m_Repository.Stereotypes.DeleteAt(idx, false)
            Console.WriteLine("Delete element")
        End if
    Next

End Sub

```

15.1.2.10.11 Working with Attributes**public Object**

```

' an example of working with attributes

Sub AttributeLifecycle

    Dim element as object
    Dim o as object
    Dim t as object
    Dim idx as Integer
    Dim idx2 as integer
    try
        element = m_Repository.GetElementByID(129)

        For idx = 0 to element.Attributes.Count -1

            Console.WriteLine("attribute=" + element.Attributes.GetAt(idx).Name)

            o = element.Attributes.GetAt(idx)
            t = o.Constraints.AddNew(">123", "Precision")
            t.Update()
            o.Constraints.Refresh
            For idx2 = 0 to o.Constraints.Count-1
                t = o.Constraints.GetAt(idx2)
                Console.WriteLine("Constraint: " + t.Name)
                If (t.Name = ">123") Then
                    o.Constraints.DeleteAt(idx2, false)
                End if
            Next

            For idx2 = 0 to o.TaggedValues.Count-1
                t = o.TaggedValues.GetAt(idx2)
                If (t.Name = "Type2") Then
                    'Console.WriteLine("deleting")
                    o.TaggedValues.DeleteAt(idx2, true)
                End if
            Next
        End try
    End Sub

```

```

Next

    t = o.TaggedValues.AddNew("Type2", "Number")
t.Update
o.TaggedValues.Refresh
For idx2 = 0 to o.TaggedValues.Count-1
    t = o.TaggedValues.GetAt(idx2)
    Console.WriteLine("Tagged Value: " + t.Name)
Next

If(element.Attributes.GetAt(idx).Name = "m_Tootle") Then
    Console.WriteLine("deleteattribute")
    element.Attributes.DeleteAt(idx, false)
End If

Next

catch e as exception
    Console.WriteLine(element.Attributes.GetLastError())
    Console.WriteLine(e)
End try
End Sub

```

15.1.2.10.12 Working with Methods

public Object

```

''example of working with the Methods collection
''of an element - and with Method collections

Sub MethodLifeCycle

    Dim element as object
    Dim method as object
    Dim t as object
    Dim idx as Integer
    Dim idx2 as integer

    try
        element = m_Repository.GetElementByID(129)

        For idx = 0 to element.Methods.Count -1
            method = element.Methods.GetAt(idx)
            Console.WriteLine(method.Name)

            t = method.PreConditions.AddNew("TestConstraint", "something")
            If t.Update = false Then
                Console.WriteLine("PreConditions: " + t.GetLastError)
            Endif

            method.PreConditions.Refresh
            For idx2 = 0 to method.PreConditions.Count-1
                t = method.PreConditions.GetAt(idx2)
                Console.WriteLine("PreConditions: " + t.Name)
                If t.Name = "TestConstraint" Then
                    method.PreConditions.DeleteAt(idx2, false)
                End If
            Next

            t = method.PostConditions.AddNew("TestConstraint", "something")
            If t.Update = false Then
                Console.WriteLine("PostConditions: " + t.GetLastError)
            Endif

            method.PostConditions.Refresh
            For idx2 = 0 to method.PostConditions.Count-1

```

```

    t = method.PostConditions.GetAt(idx2)
    Console.WriteLine("PostConditions: " + t.Name)
    If t.Name = "TestConstraint" Then
        method.PostConditions.DeleteAt(idx2, false)
    End If
Next

    t = method.TaggedValues.AddNew("TestTaggedValue", "something")
    If t.Update = false Then
        Console.WriteLine("Tagged Values: " + t.GetLastError)
    End If

    For idx2 = 0 to method.TaggedValues.Count-1
        t = method.TaggedValues.GetAt(idx2)
        Console.WriteLine("Tagged Value: " + t.Name)
        If (t.Name = "TestTaggedValue") Then
            method.TaggedValues.DeleteAt(idx2, false)
        End If
    Next

    t = method.Parameters.AddNew("TestParam", "string")
    If t.Update = false Then
        Console.WriteLine("Parameters: " + t.GetLastError)
    End If

    method.Parameters.Refresh
    For idx2 = 0 to method.Parameters.Count-1
        t = method.Parameters.GetAt(idx2)
        Console.WriteLine("Parameter: " + t.Name)
        If (t.Name = "TestParam") Then
            method.Parameters.DeleteAt(idx2, false)
        End If
    Next

    method = nothing
Next
catche as exception
    Console.WriteLine(element.Methods.GetLastError())
    Console.WriteLine(e)
End try

End Sub

```

15.2 The Project Interface (XML)

Windows ActiveX Automation provides a way for applications to control and interface with Enterprise Architect.

Typically you use an automation capable client - such as Visual Basic or MS Word, and write a suitable program to open an Enterprise Architect model and iterate through the model contents. In this way you can write custom reports, code generators and other applications which draw on the information in an Enterprise Architect project.

This section will describe the interfaces - or methods - you will use to script Enterprise Architect.

See Also

- [The Read Only Interface](#)
- [Automation Interfaces](#)
- [Opening a Project](#)
- [Enumerating Views](#)
- [Enumerating](#)
- [Enumerating Diagrams](#)
- [Get Diagrams](#)
- [Enumerating Elements](#)

- [Get Element](#)
- [Enumerating Links](#)
- [Get Link](#)
- [Get Diagram Image](#)

15.2.1 The Read Only Interface

This section describes the read only interface to EA. Using this you can retrieve any information required from the model using clients such as Visual Basic.

15.2.2 Automation Interfaces

An interface is a scripting method supported by Enterprise Architect that you can use to retrieve information about a model. To use these interfaces, you need an automation client, such as Visual Basic or MS Word.

The *Automation Interface* to read information from Enterprise Architect is defined as follows:

Project Interfaces

```
BOOL LoadProject (Const Variant FAR& FileName);  
BOOL ReloadProject ();  
void ShowWindow (Long Show);  
void Exit();
```

List Interfaces

```
BSTR EnumPackages (Const Variant FAR& PackageGUID);  
BSTR EnumElements (Const Variant FAR& PackageGUID);  
BSTR EnumLinks (Const Variant FAR& PackageID);  
BSTR EnumDiagrams (Const Variant FAR& PackageGUID);  
BSTR EnumDiagramElements (Const Variant FAR& DiagramGUID);  
BSTR EnumDiagramLinks (Const Variant FAR& DiagramID);  
BSTR EnumViews ();
```

Diagram Interfaces

```
BOOL LoadDiagram (Const Variant FAR& DiagramGUID);  
BSTR SaveDiagramImageToFile (Const Variant FAR& FileName);  
BOOL PutDiagramImageOnClipboard (Const Variant FAR& DiagramGUID, Long Type);  
BOOL PutDiagramImageToFile (Const Variant FAR& DiagramGUID, Const Variant FAR& Filename, Long Type);
```

Retrieval Interfaces

```
BSTR GetElement (Const Variant FAR& ElementGUID);  
BSTR GetLink (Const Variant FAR& LinkGUID);  
BSTR GetDiagram (Const Variant FAR& DiagramGUID);  
BSTR GetElementConstraints (Const Variant FAR& ElementGUID);  
BSTR GetElementEffort (Const Variant FAR& ElementGUID);  
BSTR GetElementMetrics (Const Variant FAR& ElementGUID);  
BSTR GetElementFiles (Const Variant FAR& ElementGUID);  
BSTR GetElementRequirements (Const Variant FAR& ElementGUID);  
BSTR GetElementProblems (Const Variant FAR& ElementGUID);  
BSTR GetElementResources (Const Variant FAR& ElementGUID);  
BSTR GetElementRisks (Const Variant FAR& ElementGUID);  
BSTR GetElementScenarios (Const Variant FAR& ElementGUID);  
BSTR GetElementTests (Const Variant FAR& ElementGUID);
```

15.2.3 Opening a Project

The first task when using automation to open work with a model is to create the automation server.

When you run EA normally, it registers itself as an automation server and makes entries into the Windows registry for other programs to find it. The following code fragment from Visual Basic shows how to create an EA automation server: Once loaded you can iterate through the model using the EnumXXX functions described later.

Visual Basic Example Code to Open an EA Model

```
OptionExplicit
'-----
'This model demonstrates how to invoke and read from
'an EA project using Automation scripting
'The scripting is based on XMI 1.1 compliant XML
'with the addition of enumeration calls (also XML based)

'the project object
Private EAProject As Object

Private Sub CMDLoad_Click()

    'create the one and only EAProject
    Set EAProject = CreateObject("EA.Project")

    'load up with specified path
    EAProject.LoadProject("C:\Projects\AutomationExample.EAP")

    'optionally hide window
    EAProject.ShowWindow(0)

End Sub
```

15.2.4 Enumerating Views

The root packages of an Enterprise Architect project are its views. These include the Use Case View, the Logical View and the Physical View - as well as custom views created by analysts during modeling.

To work with a model, the usual starting point is to get a list of the defined views - depending on what is required, you can then iterate through all sub-packages for each or only specialised views.

EA provides the interface BSTR EnumViews() for retrieving a list of views. The returned XML includes a node for each node - detailing the view name and its universal identifier (GUID). You can use the GUID to retrieve the contents of any view.

Example XML from EnumViews

```
<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <View>
    <Name>Use Case View</Name>
    <GUID>EAID_DAF38564_68F1_4929_8760_8DDF15614F77</GUID>
  </View>
  <View>
    <Name>Dynamic View</Name>
    <GUID>EAID_DE77A45B_726E_468f_83AB_86C0391CC126</GUID>
  </View>
  <View>
    <Name>Logical View</Name>
    <GUID>EAID_A634A9A5_5589_44e9_B9D6_D1D4BC5EC544</GUID>
  </View>
```

```

</View>
<View>
  <Name>Component View</Name>
  <GUID>EAID_3BA53453_8B77_4798_A0B5_DC02671804B7</GUID>
</View>
<View>
  <Name>Deployment View</Name>
  <GUID>EAID_A7E4B27B_DB98_4ef8_AE44_D58F389146F0</GUID>
</View>
<View>
  <Name>Custom</Name>
  <GUID>EAID_BE0B8A1C_2F09_4b8a_9FA3_015EA8E66056</GUID>
</View>
</Document>

```

Sample Visual Basic Code to Enumerate through Views

```

Private Sub CMDGetViews_Click()

  ' Demonstrates how to iterate through the top level
  ' view of an Enterprise Architect project and
  ' access information on packages and elements contained therein
  ' All access is XML based, so create some documents and nodes

  Dim xmlDoc As New MSXML2.DOMDocument
  Dim xmlPkg As New MSXML2.DOMDocument
  Dim xmlView As MSXML2.IXMLDOMNode
  Dim xmlPkgNode As MSXML2.IXMLDOMNode

  List1.Clear
  List2.Clear

  ' first get the View list
  xmlDoc.LoadXML EAProject.EnumViews

  ' Debug.Print xmlDoc.xml

  ' While there are views, iterate through and retrieve sub information
  Set xmlView = xmlDoc.SelectSingleNode("Document/View")
  Do While (Not xmlView Is Nothing)
    AddToTreeList xmlDoc.SelectSingleNode("Name").Text

    ' load the list of packages for this view
    xmlPkg.LoadXML EAProject.EnumPackages(xmlView.SelectSingleNode("GUID").Text)
    Debug.Print xmlPkg.xml

    Set xmlPkgNode = xmlPkg.SelectSingleNode("Document/Package")

    ' while there are packages to process
    Do While (Not xmlPkgNode Is Nothing)
      AddToTreeList " " & xmlPkgNode.SelectSingleNode("Name").Text
      GetPackage xmlPkgNode.SelectSingleNode("GUID").Text, Indent
      ' if this is class model go a bit deeper
      ' If (xmlPkgNode.SelectSingleNode("Name").Text = 'Class Model') Then
      '   GetElements xmlPkgNode.SelectSingleNode("GUID").Text
      ' End If
      Set xmlPkgNode = xmlPkgNode.NextSibling
    Loop

    Set xmlView = xmlView.NextSibling
  Loop

  ' finished
  AddToDebugList "Complete"

End Sub

```

15.2.5 Enumerating

Once you have obtained a list of views, you can recursively iterate through the packages within each view using the *EnumPackages* interface to obtain a list of child packages for each parent package/view.

EA provides the *EnumPackages(const VARIANT FAR& PackageGUID)* interface to get an XML representation of all child packages of another package/view. The XML contains a node for each package - and each node contains a Name and a GUID element. The GUID element is used to get more information on a package (including listing packages within a package).

The following is an example of the XML returned by this interface:

```
<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Package>
    <Name>UC01: Use Case Model</Name>
    <GUID>EAID_508A37AC_5B27_4b98_8BBE_81D6AA03E8B4</GUID>
  </Package>
</Document>
```

The following Visual Basic code illustrates how to access and use the EnumPackages interface:

```
Private Sub GetPackage(PackageGUID As String, Offset As String)

    Dim xmlDoc As New MSXML2.DOMDocument
    Dim xmlPkg As New MSXML2.DOMDocument
    Dim xmlView As MSXML2.IXMLDOMNode
    Dim xmlPkgNode As MSXML2.IXMLDOMNode

    'list all diagrams in this package
    GetDiagrams PackageGUID, Offset

    'display all elements in this package
    GetElements PackageGUID, Offset

    'then load the list of packages for this package
    xmlPkg.loadXML EAProject.EnumPackages(PackageGUID)
    Set xmlPkgNode = xmlPkg.selectSingleNode("Document/Package")

    'while there are packages to process
    Do While (Not xmlPkgNode Is Nothing)
        AddToTreeList " " & xmlPkgNode.selectSingleNode("Name").Text
        GetPackage xmlPkgNode.selectSingleNode("GUID").Text, Indent + Offset
        Set xmlPkgNode = xmlPkgNode.nextSibling
    Loop

End Sub
```

15.2.6 Enumerating Diagrams

You can retrieve a list of all diagrams within a package using the *EnumDiagrams* interface. This interface will return an XML list of all diagrams - including the name, type and universal identifier (GUID). You can use the GUID to retrieve more information about the diagram - and to either copy the diagram image to file or clipboard.

The *EnumDiagrams* interface is defined as *EnumDiagrams(const VARIANT FAR& PackageGUID)* - it returns an XML list of diagrams within a package - including name, type and GUID.

Example of XML Returned by Interface

```
<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Diagram>
    <Name>Overview</Name>
    <Type>Use Case</Type>
    <GUID>EAID_DA388E78_CF6B_4331_932A_565F1AADE18A</GUID>
  </Diagram>
</Document>
```

Example Visual Basic code which uses the EnumDiagrams interface

```
Private Sub GetDiagrams (PackageGUID As String, Offset As String)

  Dim xmlNode As MSXML2.IXMLDOMNode
  Dim xmlDoc As New MSXML2.DOMDocument
  Dim str As String

  'get list of elements in package
  xmlDoc.loadXML EAPProject.EnumDiagrams (PackageGUID)

  ' Debug.Print xmlDoc.xml

  Set xmlNode = xmlDoc.selectSingleNode("Document/Diagram")
  Do While (Not xmlNode Is Nothing)
    'add to list
    str = xmlNode.selectSingleNode("Name").Text + ":" + xmlNode.selectSingleNode("Type").Text
    AddToTreeList Offset + Indent + "Diagram - " + str
    GetDiagram xmlDoc.selectSingleNode("GUID").Text

    'go to next diagram
    Set xmlNode = xmlNode.nextSibling
  Loop

End Sub
```

15.2.7 Get Diagrams

Use this interface to retrieve information about a diagram - including, name, type, characteristics and a list of elements within the diagram. This interface requires you to pass in a diagram identifier (GUID) - typically retrieved from the EnumDiagrams interface, and returns an XML document detailing diagram attributes.

The interface is defined as GetDiagram(const VARIANT FAR& DiagramGUID)

An example of the XML returned by this interface is shown below:

```
<Document xmlns:UML="omg.org/UML1.3">
  <Diagram>
    <UML:Diagram name="Overview" xmi.id="EAID_DA388E78_CF6B_4331_932A_565F1AADE18A"
    diagramType="UseCaseDiagram" owner="owner" toolName="Enterprise Architect 2.5">
      <UML:ModelElement, taggedValue>
        <UML:TaggedValue tag="documentation" value="This diagram provides an overview of the Customer's
        available activities in the proposed system - from logging on to browsing the book store, selecting items and making
        purchases." />
        <UML:TaggedValue tag="version" value="1.0" />
        <UML:TaggedValue tag="author" value="Geoffrey Sparks" />
        <UML:TaggedValue tag="created_date" value="6/30/2001" />
        <UML:TaggedValue tag="modified_date" value="6/30/2001" />
        <UML:TaggedValue tag="package" value="EAPK_508A37AC_5B27_4b98_8BBE_81D6AA03E8B4" />
        <UML:TaggedValue tag="documentation" value="This diagram provides an overview of the Customer's
        available activities in the proposed system - from logging on to browsing the book store, selecting items and making
        purchases." />
      </UML:ModelElement, taggedValue>
    </UML:Diagram>
  </Diagram>
</Document>
```

```

    <UML:TaggedValue tag="type" value="Use Case" />
    <UML:TaggedValue tag="EASStyle" value="ShowPrivate=1;ShowProtected=1;ShowPublic=1;
HideRelationships=0;Locked=0;Border=1;HighlightForeign=1;PackageContents=1;SequenceNotes=0;
ScalePrintImage=0;DocSize.cx=826;DocSize.cy=1169;ShowDetails=0;Orientation=P;Zoom=100;ShowTags=1;
OpParams=1;" />
  </UML:ModelElement.taggedValue>

  <UML:Diagram.element>
    <UML:DiagramElement geometry="Left=32;Top=49;Right=222;Bottom=125;"
subject="EAID_942CA067_D717_433d_85DB_87A4A4B28660" seqno="1" />
    <UML:DiagramElement geometry="Left=124;Top=411;Right=367;Bottom=499;"
subject="EAID_D4597C98_506F_421a_AC73_75E9B3619602" seqno="2" />
    <UML:DiagramElement geometry="Left=291;Top=207;Right=336;Bottom=297;"
subject="EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2" seqno="3" />
    <UML:DiagramElement geometry="SX=0;SY=0;EX=0;EY=0;"
subject="EAID_4E742B84_D86E_4e37_B91C_8D3E25299646" style="Mode=1;;Hidden=0;" />
    <UML:DiagramElement geometry="SX=0;SY=0;EX=0;EY=0;"
subject="EAID_0D03CF69_61BF_4041_9CD8_540901F171F9" style="Mode=1;;Hidden=0;" />
  </UML:Diagram.element>
</UML:Diagram>
</Diagram>
</Document>

```

The following Visual Basic code illustrates the use of this interface

```

Private Sub GetDiagram(DiagramGUID As String)
On Error GoTo errDiagram

'get a diagram from the model - includes
'all attributes and tagged values

Dim xmlNode As MSXML2.IXMLDOMNode
Dim xmlDiagram As New MSXML2.DOMDocument
Dim xmlDoc As New MSXML2.DOMDocument
Dim xmlTagged As MSXML2.IXMLDOMNode
Dim xmlElement As MSXML2.IXMLDOMNode

Dim n As Integer

EAPProject.PutDiagramImageOnClipboard DiagramGUID, 0

xmlDiagram.loadXML EAPProject.GetDiagram(DiagramGUID)

' Debug.Print xmlDiagram.xml

Set xmlNode = xmlDiagram.selectSingleNode("Document/Diagram")

'go to first element - will be the actual diagram (eg. UML:StateChart)
Set xmlNode = xmlNode.firstChild()

AddToDebugList "-----"
AddToDebugList xmlNode.nodeName

'get attributes first
For n = 0 To xmlNode.Attributes.length - 1
    AddToDebugList xmlNode.Attributes.item(n).nodeName + " = " + xmlNode.Attributes.item(n).Text
Next n

' then get tagged values
Set xmlTagged = xmlNode.selectSingleNode("UML:ModelElement.taggedValue/UML:TaggedValue")
Do While (Not xmlTagged Is Nothing)
    AddToDebugList xmlTagged.Attributes(0).Text + " = " + xmlTagged.Attributes(1).Text
    Set xmlTagged = xmlTagged.nextSibling
Loop

'now get diagram elements
Set xmlElement = xmlNode.selectSingleNode("UML:Diagram.element/UML:DiagramElement")
Do While (Not xmlElement Is Nothing)

```

```

'getattributesfirst
AddToDebugList "... DiagramElement ..."
For n = 0 To xmlElement.Attributes.Length - 1
    AddToDebugList xmlElement.Attributes.item(n).nodeName + " =: " + xmlElement.Attributes.item(n).Text
Next n
Set xmlElement = xmlElement.nextSibling
Loop

Exit Sub

errDiagram:
AddToDebugList "ERROR: " + Error$

End Sub

```

15.2.8 Enumerating Elements

Once you have a package GUID, you can pass it into the EnumElements interface to get a list of elements within a package. An element is any kind of UML object - such as a Class or Object or Activity. The EnumElements interface returns XML that lists the name, type and GUID for each contained element. From this you can use the element GUID to obtain more information about each element.

The interface is defined as *EnumElements(const VARIANT FAR& PackageGUID)* - and returns an XML string.

Example XML returned by EnumElements

```

<Document xmlns:UML="omg.org/UML1.3">
<Element>
    <Name>Customer</Name>
    <Type>Actor</Type>
    <GUID>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</GUID>
</Element>
<Element>
    <Name>Note</Name>
    <Type>Note</Type>
    <GUID>EAID_D4597C98_506F_421a_AC73_75E9B3619602</GUID>
</Element>
<Element>
    <Name>UC01-1: User Management</Name>
    <Type>Package</Type>
    <GUID>EAID_942CA067_D717_433d_85DB_87A4A4B28660</GUID>
</Element>
</Document>

```

Example Visual Basic code used to enumerate elements

```

Private Sub GetElements(PackageGUID As String, Offset As String)

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xmlDoc As New MSXML2.DOMDocument
    Dim str As String

    'get list of elements in package
    xmlDoc.loadXML EAPProject.EnumElements(PackageGUID)

    ' Debug.Print xmlDoc.xml

    Set xmlNode = xmlDoc.selectSingleNode("Document/Element")
    Do While (Not xmlNode Is Nothing)
        'add to list
        str = xmlNode.selectSingleNode("Name").Text + " : " + xmlNode.selectSingleNode("Type").Text
        AddToTreeList Offset + Indent + str
    Loop

```

```

    GetElement xmlNode.selectSingleNode( "GUID" ).Text
'getdetails
    GetLinks xmlNode.selectSingleNode( "GUID" ).Text , Offset + Indent

    'go to next element
    Set xmlNode = xmlNode.nextSibling
Loop

End Sub

```

15.2.9 Get Element

To retrieve information about an element in EA, use the GetElement interface. This takes an Element GUID previously obtained using one of the enumeration interfaces, and returns an XML document containing all element details. For class elements, this will also include Operation and Attribute details.

The interface is defined as *GetElement(const VARIANT FAR& ElementGUID)* - it takes an element GUID and returns an XML document.

An example XML element document is given below

```

<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Element>
    <UML:Class name="Customer" xmi.id="EAID_D5663F5E_D116_4767_A13B_F9EE25BCFF9E" visibility="public"
namespace="EAPK_C14247A1_6C15_4b5a_9AFC_98A4A5611138" isRoot="false" isLeaf="false" isAbstract="true">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="ea_stype" value="Class" />
        <UML:TaggedValue tag="ea_ntype" value="0" />
        <UML:TaggedValue tag="version" value="1.0" />
        <UML:TaggedValue tag="package" value="EAPK_C14247A1_6C15_4b5a_9AFC_98A4A5611138" />
        <UML:TaggedValue tag="date_created" value="9/30/2000" />
        <UML:TaggedValue tag="date_modified" value="9/2/2001" />
        <UML:TaggedValue tag="genfile" value="C:\Documents and Settings\Administrator\Desktop\Customer.cls" />
        <UML:TaggedValue tag="gentype" value="Visual Basic" />
        <UML:TaggedValue tag="tagged" value="0" />
        <UML:TaggedValue tag="package_name" value="LM01-1: Customer Domain" />
        <UML:TaggedValue tag="phase" value="1.0" />
        <UML:TaggedValue tag="author" value="Geoffrey Sparks" />
        <UML:TaggedValue tag="complexity" value="1" />
        <UML:TaggedValue tag="documentation" value="A customer class. Contains attributes and behavior
corresponding to a customer of the on-line bookstore. A customer has a current account with the book store and
preferred shipping methods." />
        <UML:TaggedValue tag="status" value="Approved" />
        <UML:TaggedValue tag="style" value="BackColor=-1;BorderColor=-1;BorderWidth=-1;FontColor=-1;
VSwimLanes=0;HSwimLanes=0;BorderStyle=0;" />
      </UML:ModelElement.taggedValue>

      <UML:Classifier.feature>
        <UML:Attributename="Account" visibility="private" ownerScope="instance" changeable="none"
targetScope="instance">
          <UML:Attribute.initialValue>
            <UML:Expression />
          </UML:Attribute.initialValue>
          <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="CustomerAccount" />
            <UML:TaggedValue tag="derived" value="0" />
            <UML:TaggedValue tag="containment" value="Not Specified" />
            <UML:TaggedValue tag="length" value="0" />
            <UML:TaggedValue tag="ordered" value="0" />
            <UML:TaggedValue tag="precision" value="0" />
            <UML:TaggedValue tag="scale" value="0" />
            <UML:TaggedValue tag="collection" value="false" />
            <UML:TaggedValue tag="position" value="0" />
          </UML:ModelElement.taggedValue>
        </UML:Classifier.feature>
      </UML:Class>
    </Element>
  </Document>

```

```

    <UML:TaggedValue tag="description" value="The customer account object" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="genoption" value="PROPERTY=Account;" />
    <UML:TaggedValue tag="scope" value="Private" />
  </UML:ModelElement.taggedValue>

  <UML:ModelElement.constraint>
    <UML:Constraint name="Not null">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="Pre-Condition" />
        <UML:TaggedValue tag="documentation" value="Customer account cannot be null" />
      </UML:ModelElement.taggedValue>
    </UML:Constraint>
  </UML:ModelElement.constraint>
</UML:Attribute>

  <UML:AttributeName="Address" visibility="private" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="string" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="containment" value="Not Specified" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="1" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="genoption" value="PROPERTY=DED;" />
    <UML:TaggedValue tag="scope" value="Private" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:AttributeName="City" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="string" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="containment" value="Not Specified" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="static" value="1" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="2" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:AttributeName="Country" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="Country" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />

```

```

    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="3" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:Attribute name="CustomerID" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="string" />
  <UML:TaggedValue tag="derived" value="0" />
  <UML:TaggedValue tag="length" value="0" />
  <UML:TaggedValue tag="ordered" value="0" />
  <UML:TaggedValue tag="precision" value="0" />
  <UML:TaggedValue tag="scale" value="0" />
  <UML:TaggedValue tag="collection" value="false" />
  <UML:TaggedValue tag="position" value="4" />
  <UML:TaggedValue tag="description" value="Unique identifier for a customer. Used internally only." />
  <UML:TaggedValue tag="duplicates" value="0" />
  <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:Attribute name="FirstName" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="string" />
  <UML:TaggedValue tag="derived" value="0" />
  <UML:TaggedValue tag="length" value="0" />
  <UML:TaggedValue tag="ordered" value="0" />
  <UML:TaggedValue tag="precision" value="0" />
  <UML:TaggedValue tag="scale" value="0" />
  <UML:TaggedValue tag="collection" value="false" />
  <UML:TaggedValue tag="position" value="5" />
  <UML:TaggedValue tag="duplicates" value="0" />
  <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:Attribute name="LastLogin" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="string" />
  <UML:TaggedValue tag="derived" value="0" />
  <UML:TaggedValue tag="length" value="0" />
  <UML:TaggedValue tag="ordered" value="0" />
  <UML:TaggedValue tag="precision" value="0" />
  <UML:TaggedValue tag="scale" value="0" />
  <UML:TaggedValue tag="collection" value="false" />
  <UML:TaggedValue tag="position" value="6" />
  <UML:TaggedValue tag="description" value="The last time this user logged in. Display when user logs in" />
  <UML:TaggedValue tag="duplicates" value="0" />
  <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

```

```

<UML:AttributeName="Login" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="string" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="7" />
    <UML:TaggedValue tag="description" value="The user login in" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

<UML:Attribute name="Password" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="DateTime" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="8" />
    <UML:TaggedValue tag="description" value="The user's password. " />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>

  <UML:ModelElement.constraint>
    <UML:Constraint name="&gt;i=8 Characters">
      <UML:ModelElement.taggedValue />
    </UML:Constraint>
  </UML:ModelElement.constraint>
</UML:Attribute>

<UML:Attribute name="Preferences" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="CustomerPreferences" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="9" />
    <UML:TaggedValue tag="description" value="A Customer preferences object" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

<UML:Attribute name="Surname" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">

```

```

<UML:Attribute.initialValue>
  <UML:Expression />
</UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="string" />
  <UML:TaggedValue tag="derived" value="0" />
  <UML:TaggedValue tag="length" value="0" />
  <UML:TaggedValue tag="ordered" value="0" />
  <UML:TaggedValue tag="precision" value="0" />
  <UML:TaggedValue tag="scale" value="0" />
  <UML:TaggedValue tag="collection" value="false" />
  <UML:TaggedValue tag="position" value="10" />
  <UML:TaggedValue tag="duplicates" value="0" />
  <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:Attribute name="Zip" visibility="public" ownerScope="instance" changeable="none"
targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="Zip" />
  <UML:TaggedValue tag="derived" value="0" />
  <UML:TaggedValue tag="length" value="0" />
  <UML:TaggedValue tag="ordered" value="0" />
  <UML:TaggedValue tag="precision" value="0" />
  <UML:TaggedValue tag="scale" value="0" />
  <UML:TaggedValue tag="collection" value="false" />
  <UML:TaggedValue tag="position" value="11" />
  <UML:TaggedValue tag="duplicates" value="0" />
  <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

  <UML:Operation name="Account" visibility="public" ownerScope="instance" isQuery="false"
concurrency="sequential">
  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="type" value="CustomerAccount" />
  <UML:TaggedValue tag="const" value="false" />
  <UML:TaggedValue tag="static" value="1" />
  <UML:TaggedValue tag="stereotype" value="property get" />
  <UML:TaggedValue tag="synchronised" value="0" />
  <UML:TaggedValue tag="position" value="0" />
  <UML:TaggedValue tag="pure" value="0" />
  </UML:ModelElement.taggedValue>
</UML:Operation>

  <UML:Operation name="Account" visibility="public" ownerScope="instance" isQuery="false"
concurrency="sequential">
  <UML:ModelElement.taggedValue>
  <UML:TaggedValue tag="const" value="false" />
  <UML:TaggedValue tag="static" value="1" />
  <UML:TaggedValue tag="stereotype" value="property let" />
  <UML:TaggedValue tag="synchronised" value="0" />
  <UML:TaggedValue tag="position" value="0" />
  <UML:TaggedValue tag="pure" value="0" />
  </UML:ModelElement.taggedValue>

  <UML:BehavioralFeature.parameter>
  <UML:Parameter name="NewVal" visibility="public">
    <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="pos" value="0" />
    <UML:TaggedValue tag="type" value="CustomerAccount" />
    <UML:TaggedValue tag="const" value="0" />
    </UML:ModelElement.taggedValue>

    <UML:Parameter.defaultValue>

```



```

        <UML:Expression />
    </UML:Parameter.defaultValue>
</UML:Parameter>
</UML:BehavioralFeature.parameter>
</UML:Operation>

<UML:Operation name="AddCustomer" visibility="public" ownerScope="instance" isQuery="false"
concurrency="sequential">
    <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="bool" />
    <UML:TaggedValue tag="const" value="false" />
    <UML:TaggedValue tag="static" value="1" />
    <UML:TaggedValue tag="synchronised" value="0" />
    <UML:TaggedValue tag="concurrency" value="Sequential" />
    <UML:TaggedValue tag="position" value="0" />
    <UML:TaggedValue tag="returnarray" value="0" />
    <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>

    <UML:BehavioralFeature.parameter>
    <UML:Parameter name="Name" visibility="public">
        <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="pos" value="1" />
        <UML:TaggedValue tag="type" value="String" />
        <UML:TaggedValue tag="const" value="0" />
        </UML:ModelElement.taggedValue>

        <UML:Parameter.defaultValue>
        <UML:Expression />
    </UML:Parameter.defaultValue>
    </UML:Parameter>

    <UML:Parameter name="AccountID" kind="in" visibility="public">
        <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="pos" value="0" />
        <UML:TaggedValue tag="note" value="The default customer account ID" />
        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="const" value="0" />
        </UML:ModelElement.taggedValue>

        <UML:Parameter.defaultValue>
        <UML:Expression />
    </UML:Parameter.defaultValue>
    </UML:Parameter>
    </UML:BehavioralFeature.parameter>
</UML:Operation>

<UML:Operation name="DeleteCustomer" visibility="public" ownerScope="instance" isQuery="false"
concurrency="sequential">
    <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="bool" />
    <UML:TaggedValue tag="const" value="false" />
    <UML:TaggedValue tag="static" value="1" />
    <UML:TaggedValue tag="synchronised" value="0" />
    <UML:TaggedValue tag="concurrency" value="Sequential" />
    <UML:TaggedValue tag="position" value="0" />
    <UML:TaggedValue tag="returnarray" value="0" />
    <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>
</UML:Operation>

<UML:Operation name="GetAccount" visibility="public" ownerScope="instance" isQuery="false"
concurrency="sequential">
    <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="CustomerAccount" />
    <UML:TaggedValue tag="const" value="false" />
    <UML:TaggedValue tag="static" value="1" />
    <UML:TaggedValue tag="synchronised" value="0" />
    <UML:TaggedValue tag="concurrency" value="Sequential" />
    <UML:TaggedValue tag="position" value="0" />
    <UML:TaggedValue tag="returnarray" value="0" />

```

```

    <UML:TaggedValue tag="pure" value="0" />
  </UML:ModelElement.taggedValue>

  <UML:BehavioralFeature.parameter>
  <UML:Parameter name="dsd" visibility="public">
    <UML:ModelElement.taggedValue>
      <UML:TaggedValue tag="pos" value="0" />
      <UML:TaggedValue tag="type" value="Functional" />
      <UML:TaggedValue tag="const" value="0" />
    </UML:ModelElement.taggedValue>

    <UML:Parameter.defaultValue>
      <UML:Expression body="sd" />
    </UML:Parameter.defaultValue>
  </UML:Parameter>
</UML:BehavioralFeature.parameter>
</UML:Operation>

  <UML:Operation name="GetCustomerAsXML" visibility="public" ownerScope="instance" isQuery="false"
  concurrency="sequential">
    <UML:ModelElement.taggedValue>
      <UML:TaggedValue tag="type" value="string" />
      <UML:TaggedValue tag="const" value="false" />
      <UML:TaggedValue tag="static" value="1" />
      <UML:TaggedValue tag="synchronised" value="0" />
      <UML:TaggedValue tag="concurrency" value="Sequential" />
      <UML:TaggedValue tag="position" value="0" />
      <UML:TaggedValue tag="returnarray" value="0" />
      <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>
  </UML:Operation>

  <UML:Operation name="GetPreferences" visibility="public" ownerScope="instance" isQuery="false"
  concurrency="sequential">
    <UML:ModelElement.taggedValue>
      <UML:TaggedValue tag="type" value="CustomerPreferences" />
      <UML:TaggedValue tag="const" value="false" />
      <UML:TaggedValue tag="static" value="1" />
      <UML:TaggedValue tag="synchronised" value="0" />
      <UML:TaggedValue tag="concurrency" value="Sequential" />
      <UML:TaggedValue tag="position" value="0" />
      <UML:TaggedValue tag="returnarray" value="0" />
      <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>
  </UML:Operation>

  <UML:Operation name="UpdateCustomer" visibility="public" ownerScope="instance" isQuery="false"
  concurrency="sequential">
    <UML:ModelElement.taggedValue>
      <UML:TaggedValue tag="type" value="bool" />
      <UML:TaggedValue tag="const" value="false" />
      <UML:TaggedValue tag="static" value="1" />
      <UML:TaggedValue tag="synchronised" value="0" />
      <UML:TaggedValue tag="concurrency" value="Sequential" />
      <UML:TaggedValue tag="position" value="0" />
      <UML:TaggedValue tag="returnarray" value="0" />
      <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>

    <UML:BehavioralFeature.parameter>
    <UML:Parameter name="Fred" visibility="public">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="pos" value="0" />
        <UML:TaggedValue tag="type" value="xml" />
        <UML:TaggedValue tag="const" value="0" />
      </UML:ModelElement.taggedValue>
      <UML:Parameter.defaultValue>
        <UML:Expression />
      </UML:Parameter.defaultValue>
    </UML:Parameter>
  </UML:BehavioralFeature.parameter>

```

```

</UML:Operation>

<UML:Operation name="ValidateCustomer" visibility="public" ownerScope="instance" isQuery="false"
concurrency="sequential">
  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="void" />
    <UML:TaggedValue tag="const" value="false" />
    <UML:TaggedValue tag="static" value="1" />
    <UML:TaggedValue tag="synchronised" value="0" />
    <UML:TaggedValue tag="concurrency" value="Sequential" />
    <UML:TaggedValue tag="position" value="0" />
    <UML:TaggedValue tag="returnarray" value="0" />
    <UML:TaggedValue tag="pure" value="0" />
  </UML:ModelElement.taggedValue>
</UML:Operation>
</UML:Classifier.feature>
</UML:Class>
</Element>
</Document>

```

An example of Visual Basic code using this interface is given below - note that it includes calls to get element extensions - such as tests and resources

```

Private Sub GetElement (ObjectGUID As String)
On Error GoTo errElement

'get an element from the model - includes
'all attributes and tagged value
'followed by element extensions - such as constraints, tests etc.

Dim xmlNode As MSXML2.IXMLDOMNode
Dim xmlElement As New MSXML2.DOMDocument
Dim xmlDoc As New MSXML2.DOMDocument
Dim xmlTagged As MSXML2.IXMLDOMNode
Dim n As Integer

xmlElement.loadXML EAProject.GetElement (ObjectGUID)

Set xmlNode = xmlElement.selectSingleNode ("Document/Element")

'goto first element - will be the actual element (eg. UML:Actor)
Set xmlNode = xmlNode.firstChild()

'If (xmlNode.nodeName = 'UML:Class') Then Debug.Print vbCrLf + xmlElement.xml + vbCrLf

AddToDebugList "-----"
AddToDebugList xmlNode.nodeName

'get attributes first
For n = 0 To xmlNode.Attributes.length - 1
  AddToDebugList xmlNode.Attributes.item(n).nodeName + " = " + xmlNode.Attributes.item(n).Text
Next n

'get operations and attributes
GetFeatures xmlNode

'then get tagged values
Set xmlTagged = xmlNode.selectSingleNode ("UML:ModelElement.taggedValue/UML:TaggedValue")
Do While (Not xmlTagged Is Nothing)
  AddToDebugList xmlTagged.Attributes(0).Text + " = " + xmlTagged.Attributes(1).Text
  Set xmlTagged = xmlTagged.nextSibling
Loop

'now process extensions

xmlDoc.loadXML (EAProject.GetElementScenarios (ObjectGUID))
Call GetExtension (xmlDoc, "Document/Element/EAModel.scenario", "EAScenario")

```

```

xmlDoc.loadXML (EAPProject.GetElementRequirements(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/UML:ModelElement.requirement", "UML:Dependency")

xmlDoc.loadXML (EAPProject.GetElementConstraints(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/UML:ModelElement.Constraint", "UML:Constraint")

xmlDoc.loadXML (EAPProject.GetElementEffort(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.effort", "EAEffort")

xmlDoc.loadXML (EAPProject.GetElementMetrics(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.metric", "EAMetric")

xmlDoc.loadXML (EAPProject.GetElementFiles(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.file", "EAFile")

xmlDoc.loadXML (EAPProject.GetElementProblems(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.defect", "EADefect")

xmlDoc.loadXML (EAPProject.GetElementResources(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.resource", "EAResource")

xmlDoc.loadXML (EAPProject.GetElementRisks(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.risk", "EARisk")

xmlDoc.loadXML (EAPProject.GetElementTests(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.test", "EATest")

  AddToDebugList "End " + xmlNode.nodeName
AddToDebugList "_____ "
  AddToDebugList " "

Exit Sub

errElement:
  AddToDebugList Error$

End Sub

```

15.2.10 Enumerating Links

Once you have an element, you can enumerate all the links for it. To do this, first locate your element using the EnumElements interface, then pass the element GUID to the EnumLinks interface.

The interface is defined as *EnumLinks(const VARIANT FAR& ElementID)* - and returns an XML list of all links associated with an element. You can pass the Link GUID to the GetLink interface to retrieve specific information about a link.

Example XML from the EnumLinks interface

```

<Document xmlns:UML="omg.org/UML1.3">
  <Link>
    <Type>NoteLink</Type>
    <LinkID>EAID_0D03CF69_61BF_4041_9CD8_540901F171F9</LinkID>
    <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
    <OtherType>Actor</OtherType>
    <OtherIsTarget>0</OtherIsTarget>
  </Link>
  <Link>
    <Type>UseCase</Type>
    <LinkID>EAID_23D317E7_B50A_45a5_AEE8_081A26CEA18D</LinkID>
    <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
    <OtherType>Actor</OtherType>
    <OtherIsTarget>-1</OtherIsTarget>
  </Link>
</Document>

```

```

</Link>
<Link>
  <Type>UseCase</Type>
  <LinkID>EAID_4E742B84_D86E_4e37_B91C_8D3E25299646</LinkID>
  <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
  <OtherType>Actor</OtherType>
  <OtherIsTarget>-1</OtherIsTarget>
</Link>
<Link>
  <Type>UseCase</Type>
  <LinkID>EAID_77FA191A_C775_40b0_96FC_A741D1083D88</LinkID>
  <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
  <OtherType>Actor</OtherType>
  <OtherIsTarget>-1</OtherIsTarget>
</Link>
</Document>

```

Example Visual Basic code used to enumerate links

```

Private Sub GetLinks(ObjectGUID As String, Offset As String)

    ' enumerate through the list of links for an object
    ' for each link, call the GetLink function to retrieve actual details

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xml As New MSXML2.DOMDocument
    Dim str As String

    xml.LoadXML EAProject.EnumLinks(ObjectGUID)

    ' Debug.Print xml.xml

    Set xmlNode = xml.SelectSingleNode("Document/Link")

    Do While (Not xmlNode Is Nothing)
        AddToTreeList Offset + Indent + "link: " + xmlNode.SelectSingleNode("Type").Text
        GetLink xmlNode.SelectSingleNode("LinkID").Text
        Set xmlNode = xmlNode.NextSibling
    Loop

End Sub

```

15.2.11 Get Link

You may retrieve information about a link in Enterprise Architect using the GetLink interface. This takes a Link GUID obtained from the EnumLinks method and returns an XML document containing full details about a UML link.

The interface is defined as *GetLink(const VARIANT FAR& LinkGUID)* - and returns an XML document of a link.

XML example from a call to GetLink

```

<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Link>
    <UML:Association xmi.id="EAID_23D317E7_B50A_45a5_AEE8_081A26CEA18D" visibility="public"
    isRoot="false" isLeaf="false" isAbstract="false">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="style" value="1" />
        <UML:TaggedValue tag="ea_type" value="UseCase" />
        <UML:TaggedValue tag="linemode" value="1" />
        <UML:TaggedValue tag="segno" value="0" />
      </UML:ModelElement.taggedValue>
    </UML:Association>
  </Link>
</Document>

```

```

    <UML:TaggedValue tag="documentation" value="A customer uses the login use case to access the book
store" />
  </UML:ModelElement.taggedValue>
  <UML:Association.connection>
    <UML:AssociationEnd visibility="public" aggregation="none" isOrdered="false" targetScope="instance"
isNavigable="true" Type="EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="containment" value="Unspecified" />
      </UML:ModelElement.taggedValue>
    </UML:AssociationEnd>
    <UML:AssociationEnd visibility="public" aggregation="none" isOrdered="false" targetScope="instance"
isNavigable="true" Type="EAID_B00C2EA6_3DD4_4e7b_886E_A8B7D2B17AEA">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="containment" value="Unspecified" />
      </UML:ModelElement.taggedValue>
    </UML:AssociationEnd>
  </UML:Association.connection>
</UML:Association>
</Link>
</Document>

```

A Visual Basic Example of retrieving Link information using a Link GUID

```

Private Sub GetLink(LinkGUID As String)
On Error GoTo errLink

'getasinglelinkdetails

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xmlElement As New MSXML2.DOMDocument
    Dim xmlDoc As New MSXML2.DOMDocument
    Dim xmlTagged As MSXML2.IXMLDOMNode
    Dim n As Integer

    Dim str As String
    str = EAProject.GetLink(LinkGUID)
    Debug.Print str

    xmlElement.LoadXML EAProject.GetLink(LinkGUID)

    Set xmlNode = xmlElement.SelectSingleNode("Document/Link")

    If (xmlNode Is Nothing) Then
        Exit Sub
    End If

    Debug.Print xmlElement.xml

    'go to first element - will be the actual element (eg. UML:Actor)
    Set xmlNode = xmlNode.FirstChild()

    AddToDebugList " "
    AddToDebugList xmlNode.nodeName

    'getattributesfirst
    For n = 0 To xmlNode.Attributes.Length - 1
        AddToDebugList xmlNode.Attributes.Item(n).nodeName + " = " + xmlNode.Attributes.Item(n).Text
    Next n

    'then get tagged values
    Set xmlTagged = xmlElement.SelectSingleNode("UML:ModelElement.taggedValue/UML:TaggedValue")
    Do While (Not xmlTagged Is Nothing)
        AddToDebugList xmlTagged.Attributes(0).Text + " = " + xmlTagged.Attributes(1).Text
        Set xmlTagged = xmlTagged.NextSibling
    Loop

    AddToDebugList " "

    Exit Sub
errLink:

```

```
enLink:
  AddToDebugList Error$
End Sub
```

15.2.12 Get Diagram Image

You can retrieve a diagram image to either a file or onto the Windows clipboard for pasting into other applications.

In the first case you use the interface PutDiagramImageToFile(const VARIANT FAR& DiagramGUID, const VARIANT FAR& Filename, long Type) supplying the Diagram GUID, the output filename and the image type (metafile or bitmap = 0 or 1). If type = 0 then file type will be metafile, If type = 1 then it will use the file type from the name extension (ie. .bmp, .jpg, .gif, .png, .tga).

In the second case you call PutDiagramImageOnClipboard(const VARIANT FAR& DiagramGUID, long Type) passing the GUID and the image type.

A Visual Basic example of calling this code is given below:

```
' Code to select a diagram GUID
'
' Call the get image interface to put image on clipboard

EAProject.PutDiagramImageOnClipboard DiagramGUID, 0
'
'
'
```

Once complete, this Call will result In the requested diagram image being placed onto the Windows clipboard.

15.3 Add-Ins

Introduction

Add-ins let you add functionality to Enterprise Architect. The Enterprise Architect Add-in model builds on the features provided by the [Automation Interface](#) to allow you to extend the Enterprise Architect user interface.

Add-ins are ActiveX COM objects that expose public Dispatch methods. They have several advantages over stand-alone automation clients:

- Add-ins may define EA menus and sub-menus.
- Add-ins receive notifications about various EA user-interface events including menu clicks and file changes.
- Add-ins can (and should) be written as in-process (DLL) components. This provides lower call overhead and better integration into the EA environment.
- Because a current version of EA is already running there is no need to start a second copy of EA via the automation interface.
- Because the add-in receives object handles associated with the currently running copy of EA, more information is available about the current user's activity eg. which diagram objects are selected.
- The user is not required to do anything other than to install the add-in to make it usable - ie. Users of add-ins do not have to configure add-ins to run on their systems.

Creating and Using Add-Ins

This section of the help file covers the following information on Add-Ins:

- [Add-In Tasks](#)
- [Add-In Events](#)
- [Broadcast Events](#)
- [Custom Views](#)
- [MDG Add-Ins](#)

See Also

- [The Automation Interface](#)

15.3.1 Overview

The Enterprise Architect Add-in model builds on the features provided by the automation interface to allow Windows programmers to extend the Enterprise Architect user interface.

Add-Ins are ActiveX COM objects that expose public Dispatch methods that respond to EA.

Add-ins have several advantages over stand-alone automation clients:

- Add-ins may define EA menus and sub-menus.
- Add-ins receive notifications about various EA user-interface events including menu clicks and file changes.
- Add-ins can (and should) be written as in-process (DLL) components. This provides lower call overhead and better integration into the EA environment.
- Because a current version of EA is already running there is no need to start a second copy of EA via the automation interface.
- Because the add-in receives object handles associated with the currently running copy of EA, more information is available about the current user's activity eg. Which diagram objects are selected.
- The user is not required to do anything other than to install the add-in to make it usable - ie. Users of add-ins do not have to configure add-ins to run on their systems.

Because Enterprise Architect is constantly evolving in response to customer requests - the previous "hard-wired" add-in interface has been replaced by a more flexible one:

- The Addin interface no longer has its own version - rather it is identified by the version of EA it first appeared in. For example, the current version of the EA Add-in interface is version 2.1.
- There is no need to refer to a type-library when creating your add-in.
- Add-ins no longer need to implement methods that they never use.

Add-ins created using the previous mechanism will still run - though you will need to change your style of add-in to use new features.

Additional changes include:

- Add-ins now prompt users via context menus in the treeview and the diagram.
- Menu check and disable states can be controlled by the add-in.

15.3.2 Add-In Tasks

This section provides instructions to the developer on how to perform the tasks associated with creating, testing and deploying add-ins.

1. [Create an Add-In](#)
 - [Define Menu Items](#)
 - [Responding to Menu Events](#)
 - [Handling Add-In Events](#)
2. [Deploy your Add-In](#)

3. [Potential Pitfalls](#)

15.3.2.1 Creating Add-Ins

Before you start you will require an application development tool that is capable of creating ActiveX COM objects supporting the IDispatch interface. These include:

- Borland Delphi
- Microsoft Visual Basic
- Microsoft Visual Studio .Net

Create an Add-In

An EA Add-In can be created in three steps:

1. Use a development tool to create a ActiveX COM DLL project. Visual Basic users, for example, choose File>Create New Project*ActiveX DLL.
2. Connect to the interface using the syntax appropriate to the language as detailed in the [Connecting to the Interface](#) topic.
3. Create a COM class and implement each of the [general Add-In Events](#) applicable to your Add-In. You only need to define methods for events you wish to respond to.
4. Add a registry key identifying your Add-In to EA as described in [Deploying Add-Ins](#).

See Also

- [Defining Menu Items](#)
- [Examples of Automation Interfaces](#), this web page provides examples of code used to create Add-Ins for EA.

15.3.2.1.1 Defining Menu Items

Menu items are defined by responding to the GetMenuItems event.

The first time this event is called, MenuName will be an empty string, representing the top-level menu. For a simple Add-in with just a single menu option you can return a string, eg:

```
Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant
    EA_GetMenuItems = "&Joe's Add-in"
End Function
```

If you wish to define sub-menus, prefix a parent menu with a dash. Parent and sub-items are defined as follows:

```
Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant
    Select Case MenuName
        Case ""
            'Parent Menu Item
            EA_GetMenuItems = "-&Joe's Add-in"
        Case "-&Joe's Add-in"
            'Define Sub-Menu Items using the Array notation.
            'In this example, "Diagram" and "Treeview" will compose the "Joe's Add-in" sub-menu.
            EA_GetMenuItems = Array("Diagram", "&Treeview")
        Case Else
            MsgBox "InvalidMenu", vbCritical
    End Select
End Function
```

Similarly, further sub items may be defined:

```

Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant
    Select Case MenuName
    Case ""
        EA_GetMenuItems = "-Joe's Add-in"
    Case "-Joe's Add-in"
        EA_GetMenuItems = Array("-&Diagram", "&TreeView")
    Case "-&Diagram"
        EA_GetMenuItems = "&Properties"
    Case Else
        MsgBox "Invalid Menu", vbCritical
    End Select
End Function

```

If you wish to enable or disable menu items by default, this method may be used to show particular items to the user:

```

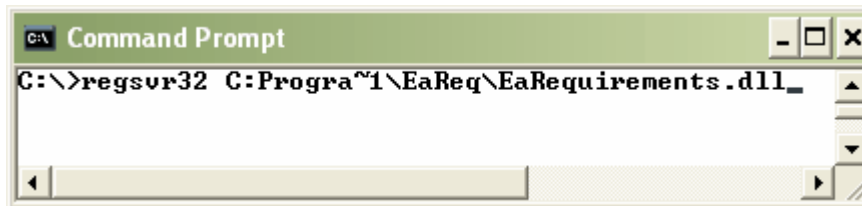
Sub EA_GetMenuState(Repository As EA.Repository, Location As String, MenuName As String, ItemName As String,
IsEnabled As Boolean, IsChecked As Boolean)
    Select Case Location
    Case "TreeView"
        'Always enable
    Case "Diagram"
        'Always enable
    Case "MainMenu"
        Select Case ItemName
        Case "&Translate", "Save &Project"
            If GetIsProjectSelected() Then
                IsEnabled = False
            End If
        End Select
    End Select
    IsChecked = GetIsCurrentSelection()
End Sub

```

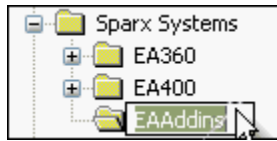
15.3.2.1.2 Deploying Add-Ins

Deploying Add-ins to users' sites requires these steps

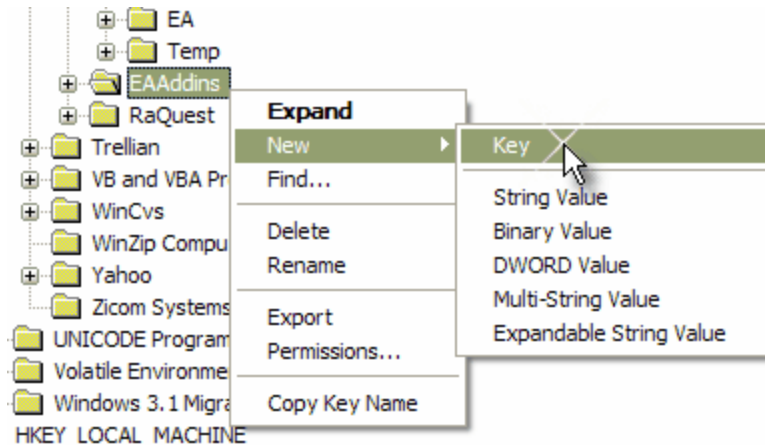
1. Add the Add-in DLL file to an appropriate directory on the user's computer i.e. C:\Program Files\ [new dir].
2. Register the DLL by using the "RegSvr32" command from the command prompt as shown in the example below.



3. Place a new entry into the registry so that EA recognizes the presence of your Add-in by using the registry editor (run regedit).
4. Add a new key value EAAddIns under the following location:
HKEY_CURRENT_USER\Software\Sparx Systems

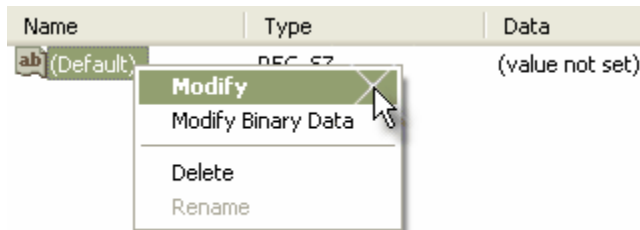


- Then add a new key under this key with the project name.

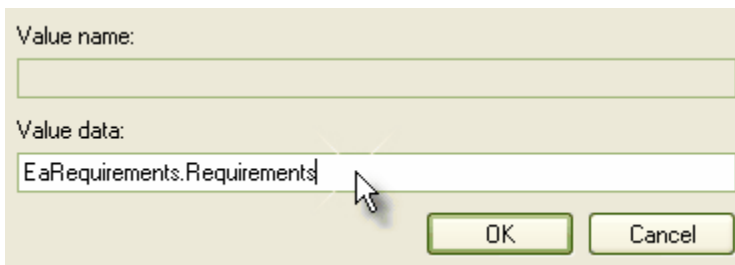


Note: [ProjectName] is not necessarily the name of your DLL, but the name of the Project. In VB, this is the value for the property "Name" corresponding to the project file.

- Then specify the default value by modifying the default value of the key.



- Enter the value of the key by entering the [project name].[class name]. i.e. EaRequirements. Requirements where EaRequirements is the project name as shown in the example below.



15.3.2.1.3 Tricks and Traps

Visual Basic 5/6 Users Note

Visual Basic 5/6 users should note that the version number of the EA interface is stored in the VBP project file in a form similar to the following:

Reference="G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\Program Files\Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02

If you experience problems moving from one version of EA to another, open the VBP file in a text editor and remove this line. Then open the project in Visual Basic and use Project-References to create a new reference to the EA Object model.

Holding State Information

It is possible for an Add-in to hold state information, meaning that data can be stored in member variables in response to one event and retrieved in another. There are some dangers in doing this:

- EA Automation Objects do not update themselves in response to user activity, to activity on other workstations, or even to the actions of other objects in the same automation client. Retaining handles to such objects between calls may result in the second event querying objects that have no relationship with the current state of EA.
- When the user asks EA to close, all Add-ins are asked to shut down. If there are any external automation clients EA will need to stay active, in which case all the add-ins will be reloaded - losing all the data.
- EA acting as an automation client will not close if an Add-in still holds a reference to it (releasing all references in the Disconnect() event avoids this problem).

It is recommended that unless there is a specific reason for doing so, the add-in should use the repository parameter and its method and properties to provide the necessary data.

EA Not Closing

.Net Specific Issues

Automation checks the use of objects and won't allow any of them to be destroyed until they are no longer being used.

As noted in the automation interface section, if your automation controller was written using the .NET framework, EA will not close even after you release all your references to it. To force the release of the COM pointers, call the memory management functions as shown below:

```
GC.Collect();
GC.WaitForPendingFinalizers();
```

Additionally, because automation clients hook into EA which create Add-ins which in turn hook back into EA, it is possible to get into a deadlock situation where EA and the Add-ins won't let go of one another and keep each other active. An Add-in may retain hooks into EA because:

- It keeps a private reference to an EA object (see Holding state information above).
- It has been created by .NET and the GC mechanism hasn't got around to releasing it.

There are two actions required to avoid deadlock situations:

- Automation controllers must call Repository.CloseAddins() at some point (presumably at the end of processing).
- Add-Ins must release all references to EA in the Disconnect() event. See the [Add-In Methods](#) page for details.

It is possible that your Automation client controls a running instance of EA where the Add-Ins have not complied to the rule above. In this case you may wish to call Repository.Exit() to terminate EA.

Miscellaneous

In developing add-Ins using the .Net framework you will be required to select COM Interoperability in the project's properties in order for it to be recognized as an add-in.

Some development environments do not automatically register com DLLs on creation. You may need to do that manually before EA recognizes it.

You can use your private Add-In key (as required for Add-in deployment) to store configuration information pertinent to your Add-in.

15.3.3 Add-In Search

Enterprise Architect allows add-in's to integrate with the [Search View](#). Searches can be defined that execute a method within your add-in and display your results in an integrated way. For more information, see [Creating Search Definitions](#).

The method that runs the search needs to be structured the following way:
variant <method name> (Rep as Repository, SearchText as String, XMLResults as String)

Parameters:

| | |
|-------------|--|
| Rep: | The currently open repository. |
| SearchText: | This is an optional field that the user may fill in through the Search View . |
| XMLResults: | At completion of the method, XMLResults should contain the results for the search. The results should be an XML String that conforms to the Search Data Format . |

Return:

The method must return a value for the results to be displayed.

Advanced Usage

In addition to the displayed results, two additional hidden fields can be passed into the xml that will provide special functionality.

CLASSTYPE

Returning a field of CLASSTYPE, containing the Object_Type value from the t_object table, will display the appropriate icon in the column you place the field.

CLASSGUID

Returning a field of CLASSGUID, containing an ea_guid value, will allow the Search View to track the object in the Project Browser and open the Properties window for the element by double-clicking in the Search View.

15.3.3.1 XML Format (Search Data)

The XML below provides the format for the SearchData parameter of the RunModel method. See [Repository](#) for more information.

```
<ReportViewData>
  <!--
    //Use this section to declare all possible fields - columns that will appear in EA's search window - that will be used
    below in <Rows/>.
    // The order of the columns of information to be appended here needs to match the order that the search run in EA
    would normally display.
    // Furthermore, if your append results onto a custom SQL Search, then order used in your Custom SQL must match
    the order used below.
  -->

  <Fields>
    <Fieldname="" />
    <Fieldname="" />
    <Fieldname="" />
    <Fieldname="" />
  </Fields>

  <Rows>
    <Row>
      <Fieldname="" value="" />
      <Fieldname="" value="" />
      <Fieldname="" value="" />
      <Fieldname="" value="" />
    </Row>
    <Row>
      <Fieldname="" value="" />
```

```

        <Fieldname="" value="" />
        <Fieldname="" value="" />
        <Fieldname="" value="" />
    </Row>
    <Row>
        <Fieldname="" value="" />
        <Fieldname="" value="" />
        <Fieldname="" value="" />
        <Fieldname="" value="" />
    </Row>
</Rows>
</ReportViewData>

```

15.3.4 Add-In Events

All EA Add-Ins may choose to respond to general Add-In events.

See Also

- [EA_Connect](#)
- [EA_Disconnect](#)
- [EA_GetMenuItems](#)
- [EA_MenuClick](#)
- [EA_GetMenuState](#)
- [EA_ShowHelp](#)
- [EA_OnOutputItemClicked](#)
- [EA_OnOutputItemDoubleClicked](#)

15.3.4.1 EA Connect

EA_Connect events allow Add-Ins to identify their type and to respond to EA start up.

Syntax

Function EA_Connect(*Repository As EA.Repository*) As String

The **EA_Connect** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

Return Value

String identifying a specialized type of Add-In.:

| Type | Details |
|-------|--|
| "MDG" | MDG Add-Ins receive MDG Events and extra menu options. |
| "" | None-specialized Add-in |

Details

This event occurs when EA first loads your Add-In. EA itself is loading at this time so that while a Repository object is supplied, there is limited information that you can extract from it.

The chief uses for EA_Connect are in initializing global Add-In data and for identifying the Add-In as an MDG Add-In.

See Also

- [EA Disconnect](#)
- [MDG Add-Ins](#)

15.3.4.2 EA Disconnect

The EA_Disconnect event allows the Add-In to respond to user requests to disconnect the model branch from an external project.

Syntax

Sub EA_Disconnect()

Return Value

None

Details

This function will be called when the EA closes. If you have stored references to EA objects (not particularly recommended anyway), you must release them here.

In addition, .NET users must call memory management functions as shown below:

```
GC.Collect();
GC.WaitForPendingFinalizers();
```

See Also

- [EA Connect](#)

15.3.4.3 EA GetMenuItems

The EA_GetMenuItems allows the Add-In to provide the EA user interface with additional "add-In" menu items in various context and main menus. When a user selects an Add-In menu option, an event is raised and passed back to the Add-In that originally defined that menu item.

Syntax

Function EA_GetMenuItems(*Repository As EA.Repository, MenuLocation As String, MenuName As String*) As Variant

The EA_GetMenuItems function syntax contains the following elements

| Parameter | Type | Direction | Description |
|---------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>MenuLocation</i> | String | | String representing the part of the user interface that brought up the menu. May be "TreeView", "MainMenu" or "Diagram". |
| <i>MenuName</i> | String | | The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string. |

Return Value

One of the following types:

- A string indicating the label for a single menu item.
- An array of strings indicating a multiple menu items.
- Empty (Visual Basic/VB.NET) or null (C#) to indicate that no menu should be displayed.

In the case of the top-level menu it should be a single string or an array containing only one item, or Empty/null.

Details

This event is raised just before EA needs to show particular menu items to the user, and its use is described in the [Defining Menu Items](#) topic.

See Also

- [EA_MenuClick](#)
- [EA_GetMenuState](#)

15.3.4.4 EA_GetMenuState

The EA_GetMenuState allows the Add-In to set a particular menu option to either enabled or disabled. This is useful when dealing with locked packages and other situations where it is convenient to show a menu item - but not enable it for use. .

Syntax

Sub EA_GetMenuState(Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String, IsEnabled as Boolean, IsChecked as Boolean)

The EA_GetMenuItems function syntax contains the following elements

| Parameter | Type | Direction | Description |
|---------------------|---------------|------------------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>MenuLocation</i> | String | | String representing the part of the user interface that brought up the menu. May be "Treeview", "MainMenu" or "Diagram". |
| <i>MenuName</i> | String | | The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string. |
| <i>ItemName</i> | String | | The name of the option actually clicked, eg. "Create &New Invoice". |
| <i>IsEnabled</i> | Boolean | | Boolean. Set to False to disable this particular menu item. |
| <i>IsChecked</i> | Boolean | | Boolean. Set to True to check this particular menu item. |

Return Value

None.

Details

This event is raised just before EA needs to show particular menu items to the user, and its use is described in the [Defining Menu Items](#) topic.

See Also

- [EA_GetMenuItems](#)

15.3.4.5 EA MenuClick

EA_MenuClick events are received by an add-in in response to user selection of a menu item.

Syntax

Sub EA_MenuClick(Repository As EA.Repository, MenuLocation As String, MenuName As String, ItemName As String)

The **EA_GetMenuClick** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>MenuName</i> | String | | The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string. |
| <i>ItemName</i> | String | | The name of the option actually clicked, eg. "Create &New Invoice". |

Return Value

None.

Details

The event is raised when the user clicks on a particular menu item. When a user clicks on one of your non-parent menu items, your Add-in will receive a MenuClick event, defined as follows:

```
Sub EA_MenuClick(Repository As EA.Repository, ByVal MenuName As String, ByVal ItemName As String)
```

The code below illustrates an example of use:

```
If MenuName = "-&Diagram" And ItemName = "&Properties" then
  MsgBox Repository.GetCurrentDiagram.Name, vbInformation
Else
  MsgBox "Not Implemented", vbCritical
End If
```

Notice that your code can directly access EA data and UI elements using [Repository](#) methods.

See Also

- [EA_GetMenuItems](#)

15.3.4.6 EA OnOutputItemClicked

EA_OnOutputItemClicked events informs Add-Ins that the user has used the mouse to click on a list entry in the system tab or one of the user defined output tabs.

Syntax

EA_OnOutputItemClicked(*Repository As EA.Repository, TabName As String, LineText As String, LineNo As Long*)

The **EA_OnOutputItemClicked** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>TabName</i> | String | IN | The name of the tab that the click occurred in. Usually this will have been created through Repository.AddTab(). |
| <i>LineText</i> | String | IN | The text that had been supplied as the String parameter in the original call to Repository.WriteLine(). |
| <i>ID</i> | Long | IN | The ID value specified in the original call to Repository.WriteLine(). |

Return Value

None.

Details

Usually an Add-In will respond to this event in order to capture activity on an output tab they had previously created through a call to Repository.AddTab().

Note that every loaded Add-In receives this event for every click on an output tab in EA - irrespective of whether the Add-In created that tab. Add-Ins should therefore check the TabName parameter supplied by this event to ensure that they are not responding to other Add-Ins' events.

See Also

- [EA_OnOutputItemDoubleClicked](#)

15.3.4.7 EA_OnOutputItemDoubleClicked

EA_OnOutputItemDoubleClicked events informs Add-Ins that the user has used the mouse to double-click on a list entry in one of the user defined output tabs.

Syntax

EA_OnOutputItemDoubleClicked(*Repository As EA.Repository, TabName As String, LineText As String, LineNo As Long*)

The **EA_OnOutputItemClicked** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

| | | | |
|-----------------|--------|----|--|
| <i>TabName</i> | String | IN | The name of the tab that the click occurred in. Usually this will have been created through Repository.AddTab(). |
| <i>LineText</i> | String | IN | The text that had been supplied as the String parameter in the original call to Repository.WriteLine(). |
| <i>ID</i> | Long | IN | The ID value specified in the original call to Repository.WriteLine(). |

Return Value

None.

Details

Usually an Add-In will respond to this event in order to capture activity on an output tab they had previously created through a call to Repository.AddTab().

Note that every loaded Add-In receives this event for every double-click on an output tab in EA - irrespective of whether the Add-In created that tab. Add-Ins should therefore check the TabName parameter supplied by this event to ensure that they are not responding to other Add-Ins' events.

See Also

- [EA_OnOutputItemClicked](#)

15.3.4.8 EA_ShowHelp

The EA_ShowHelp allows the Add-In to show a help topic for a particular menu option. When the user has an Add-In menu option selected, pressing F1 can be delegated to the required Help topic by the Add-In and a suitable help message shown. .

Syntax

Sub EA_ShowHelp(Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String)

The EA_GetMenuItems function syntax contains the following elements

| Parameter | Type | Direction | Description |
|---------------------|---------------|------------------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>MenuLocation</i> | String | | String representing the part of the user interface that brought up the menu. May be "Treeview", "MainMenu" or "Diagram". |
| <i>MenuName</i> | String | | The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string. |
| <i>ItemName</i> | String | | The name of the option actually clicked, eg. "Create &New Invoice". |

Return Value

None.

Details

This event is raised when the user presses F1 on a menu option that is not a parent menu.

See Also

- [EA_GetMenuItems](#)

15.3.5 Broadcast Events

General Broadcast are sent to all loaded Add-Ins. For an Add-In to receive the event, they must first implement the required automation event interface. If EA detects that the Add-In has the required interface, the event is dispatched on to the Add-In. MDG Events add quite a number of additional events - but the Add-In must first have registered as an MDG style Add-In, rather than as a generic Add-In.

See Also

- [EA_FileOpen](#)
- [EA_FileClose](#)
- [EA_OnPreDeleteElement](#)
- [EA_OnPreDeleteConnector](#)
- [EA_OnPreDeleteDiagram](#)
- [EA_OnPreDeletePackage](#)
- [EA_OnPreNewElement](#)
- [EA_OnPreNewConnector](#)
- [EA_OnPreNewAttribute](#)
- [EA_OnPreNewMethod](#)
- [EA_OnPreNewPackage](#)
- [EA_OnPostNewElement](#)
- [EA_OnPostNewConnector](#)
- [EA_OnPostNewAttribute](#)
- [EA_OnPostNewMethod](#)
- [EA_OnPostNewPackage](#)
- [EA_OnPreDeleteTechnology](#)
- [EA_OnDeleteTechnology](#)
- [EA_OnImportTechnology](#)
- [EA_OnContextItemChanged](#)
- [EA_OnContextItemDoubleClicked](#)
- [EA_OnNotifyContextItemModified](#)
- [EA_OnPostTransform](#)

15.3.5.1 EA_FileOpen

The EA_FileOpen allows the Add-In to respond to a File Open event. When EA opens a new Model file, this event is raised and passed to all Add-Ins implementing this method.

Syntax

Sub EA_FileOpen(*Repository* As EA.Repository)

The EA_FileOpen function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

Return Value

None

Details

This event occurs when the model being viewed by the EA user changes, for whatever reason (through user interaction or Add-In activity).

See Also

- [EA_FileClose](#)

15.3.5.2 EA FileClose

The EA_FileClose allows the Add-In to respond to a File Close event. When EA closes an opened Model file, this event is raised and passed to all Add-Ins implementing this method.

Syntax

Sub EA_FileClose(*Repository As EA.Repository*)

The EA_FileClose function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the EA model about to be closed. Poll its members to retrieve model data and user interface status information. |

Return Value

None

Details

This event occurs when the model currently opened within EA is about to be closed (when another model is about to be opened or when EA is about to shutdown).

See Also

- [EA_FileOpen](#)

15.3.5.3 EA OnPreDeleteElement

EA_OnPreDeleteElement notifies Add-ins that an element is to be deleted from the model. It allows add-ins to permit or deny the deletion of the element

Syntax

Function EA_OnPreDeleteElement(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The EA_OnPreDeleteElement function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-----------|------|-----------|-------------|
|-----------|------|-----------|-------------|

| | | | |
|-------------------|--------------------|----|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the element to be created : <ul style="list-style-type: none"> • ElementID : A long value corresponding to Element.ElementID |

Return Value

Return True to allow the deletion of the element from the model. Return False to disallow the deletion of the element.

Details

This event occurs when a user deletes an element from the Project Browser or on a diagram. The notification is provided immediately before the element is deleted, so that the add-in may disallow the deletion of the element.

See Also

- [EA.EventProperties](#)

15.3.5.4 EA_OnPreDeleteConnector

EA_OnPreDeleteConnector notifies Add-ins that a connector is to be deleted from the model. It allows add-ins to permit or deny the deletion of the connector

Syntax

Function EA_OnPreDeleteConnector(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreDeleteConnector** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|------------------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the connector to be created: <ul style="list-style-type: none"> • ConnectorID : A long value corresponding to Connector.ConnectorID |

Return Value

Return True to allow the deletion of the connector from the model. Return False to disallow the deletion of the connector.

Details

This event occurs when a user attempts to permanently delete a connector on a diagram. The notification is provided immediately before the connector is deleted, so that the add-in may disallow the deletion of the connector.

See Also

- [EA.EventProperties](#)

15.3.5.5 EA_OnPreDeleteDiagram

EA_OnPreDeleteDiagram notifies Add-ins that a diagram is to be deleted from the model. It allows add-ins to permit or deny the deletion of the diagram

Syntax

Function EA_OnPreDeleteDiagram(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreDeleteDiagram** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the connector to be created: <ul style="list-style-type: none"> • DiagramID : A long value corresponding to Diagram.DiagramID |

Return Value

Return True to allow the deletion of the diagram from the model. Return False to disallow the deletion of the diagram.

Details

This event occurs when a user attempts to permanently delete a diagram from the Project Browser. The notification is provided immediately before the diagram is deleted, so that the add-in may disallow the deletion of the diagram.

See Also

- [EA.EventProperties](#)

15.3.5.6 EA OnPreDeletePackage

EA_OnPreDeletePackage notifies Add-ins that a package is to be deleted from the model. It allows add-ins to permit or deny the deletion of the package

Syntax

Function EA_OnPreDeletePackage(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreDeletePackage** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|------------------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the connector to be created: <ul style="list-style-type: none"> PackageID : A long value corresponding to Package.PackageID |

Return Value

Return True to allow the deletion of the package from the model. Return False to disallow the deletion of the package.

Details

This event occurs when a user attempts to permanently delete a package from the Project Browser. The notification is provided immediately before the package is deleted, so that the add-in may disallow the deletion of the package.

See Also

- [EA.EventProperties](#)

15.3.5.7 EA OnPreDeleteTechnology

EA_OnPreDeleteTechnology notifies Add-ins that an MDG Technology resource is about to be deleted from the model.

Syntax

Function EA_OnPreDeleteTechnology(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreDeleteTechnology** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|------------------|-------------|------------------|--------------------|
|------------------|-------------|------------------|--------------------|

| | | | |
|-------------------|--------------------|----|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the MDG Technology to be deleted : <ul style="list-style-type: none"> • TechnologyID: A string value corresponding to the MDG Technology ID |

Return Value

Return True to allow the deletion of the MDG Technology resource from the model. Return False to disallow the deletion of the MDG Technology resource.

Details

This event occurs when a user deletes an MDG Technology resource from the model. The notification is provided immediately after the user confirms their request to delete the MDG Technology, so that the add-in may disallow the deletion of the MDG Technology.

See Also

- [EA.EventProperties](#)
- [EA_OnImportTechnology](#)
- [EA_OnDeleteTechnology](#)

15.3.5.8 EA_OnPreNewElement

EA_OnPreNewElement notifies Add-ins that a new element is about to be created on a diagram. It allows add-ins to permit or deny the creation of the new element.

Syntax

Function EA_OnPreNewElement(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreNewElement** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|------------------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the element to be created : <ul style="list-style-type: none"> • Type : A string value corresponding to Element.Type • Stereotype : A string value corresponding to Element.Stereotype • ParentID : A long value corresponding to Element.ParentID • DiagramID : A long value corresponding to the ID of the diagram to which the element is being added |

Return Value

Return True to allow the addition of the new element to the model. Return False to disallow the addition of the new element.

Details

This event occurs when a user drags a new element from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately before the element is created, so that the add-in may disallow the addition of the element.

See Also

- [EA.EventProperties](#)
- [EA_OnPostNewElement](#)

15.3.5.9 EA_OnPreNewConnector

EA_OnPreNewConnector notifies Add-ins that a new connector is about to be created on a diagram. It allows add-ins to permit or deny the creation of new connector.

Syntax

Function EA_OnPreNewConnector(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreNewConnector** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

| | | | |
|-------------|--------------------|----|--|
| <i>Info</i> | EA.EventProperties | IN | <p>Contains the following EventProperty Objects for the connector to be created :</p> <ul style="list-style-type: none"> • Type : A string value corresponding to Connector.Type • Subtype : A string value corresponding to Connector.Subtype • Stereotype : A string value corresponding to Connector.Stereotype • ClientID : A long value corresponding to Connector.ClientID • SupplierID : A long value corresponding to Connector.SupplierID <p>DiagramID : A long value corresponding to Connector.DiagramID</p> |
|-------------|--------------------|----|--|

Return Value

Return True to allow the addition of the new connector to the model. Return False to disallow the addition of the new connector.

Details

This event occurs when a user drags a new connector from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately before the connector is created, so that the add-in may disallow the addition of the connector.

See Also

- [EA.EventProperties](#)
- [EA_OnPostNewConnector](#)

15.3.5.10 EA_OnPreNewAttribute

EA_OnPreNewAttribute notifies Add-ins that a new attribute is about to be created on an element. It allows add-ins to permit or deny the creation of the new attribute.

Syntax

Function EA_OnPreNewAttribute(*Repository* As EA.Repository, *Info* As EA.EventProperties) As Boolean

The **EA_OnPreNewAttribute** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|------------------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

| | | | |
|-------------|------------------------|----|--|
| <i>Info</i> | EA. EventProperties | IN | <p>Contains the following EventProperty Objects for the attribute to be created :</p> <ul style="list-style-type: none"> • Type : A string value corresponding to Attribute.Type • Stereotype : A string value corresponding to Attribute.Stereotype • ParentID : A long value corresponding to Attribute.ParentID • ClassifierID : A long value corresponding to Attribute.ClassifierID |
|-------------|------------------------|----|--|

Return Value

Return True to allow the addition of the new attribute to the model. Return False to disallow the addition of the new attribute.

Details

This event occurs when a user creates a new attribute on an element by either drag-dropping from the Project Browser, using the Attributes properties dialog, or the the in-place editor on the diagram. The notification is provided immediately before the attribute is created, so that the add-in may disallow the addition of the attribute.

See Also

- [EA.EventProperties](#)
- [EA_OnPostNewAttribute](#)

15.3.5.11 EA_OnPreNewMethod

EA_OnPreNewMethod notifies Add-ins that a new method is about to be created on an element. It allows add-ins to permit or deny the creation of the new method.

Syntax

Function EA_OnPreNewMethod(*Repository* As EA.Repository, *Info* As EA.EventProperties) As Boolean

The **EA_OnPreNewMethod** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|------------------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA. EventProperties | IN | <p>Contains the following EventProperty Objects for the method to be created :</p> <ul style="list-style-type: none"> • ReturnType : A string value corresponding to Method.ReturnType • Stereotype : A string value corresponding to Method.Stereotype • ParentID : A long value corresponding to Method.ParentID • ClassifierID : A long value corresponding to Method.ClassifierID |

Return Value

Return True to allow the addition of the new method to the model. Return False to disallow the addition of the new method.

Details

This event occurs when a user creates a new method on an element by either drag-dropping from the Project Browser, using the Method properties dialog, or the the in-place editor on the diagram. The notification is provided immediately before the method is created, so that the add-in may disallow the addition of the method.

See Also

- [EA.EventProperties](#)
- [EA_OnPostNewMethod](#)

15.3.5.12 EA_OnPreNewPackage

EA_OnPreNewPackage notifies Add-ins that a new package is about to be created in the model. It allows add-ins to permit or deny the creation of the new package.

Syntax

Function EA_OnPreNewPackage(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreNewPackage** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the package to be created : <ul style="list-style-type: none"> • Stereotype : A string value corresponding to Package.Stereotype • ParentID : A long value corresponding to Package.ParentID • DiagramID : A long value corresponding to the ID of the diagram to which the package is being added |

Return Value

Return True to allow the addition of the new package to the model. Return False to disallow the addition of the new package.

Details

This event occurs when a user drags a new package from the UML toolbox or Resource Tree, onto a diagram, or by invoking the New Package command from the Project Browser. The notification is provided immediately before the package is created, so that the add-in may disallow the addition of the package.

See Also

- [EA.EventProperties](#)
- [EA_OnPostNewPackage](#)

15.3.5.13 EA_OnPostNewElement

EA_OnPostNewElement notifies Add-ins that a new element has been created on a diagram. It allows add-ins to modify the element upon creation.

Syntax

Function EA_OnPostNewElement(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPostNewElement** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|------------------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the new element: <ul style="list-style-type: none"> • ElementID : A long value corresponding to Element.ElementID |

Return Value

Return True if the element has been updated during this notification. Return False otherwise.

Details

This event occurs after a user has dragged a new element from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately after the element is added to the model. Set Repository.SuppressEADialogs = true to suppress EA from showing its default dialogs.

See Also

- [EA.EventProperties](#)
- [EA.Repository](#)
- [EA_OnPreNewElement](#)

15.3.5.14 EA_OnPostNewConnector

EA_OnPostNewConnector notifies Add-ins that a new connector has been created on a diagram. It allows add-ins to modify the connector upon creation.

Syntax

Function EA_OnPostNewConnector(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPostNewConnector** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the new connector: <ul style="list-style-type: none"> ConnectorID : A long value corresponding to Connector.ConnectorID |

Return Value

Return True if the connector has been updated during this notification. Return False otherwise.

Details

This event occurs after a user has dragged a new connector from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately after the connector is added to the model. Set Repository.SuppressEADialogs = true to suppress EA from showing its default dialogs.

See Also

- [EA.EventProperties](#)
- [EA.Repository](#)
- [EA_OnPreNewConnector](#)

15.3.5.15 EA_OnPostNewAttribute

EA_OnPostNewAttribute notifies Add-ins that a new attribute has been created on a diagram. It allows add-ins to modify the attribute upon creation.

Syntax

Function EA_OnPostNewAttribute(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPostNewAttribute** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

| | | | |
|-------------|--------------------|----|---|
| <i>Info</i> | EA.EventProperties | IN | <p>Contains the following EventProperty Objects for the new attribute :</p> <ul style="list-style-type: none"> • AttributeID : A long value corresponding to Attribute.AttributeID |
|-------------|--------------------|----|---|

Return Value

Return True if the attribute has been updated during this notification. Return False otherwise.

Details

This event occurs when a user creates a new attribute on an element by either drag-dropping from the Project Browser, using the Attributes properties dialog, or the the in-place editor on the diagram. The notification is provided immediately after the attribute is created. Set Repository.SuppressEADialogs = true to suppress EA from showing its default dialogs.

See Also

- [EA.EventProperties](#)
- [EA.Repository](#)
- [EA_OnPreNewAttribute](#)

15.3.5.16 EA_OnPostNewMethod

EA_OnPostNewMethod notifies Add-ins that a new method has been created on a diagram. It allows add-ins to modify the method upon creation.

Syntax

Function EA_OnPostNewMethod(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPostNewMethod** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | <p>Contains the following EventProperty Objects for the new method :</p> <ul style="list-style-type: none"> • MethodID : A long value corresponding to Method.MethodID |

Return Value

Return True if the method has been updated during this notification. Return False otherwise.

Details

This event occurs when a user creates a new method on an element by either drag-dropping from the Project Browser, using the Methods properties dialog, or the the in-place editor on the diagram. The notification is provided immediately after the method is created. Set `Repository.SuppressEADialogs = true` to suppress EA from showing its default dialogs.

See Also

- [EA.EventProperties](#)
- [EA.Repository](#)
- [EA_OnPreNewMethod](#)

15.3.5.17 EA_OnPostNewPackage

`EA_OnPostNewPackage` notifies Add-ins that a new package has been created on a diagram. It allows add-ins to modify the package upon creation.

Syntax

Function `EA_OnPostNewPackage(Repository As EA.Repository, Info As EA.EventProperties) As Boolean`

The `EA_OnPostNewPackage` function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the new package: <ul style="list-style-type: none"> • PackageID : A long value corresponding to Package.PackageID |

Return Value

Return True if the package has been updated during this notification. Return False otherwise.

Details

This event occurs when a user drags a new package from the UML toolbox or Resource Tree, onto a diagram, or by invoking the New Package command from the Project Browser. Set `Repository.SuppressEADialogs = true` to suppress EA from showing its default dialogs.

See Also

- [EA.EventProperties](#)
- [EA.Repository](#)
- [EA_OnPreNewPackage](#)

15.3.5.18 EA OnDeleteTechnology

EA_OnDeleteTechnology notifies Add-ins that an MDG Technology resource has been deleted from the model.

Syntax

Sub EA_OnDeleteTechnology(Repository As EA.Repository, Info As EA.EventProperties)

The **EA_OnDeleteTechnology** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|------------------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects : <ul style="list-style-type: none"> • TechnologyID : A string value corresponding to the MDG Technology ID |

Return Value

None

Details

This event occurs after a user has deleted an MDG Technology resource from the model. Add-ins that require an MDG Technology resource to be loaded, may catch this add-in to disable certain functionality

See Also

- [EA.EventProperties](#)
- [EA_OnImportTechnology](#)
- [EA_OnPreDeleteTechnology](#)

15.3.5.19 EA OnImportTechnology

EA_OnImportTechnology notifies Add-ins that the user has imported an MDG Technology resource into the model.

Syntax

Sub EA_OnImportTechnology(Repository As EA.Repository, Info As EA.EventProperties)

The **EA_OnImportTechnology** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|------------------|-------------|------------------|--------------------|
|------------------|-------------|------------------|--------------------|

| | | | |
|-------------------|--------------------|----|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects: <ul style="list-style-type: none"> TechnologyID : A string value corresponding to the MDG Technology ID |

Return Value

None

Details

This event occurs after a user has imported an MDG Technology resource into the model. Add-ins that require an MDG Technology resource to be loaded, may catch this add-in to enable certain functionality

See Also

- [EA.EventProperties](#)
- [EA_OnDeleteTechnology](#)

15.3.5.20 EA_OnContextItemChanged

EA_OnContextItemChanged notifies Add-ins that a different item is now in context.

Syntax

Sub EA_OnContextItemChanged(Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The **EA_OnContextItemChanged** function syntax contains the following elements

| Parameter | Type | Direction | Description | | |
|---|--|-----------|--|---|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. | | |
| <i>GUID</i> | String | IN | Contains the GUID of the new context item. This value corresponds to the following properties, depending on the value of the <i>ot</i> parameter: <table border="0" style="width: 100%; margin-top: 10px;"> <tr> <td style="vertical-align: top; width: 50%;"> ot (ObjectType) otElement otPackage otDiagram otAttribute otMethod otConnector otRepository </td> <td style="vertical-align: top; width: 50%;"> GUID value Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID NOT APPLICABLE, GUID is an empty string </td> </tr> </table> | ot (ObjectType) otElement otPackage otDiagram otAttribute otMethod otConnector otRepository | GUID value Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID NOT APPLICABLE, GUID is an empty string |
| ot (ObjectType) otElement otPackage otDiagram otAttribute otMethod otConnector otRepository | GUID value Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID NOT APPLICABLE, GUID is an empty string | | | | |

| | | | |
|-----------|-----------------------|----|---|
| <i>ot</i> | EA. ObjectT ype | IN | Specifies the type of the new context item. |
|-----------|-----------------------|----|---|

Return Value

None

Details

This event occurs after a user has selected an item anywhere in the EA GUI. Add-ins that require knowledge of the current item in context may subscribe to this broadcast function. If *ot* = *otRepository*, then this function behaves the same as [EA_FileOpen](#).

See Also

- [EA.EventProperties](#)
- [EA_FileOpen](#)
- [EA_OnContextItemDoubleClicked](#)
- [EA_OnNotifyContextItemModified](#)

15.3.5.21 EA_OnContextItemDoubleClicked

EA_OnContextItemDoubleClicked notifies Add-ins that the user has double-clicked the item currently in context.

Syntax

Function EA_OnContextItemDoubleClicked(Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The **EA_OnContextItemDoubleClicked** function syntax contains the following elements

| Parameter | Type | Direction | Description | | |
|---|---|-----------|--|---|---|
| <i>Repository</i> | EA. Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. | | |
| <i>GUID</i> | String | IN | Contains the GUID of the new context item. This value corresponds to the following properties, depending on the value of the <i>ot</i> parameter: <table style="margin-left: 40px; border: none;"> <tr> <td style="vertical-align: top;">ot (ObjectType) otElement otPackage otDiagram otAttribute otMethod otConnector</td> <td style="vertical-align: top; padding-left: 20px;">GUID value Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID</td> </tr> </table> | ot (ObjectType) otElement otPackage otDiagram otAttribute otMethod otConnector | GUID value Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID |
| ot (ObjectType) otElement otPackage otDiagram otAttribute otMethod otConnector | GUID value Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID | | | | |
| <i>ot</i> | EA. ObjectT ype | IN | Specifies the type of the new context item | | |

Return Value

Return True to notify EA that the double-click event has been handled by an Add-in. Return False to allow EA to continue processing the event.

Details

This event occurs when a user has double-clicked (or pressed the enter key) on the item in context either in a diagram or on the Project Browser tree. Add-ins that wish to handle event may subscribe to this broadcast function

See Also

- [EA.EventProperties](#)
- [EA_OnContextItemChanged](#)
- [EA_OnNotifyContextItemModified](#)

15.3.5.22 EA_OnNotifyContextItemModified

EA_OnNotifyContextItemModified notifies Add-ins that the current context item has been modified.

Syntax

Sub EA_OnNotifyContextItemModified(Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The **EA_OnNotifyContextItemModified** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>GUID</i> | String | IN | Contains the GUID of the new context item. This value corresponds to the following properties, depending on the value of the <i>ot</i> parameter: <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <div style="width: 45%;"> <p>ot (ObjectType)</p> <ul style="list-style-type: none"> otElement otPackage otDiagram otAttribute otMethod otConnector </div> <div style="width: 45%;"> <p>GUID value</p> <ul style="list-style-type: none"> Element.ElementGUID Package.PackageGUID Diagram.DiagramGUID Attribute.AttributeGUID Method.MethodGUID Connector.ConnectorGUID </div> </div> |
| <i>ot</i> | EA.ObjectType | IN | Specifies the type of the new context item |

Return Value

None.

Details

This event occurs when a user has modified the context item. Add-ins that require knowledge of when an item has been modified may subscribe to this broadcast function

See Also

- [EA.EventProperties](#)
- [EA_OnContextItemChanged](#)
- [EA_OnContextItemDoubleClicked](#)

15.3.5.23 EA OnPostTransform

EA_OnPostTransform notifies Add-ins that an MDG transformation has taken place with the output in the specified target package

Syntax

Function **EA_OnPostTransform(Repository As EA.Repository, Info As EA.EventProperties) As Boolean**

The **EA_OnPostTransform** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|--------------------|-----------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>Info</i> | EA.EventProperties | IN | Contains the following EventProperty Objects for the transform performed : <ul style="list-style-type: none"> • Transform : A string value corresponding to the name of the transform used • PackageID : A long value corresponding to Package.PackageID of the destination package |

Return Value

Reserved for future use.

Details

This event occurs when a user runs an MDG transform on one or more target packages. The notification is provided, for each transform/target package immediately after all transform processes have completed.

15.3.6 Custom Views

Enterprise Architect version 4.5 (and later) allows custom windows to be inserted as a tab into the Diagram Tabs that appear at the center of the EA frame.

Providing a custom view allows users to easily and quickly tab between a custom interface and diagrams and other views normally provided by EA.

Uses for this facility include:

- Reports and graphs showing summary data of the model.
- Alternate views of a diagram.
- Alternate views of the model.
- Views of external data related to model data.

- Documentation tools.

See Also

- [Creating a Custom View](#)

15.3.6.1 Creating a Custom View

A custom view needs to be designed as an ActiveX custom control and inserted through the automation interface.

ActiveX custom controls can be created using most well-known programming tools including Microsoft Visual Studio.NET. Refer to the documentation provided by the relevant vendor on how to create a custom control to produce an OCX file.

Once it has been created and registered on the target system, it can be added through the AddTab() method of the [Repository](#) object.

While it is possible to call AddTab() from any automation client, it is likely that you will want to call it from an add-in, and that add-in will be defined in the same OCX that provides the custom view.

C# example code is shown here:

```
public class Addin
{
    UserControl1 m_MyControl;

    public void EA_Connect(EA.Repository Rep)
    {
    }

    public object EA_GetMenuItems(EA.Repository Repository, string Location, string MenuName)
    {
        if ( MenuName == "" )
            return "-&C# Control Demo";
        else
        {
            String[] ret = {"&Create", "&ShowButton"};
            return ret;
        }
    }

    public void EA_MenuClick(EA.Repository Rep, string Location, string MenuName, string ItemName)
    {
        if ( ItemName == "&Create" )
            m_MyControl = (UserControl1) Rep.AddTab( "C# Demo", "ContDemo.UserControl1" );
        else
            m_MyControl.ShowButton();
    }
}
```

15.3.7 MDG Add-Ins

MDG Add_Ins are specialized types of Add-Ins that have additional features and extra requirements for Add-In authors wishing to contribute to EA's goal of Model Driven Generation.

Unlike general Add-In events, MDG Add-In events are only sent to the add-in that has taken ownership of a EA model branch on a particular PC.

One of the additional responsibilities of an MDG Add-In is to take ownership of a branch of an EA model which is done through the [MDG_Connect](#) event.

MDG Add-Ins identify themselves as such during [EA_Connect](#) by returning the string "MDG".

Unlike ordinary Add-Ins, responding to MDG Add-In events is not optional, and methods need to be published for each of the [MDG Events](#).

Two examples of MDG Add-Ins are the commercially available MDG Link for Eclipse and MDG Link for Visual Studio, published by [Sparx Systems](#).

15.3.7.1 MDG Events

An MDG Add-In must respond to all MDG Events, these events usually identify processes such as Build, Run, Synchronize, PreMerge, PostMerge and many others.

An MDG Link Add-In is expected to implement some form of forward and reverse engineering capability within EA - and as such requires access to a specific set of events - all to do with generation, synchronization and general processes concerned with converting Models to Code and Code to Models.

See Also

- [MDG_BuildProject](#)
- [MDG_Connect](#)
- [MDG_Disconnect](#)
- [MDG_GetConnectedPackages](#)
- [MDG_GetProperty](#)
- [MDG_Merge](#)
- [MDG_NewClass](#)
- [MDG_PostGenerate](#)
- [MDG_PostMerge](#)
- [MDG_PreGenerate](#)
- [MDG_PreMerge](#)
- [MDG_PreReverse](#)
- [MDG_RunExe](#)
- [MDG_View](#)

15.3.7.1.1 MDG Add-Ins MDGBuild Project

MDG_BuildProject

The MDG_BuildProject allows the Add-In to handle file changes caused by generation.

Syntax

Sub MDG_BuildProject(*Repository As EA.Repository, PackageGuid As String*)

The MDG_BuildProject function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-----------|------|-----------|-------------|
|-----------|------|-----------|-------------|

| | | | |
|--------------------|---------------|----|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package subtree that is controlled by the Add-In. |

Return Value

None

Details

This function will be called in response to a user selecting "Build Project" from the Add-ins menu. Respond to this event by compiling the project source files into a running application.

See Also

- [MDG_RunExe](#)

15.3.7.1.2 MDG Add-Ins MDGConnect**MDG Connect**

The MDG_Connect allows the Add-In to handle user driven request to connect a model branch to an external application.

Syntax

Function MDG_Connect(*Repository As EA.Repository, PackageID as Long, PackageGuid As String*) As Long

The **MDG_Connect** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|------------------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageID</i> | Long | IN | The PackageID of the EA package the user has requested to have connected to an external project. |

| | | | |
|--------------------|--------|----|---|
| <i>PackageGuid</i> | String | IN | The unique ID identifying the project provided by the add-in when a connection to a project branch of an EA model was first established |
|--------------------|--------|----|---|

Return Value

Return a non-zero to indicate that a connection has been made, a zero to indicates that the user has not nominated a project and connection should not proceed

Details

This function will be called when the user attempts to connect a particular EA package to an as yet unspecified external project. This event allows the Add-In to interact with the user to specify such a project.

The Add-In is responsible for retaining the connection details, which should be stored on a per-user or per-workstation basis. i.e. Users who share a common EA model over a network should be able to connect and disconnect to external projects independently of one another.

The Add-In should therefore not store connection details in an EA repository. A suitable place to store such details would be `SHGetFolderPath(..CSIDL_APPDATA..)AddinName`.

The `PackageGuid` parameter is the same identifier as required for most events relating to the MDG Add-In. Therefore it is recommended that the connection details be indexed using the `PackageGuid` value.

The `PackageID` parameter is provided to aid fast retrieval of package details from EA should this be required.

See Also

- [MDG Disconnect](#)

15.3.7.1.3 MDG Add-Ins MDGDisconnect

MDG Disconnect

The `MDG_Disconnect` allows the Add-In to respond to user requests to disconnect the model branch from an external project.

Syntax

Function `MDG_Disconnect(Repository As EA.Repository, PackageGuid As String)` As Long

The `MDG_Disconnect` function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |

Return Value

Return a non-zero to indicate that a disconnection has occurred allowing EA to update the user interface. A zero to indicates that the user has not disconnected from an external project.

Details

This function will be called when the user has attempts to disconnect an associated external project. The Add-In is required to delete the details the connection.

See Also

- [MDG_Connect](#)

15.3.7.1.4 MDG Add-Ins MDGGetConnectedPackages

MDG_GetConnectedPackages

The MDG_GetConnectedPackages allows the Add-In to return a list of current connection between EA and an external application.

Syntax

Function MDG_GetConnectedPackages(*Repository As EA.Repository*) As Variant

The **MDG_GetConnectedPackages** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

Return Value

Return an array of GUID strings representing individual EA packages.

Details

This function will be called when the Add-In is first loaded, and is expected to return a list of the available connections to external projects for this Add-In.

See Also

- [MDG_Connect](#)

15.3.7.1.5 MDG Add-Ins MDGGetProperty

MDG_GetProperty

MDG_GetProperty provides miscellaneous Add-In details to EA.

Syntax

Function MDG_GetProperty(*Repository As EA.Repository, PackageGuid As String, PropertyName As String*) As Variant

The **MDG_GetProperty** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|---------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
| <i>PropertyName</i> | String | IN | The name of the property that will be used by EA See details for the possible values |

Return Value

See Details

Details

This function will be called by EA to poll the Add-In for information relating to the PropertyName. This event should occur in as short a duration as possible as EA does not cache the information provided by the function.

Values corresponding to the following PropertyNames need to be provided:

IconID

Return the name of a DLL and a resource identifier in the format #ResID, where the resource ID indicates an Icon e.g. c:\program files\myapp\myapp.dll#101

Language

Return the default language that classes should be assigned when they are created in EA.

HiddenMenus

Return one or more values from the MDGMenus enumeration to hide menus that do not apply to your Add-In. eg.

```
if( PropertyName == "HiddenMenus" )
    return mgBuildProject + mgRun;
```

15.3.7.1.6 MDG Add-Ins MDGMerge

MDG Merge

The MDG_Merge allows the Add-In to jointly handle changes to both the model branch and the code project that the model branch is connected to.

Syntax

Function MDG_Merge(*Repository As EA.Repository, PackageGuid As String, SynchObjects As Variant, SynchType As Variant, ExportObjects As Variant, ExportFiles As Variant, ImportFiles As Variant, IgnoreLocked As Variant, Language As String*) As Long

The **MDG_Merge** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

| | | | |
|----------------------|---------|-----|--|
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
| <i>SynchObjects</i> | Variant | OUT | A string array containing a list of objects (Object ID format) to be jointly synchronized between the model branch and the project. Refer below for the format of the Object IDs |
| <i>SynchType</i> | Variant | OUT | The integer value determining the user-selected type of synchronization to take place. Refer below for a list of valid values |
| <i>ExportObjects</i> | Variant | OUT | The string array containing the list of new model objects (in Object ID format) to be exported by EA to the code project. |
| <i>ExportFiles</i> | Variant | OUT | A string array containing the list of files for each model object chosen for export by the add-in. Each entry in this array shall have a corresponding entry in the ExportObjects parameter at the same array index, so ExportFiles(2) shall contain the filename of the object by ExportObjects(2). |
| <i>ImportFiles</i> | Variant | OUT | A string array containing the list of code files made available to the code project to be newly imported to the model. EA will import each file listed in this array for import into the connected model branch. |
| <i>IgnoreLocked</i> | Variant | OUT | A boolean value containing the user-selected option to ignore any files locked by the code project. |
| <i>Language</i> | String | OUT | The string value containing the name of the code language supported by the code project connected to the model branch. |

Return Value

Return a non-zero if the Merge operation completed successfully and a zero value when the operation has been unsuccessful.

Details

This event shall be called whenever the user has asked to merge their model branch with its connected code project, or whenever the user has established a new connection to a code project. The purpose of this event is to allow the add-in to interact with the user to perform a merge between the model branch and the connected project.

Merge

A merge comprises of three major operations:

- Export: Where newly created model objects are exported into code and made available to the code project.
- Import: Where newly created code objects, classes, etc... are imported into the model.
- Synchronize: Where objects available both to the model and in code are jointly updated to reflect changes made in either the model, code project or both.

Synchronize Type

The Synchronize operation can take place in one of four different ways, each of these ways correspond to a value returned by SynchType:

- None: (SynchType = 0) No synchronization is to be performed.

- Forward: (SynchType = 1) Forward synchronization, between the model branch and the code project is to occur.
- Reverse: (SynchType = 2) Reverse synchronization, between the code project and the model branch is to occur.
- Both: (SynchType = 3) Reverse, then Forward synchronization's are to occur.

Object ID Format

Each of the Object IDs listed in the string arrays described above shall be composed of the following format: (@namespace)*(#class)*(\$attribute|%operation|:property)*

See Also

- [MDG_Connect](#)
- [MDG_PreMerge](#)
- [MDG_PostMerge](#)

15.3.7.1.7 MDG Add-Ins MDGNewClass

MDG_NewClass

The MDG_NewClass allows the Add-In to alter details of a class before it is created.

Syntax

Function MDG_NewClass(Repository As EA.Repository, PackageGuid As String, CodeID As String, Language As String) As String

The **MDG_NewClass** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
| <i>CodeID</i> | String | IN | A string used to identify the code element before it is created, for more information see MDG_View . |
| <i>Language</i> | String | IN | A string used to identify the programming language for the new class. The language needs to be supported by EA. |

Return Value

Return a string containing the file path that should be assigned to the class.

Details

This method is called when EA generates a new class, and needs information relating to assigning the language and file path. The file path should be passed back as a return value and the language should be passed back via the language parameter.

See Also

- [MDG_PreGenerate](#)

15.3.7.1.8 MDG Add-Ins MDGPostGenerate

MDG PostGenerate

The MDG_PostGenerate allows the Add-In to handle file changes caused by generation.

Syntax

Function MDG_PostGenerate(*Repository As EA.Repository, PackageGuid As String, FilePath As String, FileContents As String*) As Long

The **MDG_PostGenerate** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|---------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
| <i>FilePath</i> | String | IN | The path of the file EA intends to overwrite. |
| <i>FileContents</i> | String | IN | A string containing the proposed contents of the file. |

Return Value

Return a value depends on the type of event that this function is responding to (see Details). This function is required to handle two sperate and distinct cases.

Details

This event will be called after EA has prepared text to replace the existing contents of a file. Responding to this event will allow the add-in to write to the linked application's user interface rather than modify the file directly.

When the contents of a file is changed, EA will pass FileContents as a non-empty string. New files created as a result of code generation are also sent through this mechanism, allowing Add-Ins to add new files to the linked project's file list. When new files are created EA will pass FileContents as an empty string. When a non - zero is returned by this function, the Add-In has successfully written the contents of the file. A zero value for the return indicates to EA that the file needs to be saved.

See Also

- [MDG_PreGenerate](#)

15.3.7.1.9 MDG Add-Ins MDGPostMerge

MDG PostMerge

The MDG_PostMerge is called after a merge process has been completed.

Syntax

Function MDG_PostMerge(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_PostMerge** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|-------------------|-----------|--|
| <i>Repository</i> | EA. Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |

Return Value

Return a zero value if the post-merge process has failed, a non-zero return indicates that the post-merge has been successful. EA assumes a non-zero return if this method is not implemented

Details

This function will be called by EA after the merge process has been completed.

Note: File save checking should not be performed with this function, but should be handled by Pre/Post Generate and Reverse.

See Also

- [MDG_PreMerge](#)
- [MDG_PreGenerate](#)
- [MDG_PostGenerate](#)
- [MDG_PreReverse](#)
- [MDG_Merge](#)

15.3.7.1.10 MDG Add-Ins MDGPreGenerate

MDG_PreGenerate

The MDG_PreGenerate allows the Add-In to deal with unsaved changes.

Syntax

Function MDG_PreGenerate(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_PreGenerate** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|-------------------|-----------|--|
| <i>Repository</i> | EA. Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |

Return Value

Return a zero value to abort generation. Any other value will allow the generation to continue.

Details

This function will be called immediately before EA attempts to generate files from the model. A possible use of this function would be to prompt the user to save unsaved source files.

See Also

- [MDG_PostGenerate](#)

15.3.7.1.11 MDG Add-Ins MDGPreMerge

MDG PreMerge

The MDG_PreMerge is called after merge process has been initiated by the user and before EA performs the merge process.

Syntax

Function MDG_PreMerge(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_PreMerge** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|---------------|------------------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |

Return Value

A return value of zero indicates that the merge process will not occur, if the value is not zero the merge process will go ahead. If this method is not implemented then it is assumed that a merge process will be used.

Details

This event will be called after a user has performed their interactions with the merge screen and has confirmed the merge with the **OK** button, but before EA performs the merge process using the data provided by the MDG_Merge call, before any changes have been made to the model or the connected project.

This event is made available to provide the Add-In the opportunity to generally set internal Add-In flags to augment the MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse events.

Note: File save checking should not be performed with this function, but should be handled by Pre/Post Generate and Reverse.

See Also

- [MDG_PreGenerate](#)
- [MDG_PostGenerate](#)
- [MDG_PreReverse](#)
- [MDG_Merge](#)
- [MDG_PostMerge](#)

15.3.7.1.12 MDG Add-Ins MDGPreReverse**MDG PreReverse**

The MDG_PreReverse allows the Add-In to save file changes before being imported into EA.

Syntax

Sub MDG_PreReverse(*Repository As EA.Repository, PackageGuid As String, FilePaths As Variant*)

The **MDG_PreReverse** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
| <i>FilePaths</i> | String array | IN | An array of filepaths pointed to the files that are to be reverse engineered |

Return Value

None.

Details

This function will be with a list of files that about to be reverse-engineered into EA. If the user is working on unsaved versions of these files in an editor, you may wish to prompt the user or save automatically.

See Also

- [MDG PostGenerate](#)
- [MDG PreGenerate](#)

15.3.7.1.13 MDG Add-Ins MDGRunExe**MDG RunExe**

The MDG_RunExe allows the Add-In to run the target application.

Syntax

Sub MDG_RunExe(*Repository As EA.Repository, PackageGuid As String*)

The **MDG_RunExe** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|-------------------|---------------|-----------|--|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |

| | | | |
|--------------------|--------|----|--|
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
|--------------------|--------|----|--|

Return Value

None

Details

This function will be called in response to a user selecting "Run Exe" from the Add-ins menu. Respond to this event by launching the compiled application.

See Also

- [MDG_BuildProject](#)

15.3.7.1.14 MDG Add-Ins MDGView**MDG View**

The MDG_View allows the Add-In to display user specified code elements.

Syntax

Function MDG_View(*Repository As EA.Repository, PackageGuid As String, CodeID as String*) As Long

The **MDG_View** function syntax contains the following elements

| Parameter | Type | Direction | Description |
|--------------------|---------------|------------------|---|
| <i>Repository</i> | EA.Repository | IN | An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information. |
| <i>PackageGuid</i> | String | IN | The GUID identifying the EA package sub-tree that is controlled by the Add-In. |
| <i>CodeID</i> | String | IN | <p>A CodeID identifies the code element in the following format:</p> <pre><type>ElementPart<type>ElementPart...</pre> <p>where each element is preceded with a token identifying its type:</p> <ul style="list-style-type: none"> @ - namespace # - class \$ - attribute % - operation <p>For example if a user has selected the m_Name attribute of Class1 located in namespace Name1, the class ID would be passed through in the following format:</p> <pre>@Name1#Class1%m_Name</pre> |

Return Value

Return a non-zero value to indicate that the Add-In has processed the request. Returning a zero value will result in EA employing the standard viewing process which is to launch the associated source file.

Details

This function will be called by EA when the user asks to view a particular code element. This allows the Add-In to present that element in its own way – usually in a code editor.

Part

16

16 Glossary of Terms

This section provides a detailed glossary for Enterprise Architect.

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#)

16.1 A (Glossary)

~A~

abstract class

A class that cannot be directly instantiated.

Contrast: concrete class

abstraction

The essential characteristics of an entity that distinguish it from all other kinds of entities. An abstraction defines a boundary relative to the perspective of the viewer.

action

The specification of an executable statement that forms an abstraction of a computational procedure. An action typically results in a change in the state of the system, and can be realized by sending a message to an object or modifying a link or a value of an attribute. .

action sequence

An expression that resolves to a sequence of actions.

action state

A state that represents the execution of an atomic action, typically the invocation of an operation.

activation

The execution of an action.

active class

A class whose instances are active objects. When instantiated, an active class will control its execution. Rather than being invoked or activated by other objects, it can operate standalone, and define its own thread of behavior.

See also: active object

activation

An object that owns a thread and can initiate control activity. An instance of active class.

See also: Active class, thread

activity

An activity defines the bounds for the structural organization that contains a set of basic or fundamental behaviors. It can be used to model procedural type application development for system design through to



modeling business processes in organizational structures and workflow.

activity diagram

An activity diagram can be used to model procedural type application development for system design through to modeling business processes in organizational structures and workflow.

activity graph

A special case of a state machine that is used to model processes involving one or more classifiers.

Contrast: state chart diagram

actor [class]

A coherent set of roles that users of use cases play when interacting with these use cases. An actor has one role for each use case with which it communicates.

actual parameter

Synonym: argument

aggregate [class]

A class that represents the "whole" in an aggregation (whole-part) relationship.

See also: aggregation

aggregation

A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part.

See also: composition

analysis

The part of the software development process whose primary purpose is to formulate a model of the problem domain. Analysis focuses on what to do, design focuses on how to do it.

Contrast: design

analysis diagram

An *Analysis diagram* is used to capture high level business processes and early models of system behavior and elements. It is less formal than some other diagrams, but provides a good means of capturing the essential business characteristics and needs.

analysis time

Refers to something that occurs during an analysis phase of the software development process.

See also: design time, modeling time

architecture

The organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

argument

A binding for a parameter that resolves to a run-time instance.

Synonym: actual parameter

Contrast: parameter

artifact

A physical piece of information that is used or produced by a software development process. Examples of Artifacts include models, source files, scripts, and binary executable files. An artifact may constitute the implementation of a deployable component.

Synonym: product

Contrast: component

assembly

An assembly connector bridges the required interface of a component with the provided interface of a second component.

association

The semantic relationship between two or more classifiers that specifies connections among their instances.

association class

A model element that has both association and class properties. An association class can be seen as an association that also has class properties, or as a class that also has association properties.

association end

The endpoint of an association, which connects the association to a classifier.

attribute

A feature within a classifier that describes a range of values that instances of the classifier may hold.

auxiliary class

A stereotyped class that supports another more central or fundamental class, typically by implementing secondary logic or control flow. Auxiliary classes are typically used together with focus classes, and are particularly useful for specifying the secondary business logic or control flow of components during design.

See also: focus

16.2 B (Glossary)

~B~

behavior

The observable effects of an operation or event, including its results.

behavioral diagram

Behavioral diagrams depict the behavioral features of a system or business process. Behavioral diagrams include Activity diagrams, State Machine diagrams, Communication diagrams, Interaction Overview diagrams, Sequence diagrams, Timing diagrams and Use Case diagrams.



behavioral feature

A dynamic feature of a model element, such as an operation or method.

behavioral model aspect

A model aspect that emphasizes the behavior of the instances in a system, including their methods, collaborations, and state histories.

binary association

An association between two classes. A special case of an n-ary association.

binding

The creation of a model element from a template by supplying arguments for the parameters of the template.

bookmark

A marker in a rich text document that allows you to link inner sections of a document into a master document (using Word insert file function).

boolean

An enumeration whose values are true and false.

boolean expression

An expression that evaluates to a boolean value.

boundary

1. A *Boundary* is a stereotyped class that models some system boundary – typically a user interface screen. It is used in the conceptual phase to capture users interacting with the system at a screen level (or some other boundary interface type). It is often used in sequence and robustness (analysis) diagrams. It is the View in the Model-View-Controller pattern.
2. A *System Boundary* element is used to delineate a particular part of the system.

16.3 C (Glossary)

~C~

C++

An object-oriented programming language based on the earlier 'C' language.

call

An action state that invokes an operation on a classifier.

cardinality

The number of elements in a set.

Contrast: multiplicity

CASE

Computer Aided Software Engineering. A tool designed for the purpose of modeling and building



software systems.

child

In a generalization relationship, the specialization of another element, the parent.

See also: subclass, subtype.

Contrast: parent

choice

The choice pseudo-state is used to compose complex transitional paths, where the outgoing transition path is decided by dynamic, run-time conditions determined by the actions performed by the state machine on the path leading to the choice.

class

A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment.

See also: interface

class diagram

A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.

classification

The assignment of an object to a classifier. See dynamic classification, multiple classification and static classification.

classifier

A mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, datatypes, and components.

client

A classifier that requests a service from another classifier.

Contrast: supplier

collaboration

The specification of how an operation or classifier, such as a use case, is realized by a set of classifiers and associations playing specific roles used in a specific way. The collaboration defines an interaction.

See also: interaction

collaboration diagram

Used pre - UML 2.0.

collaboration occurrence

Use an Occurrence to apply a pattern defined by a collaboration to a specific situation.

combined fragment

A combined fragment reflects a piece or pieces of interaction (called interaction operands) controlled by

an interaction operator, whose corresponding boolean conditions are known as interaction constraints. It appears graphically as a transparent window, divided by horizontal dashed lines for each operand.

comment

An annotation attached to an element or a collection of elements. A note has no semantics.

Contrast: constraint

communication diagram

A Communication diagram shows the interactions between elements at run-time in much the same manner as a sequence diagram. However, communication diagrams are used to visualize inter-object relationships, while sequence diagrams are more effective at visualizing processing over time.

compile time

Refers to something that occurs during the compilation of a software module.

See also: modeling time, run time

component

A modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component is typically specified by one or more classifiers (e.g., implementation classes) that reside on it, and may be implemented by one or more artifacts (e.g., binary, executable, or script files).

Contrast: artifact

component diagram

A diagram that shows the organizations and dependencies among components.

composite [class]

A class that is related to one or more classes by a composition relationship.

See also: composition

composite state

A state that consists of either concurrent (orthogonal) substates or sequential (disjoint) substates.

See also: substate

composite structure diagram

A composite structure diagram reflects the internal collaboration of classes, interfaces, or components to describe a functionality. Composite structure diagrams are similar to class diagrams, except that they model a specific usage of the structure.

composition

A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive.

Synonym: composite aggregation

concrete class

A class that can be directly instantiated.

Contrast: abstract class

concurrency

The occurrence of two or more activities during the same time interval. Concurrency can be achieved by interleaving or simultaneously executing two or more threads.

See also: thread

concurrent substate

A substate that can be held simultaneously with other substates contained in the same composite state.

See also: composite state

Contrast: disjoint substate

connector

A logical link between model elements. May be structural, dynamic or possessive.

constraint

A semantic condition or restriction. Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML.

See also: tagged value, stereotype

constraint

A rule or condition that applies to some element. It is often modeled as a pre- or post- condition.

container

1. An instance that exists to contain other instances, and that provides operations to access or iterate over its contents.(for example, arrays, lists, sets).
2. A component that exists to contain other components.

containment hierarchy

A namespace hierarchy consisting of model elements, and the containment relationships that exist between them. A containment hierarchy forms a graph.

context

A view of a set of related modeling elements for a particular purpose, such as specifying an operation.

continuation

A Continuation is used in seq and alt combined fragments, to indicate the branches of continuation an operand follows.

control

A Control is a stereotyped class that represents a controlling entity or manager. A control organizes and schedules other activities and elements. It is the controller of the Model-View-Controller pattern.

control flow

The control flow is a connector linking two nodes in an activity diagram. Control Flow connectors start a nodes activity when the preceding nodes action is finished.

16.4 D (Glossary)

~D~



database schema

A database schema is the description of a database structure. It defines tables and fields and the relationship between them.

datastore

A datastore element is used to define permanently stored data. A token of data that is stored in the Datastore is stored permanently. A token of data that comes out of the Datastore is a copy of the original data. The tokens imported are kept for the life of the Activity in which it exists.

datatype

A descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations.

decision

A Decision is an element of an Activity diagram that indicates a point of conditional progression: if a condition is true, then processing continues one way, if not, then another.

defining model [MOF]

The model on which a repository is based. Any number of repositories can have the same defining model.

delegate

A delegate connector defines the internal assembly of a component's external ports and interfaces. Using a delegate connector wires the internal workings of the system to the outside world, by a delegation of the external interfaces' connections.

delegation

The ability of an object to issue a message to another object in response to a message. Delegation can be used as an alternative to inheritance.

Contrast: inheritance

dependency

A relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).

deployment

A deployment is a type of dependency relationship that indicates the deployment of an artifact onto a node or executable target.

deployment diagram

A diagram that shows the configuration of run-time processing nodes and the components, processes, and objects that live on them. Components represent run-time manifestations of code units.

See also: component diagrams

deployment spec

A deployment specification specifies parameters guiding deployment of an artifact, as is common with most hardware and software technologies.

derived element

A model element that can be computed from another element, but that is shown for clarity or that is included for design purposes even though it adds no semantic information.

design

The part of the software development process whose primary purpose is to decide how the system will be implemented. During design strategic and tactical decisions are made to meet the required functional and quality requirements of a system.

design time

Refers to something that occurs during a design phase of the software development process.

See also: modeling time

Contrast: analysis time

development process

A set of partially ordered steps performed for a given purpose during software development, such as constructing models or implementing models.

diagram

A graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs (relationships) and vertices (other model elements). UML supports the following diagrams: class diagram, object diagram, use case diagram, sequence diagram, collaboration diagram, state diagram, activity diagram, component diagram, and deployment diagram.

diagram gate

A diagram gate is a simple graphical way to indicate the point at which messages can be transmitted into and out of interaction fragments.

diagram view

The workspace area where the UML diagrams are displayed.

disjoint substate

A substate that cannot be held simultaneously with other substates contained in the same composite state.

See also: composite state

Contrast: concurrent substate

distribution unit

A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.

domain

An area of knowledge or activity characterized by a set of concepts and terminology understood by

practitioners in that area.

dynamic classification

A semantic variation of generalization in which an object may change its classifier.

Contrast: static classification

16.5 E (Glossary)

~E~

element

An atomic constituent of a model.

element

A model object of any type - class, component, node, object or etc.

endpoint

An endpoint is used in interaction diagrams to reflect a lost message in sequence.

entity

An Entity is a store or persistence mechanism that captures the information or knowledge in a system. It is the Model in the Model-View-Controller pattern.

entry action

An action executed upon entering a state in a state machine regardless of the transition taken to reach that state.

entry point

Entry points are used to define where external states can enter a submachine.

enumeration

A list of named values used as the range of a particular attribute type. For example, RGBColor = {red, green, blue}. Boolean is a predefined enumeration with values from the set {false, true}.

event

The specification of a significant occurrence that has a location in time and space. In the context of state diagrams, an event is an occurrence that can trigger a transition.

exception handler

The exception handler element defines the group of operations to carry out when an exception occurs.

exit action

An action executed upon exiting a state in a state machine regardless of the transition taken to exit that state.

exit point

Exit points are used in submachine states and state machines to denote the point where the machine will



be exited and the transition sourcing this exit point, for submachines, will be triggered. Exit points are a type of pseudo-state used in the state machine diagram.

export

In the context of packages, to make an element visible outside its enclosing namespace.

See also: visibility

Contrast: export [OMA], import

expose interface

The expose interface toolbox element is a graphical way to depict the required and supplied interfaces of a component, class, or part.

expression

A string that evaluates to a value of a particular type. For example, the expression "(7 + 5 * 3)" evaluates to a value of type number. A relationship from an extension use case to a base use case, specifying how the behavior defined for the extension use case augments (subject to conditions specified in the extension) the behavior defined for the base use case. The behavior is inserted at the location defined by the extension point in the base use case. The base use case does not depend on performing the behavior of the extension use case. See extension point, include.

extend

An Extend connector is used to indicate an element extends the behavior of another. Extensions are used in use case models to indicate one use case (optionally) extends the behavior of another.

16.6 F (Glossary)

~F~

facade

A stereotyped package containing only references to model elements owned by another package. It is used to provide a 'public view' of some of the contents of a package.

feature

A property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a datatype.

final

A final pseudo-state indicates an end.

final state

A special kind of state signifying that the enclosing composite state or the entire state machine is completed.

fire

To execute a state transition.

See also: transition

flow final

The flow final element depicts an exit from the system as opposed to the Activity Final which represents the completion of the activity.

focus class

A stereotyped class that defines the core logic or control flow for one or more auxiliary classes that support it. Focus classes are typically used together with one or more auxiliary classes, and are particularly useful for specifying the core business logic or control flow of components during design.

See also: auxiliary

focus of control

A symbol on a sequence diagram that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

forward engineering

The process of generating source code from the UML model.

fork

Forks are utilized in State Machine diagrams as pseudo-states. With respect to state machine diagrams, a fork pseudo-state signifies that its incoming transition will come from a single state, and it will have multiple outgoing transitions.

framework

A stereotyped package that contains model elements which specify a reusable architecture for all or part of a system. Frameworks typically include classes, patterns or templates. When frameworks are specialized for an application domain, they are sometimes referred to as application frameworks.

See also: pattern

16.7 G (Glossary)

~G~**generalizable element**

A model element that may participate in a generalization relationship.

See also: generalization

generalization

A taxonomic relationship between a more general element and a more specific element. The more specific element is fully consistent with the more general element and contains additional information. An instance of the more specific element may be used where the more general element is allowed.

See also: inheritance

guard condition

A condition that must be satisfied in order to enable an associated transition to fire.



16.8 H (Glossary)

~H~

history state

There are two types of history pseudo-states defined in UML, shallow and deep history. A shallow history sub-state is used to represent the most recently active sub-state of a composite state. A deep history sub-state, in contrast, reflects the most recent active configuration of the composite state.



16.9 I (Glossary)

~I~

implementation

A definition of how something is constructed or computed. For example, a class is an implementation of a type, a method is an implementation of an operation.

implementation class

A stereotyped class that specifies the implementation of a class in some programming language (e.g., C++, Smalltalk, Java) in which an instance may not have more than one class. An Implementation class is said to realize a type if it provides all of the operations defined for the type with the same behavior as specified for the type's operations.

See also: type

implementation inheritance

The inheritance of the implementation of a more general element. Includes inheritance of the interface.

Contrast: interface inheritance

import

In the context of packages, a dependency that shows the packages whose classes may be referenced within a given package (including packages recursively embedded within it).

Contrast: export

include

A relationship from a base use case to an inclusion use case, specifying how the behavior for the base use case contains the behavior of the inclusion use case. The behavior is included at the location which is defined in the base use case. The base use case depends on performing the behavior of the inclusion use case, but not on its structure (ie., attributes or operations).

See also: extend

inheritance

The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior.

See also: generalization

initial state

The Initial pseudo-state is used to denote the default state of a composite state; there can be one initial vertex in each region of the composite state.



interaction diagram

Interaction diagrams can be sequence diagrams, communication diagrams, interaction overview diagrams, and timing diagrams. Interaction diagrams include Timing Diagrams, Sequence Diagrams, Interaction Overview Diagrams and Communication Diagrams.

instance

An entity that has unique identity, a set of operations that can be applied to it, and state that stores the effects of the operations.

See also: object

interaction

A specification of how stimuli are sent between instances to perform a specific task. The interaction is defined in the context of a collaboration.

See also: collaboration

interaction diagram

A generic term that applies to several types of diagrams that emphasize object interactions. These include collaboration diagrams and sequence diagrams.

interaction occurrence

An interaction occurrence is a reference to an existing interaction element. Interaction occurrences are visually represented by a frame, with "ref" in the frame's title space. The diagram name is indicated in the frame contents.

interaction overview diagram

Interaction Overview diagrams visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose. As interaction overview diagrams are a variant of activity diagrams, most of the diagram notation is similar, as is the process in constructing the diagram.

interface

A named set of operations that characterize the behavior of an element.

interface inheritance

The inheritance of the interface of a more general element. Does not include inheritance of the implementation.

Contrast: implementation inheritance

internal transition

A transition signifying a response to an event without changing the state of an object.

interrupt flow

A EA defined toolbox element used to define the exception handler and interruptible activity region concepts.

16.10 J (Glossary)

~J~

Java

A fully object-oriented, cross platform language based on elements from Smalltalk, C++ and other OO languages.

join

Joins are utilized in state machine diagrams and in activity diagrams to synchronize multiple flows.

junction

Junction pseudo-states are used to design complex transitional paths. A junction can be used to combine, or merge, multiple paths into a shared transition path or to split an incoming path into multiple paths.



16.11 L (Glossary)

~L~

layer

The organization of classifiers or packages at the same level of abstraction. A layer represents a horizontal slice through an architecture, whereas a partition represents a vertical slice.

Contrast: partition

lifeline

A lifeline is an individual participant in an interaction (i.e., lifelines cannot have multiplicity). A lifeline represents a distinct connectable element.

link

A semantic connector among a tuple of objects. An instance of an association.

See also: association

link end

An instance of an association end.

See also: association end

local path

A relative path on a local machine. Allows developers to store shared source code in machine specific directories, but still generate and synchronize code.



16.12 M (Glossary)

~M~

maintenance

The support of a software system after it is deployed.



manifest

A manifest relationship indicates that the artifact source embodies the target model element. Stereotypes can be added to Enterprise Architect to classify the type of manifestation of the model element.

message

Messages indicate a flow of information, or transition of control, between elements. Messages are used by Communication diagrams, Sequence diagrams, Interaction Overview diagrams and Timing diagrams.

message endpoint

The Message Endpoint element defines an endpoint of a Lifeline such a State or Value Lifeline in a Timing diagram.

message label

A message label is used for messages sent between lifelines to make the diagram appear less cluttered. Labels with the same name indicate that a message may be interrupted.

metaclass

A class whose instances are classes. Metaclasses are typically used to construct metamodels.

metafile

A vector based image format native to Windows. Supports high detail and excellent scaling. Typically used for saving diagram images for placement in documents. Comes in Placeable (an older format) and Enhanced (current standard format).

meta-metamodel

A model that defines the language for expressing a metamodel. The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.

metamodel

A model that defines the language for expressing a model.

metaobject

A generic term for all metaentities in a metamodeling language. For example, metatypes, metaclasses, metaattributes, and metaassociations.

method

The implementation of an operation. It specifies the algorithm or procedure associated with an operation.

model [MOF]

An abstraction of a physical system with a certain purpose.

See also: physical system

Usage note: In the context of the MOF specification, which describes a meta-metamodel, for brevity the meta-metamodel is frequently to as simply the model.

model aspect

A dimension of modeling that emphasizes particular qualities of the metamodel. For example, the structural model aspect emphasizes the structural qualities of the metamodel.

model elaboration

The process of generating a repository type from a published model. Includes the generation of interfaces and implementations which allows repositories to be instantiated and populated based on, and in compliance with, the model elaborated.

model element [MOF]

An element that is an abstraction drawn from the system being modeled.

Contrast: view element. In the MOF specification model elements are considered to be metaobjects.

model library

A stereotyped package that contains model elements which are intended to be reused by other packages. A model library differs from a profile in that a model library does not extend the metamodel using stereotypes and tagged definitions. A model library is analogous to a class library in some programming languages.

modeling time

Refers to something that occurs during a modeling phase of the software development process. It includes analysis time and design time. Usage note: When discussing object systems, it is often important to distinguish between modeling-time and run-time concerns.

See also: analysis time, design time

Contrast: run time

module

A software unit of storage and manipulation. Modules include source code modules, binary code modules, and executable code modules.

See also: component

multiple classification

A semantic variation of generalization in which an object may belong directly to more than one classifier.

See also: static classification, dynamic classification

multiple inheritance

A semantic variation of generalization in which a type may have more than one supertype.

Contrast: single inheritance

multiplicity

A specification of the range of allowable cardinalities that a set may assume. Multiplicity specifications may be given for roles within associations, parts within composites, repetitions, and other purposes. Essentially a multiplicity is a (possibly infinite) subset of the non-negative integers.

Contrast: cardinality

multi-valued [MOF]

A model element with multiplicity defined whose Multiplicity Type:: upper attribute is set to a number greater than one. The term multi-valued does not pertain to the number of values held by an attribute, parameter, etc. at any point in time.

Contrast: single-valued

16.13 N (Glossary)

~N~

name

A string used to identify a model element.

namespace

A part of the model in which the names may be defined and used. Within a namespace, each name has a unique meaning.

See also: name

n-ary association

An association among three or more classes. Each instance of the association is an n-tuple of values from the respective classes.

Contrast: binary association

nesting

The nesting connector is an alternative membership notation used to indicate nested members within an element, for example, a package which has nested members. The nested members of a package could also be shown inside the packaged rather than linked by the nesting connector.

node

A node is classifier that represents a run-time computational resource, which generally has at least a memory and often processing capability. Run-time objects and components may reside on nodes.



16.14 O (Glossary)

~O~

object

An entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines. An object is an instance of a class.

See also: class, instance

object diagram

A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a collaboration diagram.

See also: class diagram, collaboration diagram

object flow

An Object Flow is a sub type of the State Flow or Transition. It implies the passing of an object instance between elements at run-time.

object flow state



A state in an activity graph that represents the passing of an object from the output of actions in one state to the input of actions in another state.

object lifeline

A line in a sequence diagram that represents the existence of an object over a period of time.

See also: sequence diagram

Object Management Group

The standards body responsible for the UML specification and management. Their website is www.omg.org - follow the links to the UML pages.

object toolbar

The main toolbar running down the centre of EA from which you can select model elements to insert into diagrams. This is also known as the UML Toolbox and the Toolbox.

occurrence

An occurrence relationship indicates that a collaboration represents a classifier. An occurrence connector is drawn from the collaboration to the classifier.

operation

A service that can be requested from an object to effect behavior. An operation has a signature, which may restrict the actual parameters that are possible.

16.15 P (Glossary)

~P~

package

1. A package is a namespace as well as an element that can be contained in other package's namespaces. Packages can own or merge with other packages, and its elements can be imported into a package's namespace.
2. A logical container of model elements. Groups elements and may also contain other packages.

The OMG UML specifications states:

"A package is a grouping of model elements. Packages themselves may be nested within other packages. A package may contain subordinate packages as well as other kinds of model elements. All kinds of UML model elements can be organized into packages."

Note that packages own model elements and are the basis for configuration control, storage, and access control. Each element can be directly owned by a single package, so the package hierarchy is a strict tree. However, packages can reference other packages, modeled by using one of the stereotypes «import» and «access» of Permission dependency, so the usage network is a graph. Other kinds of dependencies between packages usually imply that one or more dependencies among the elements exists.

A package is shown as a large rectangle with a small rectangle (a "tab") attached to the left side of the top of the large rectangle. It is the common folder icon.

package diagram

Package diagrams are used to reflect the organization of packages and their elements, and provide a visualization of their corresponding namespaces.



package import

A package import relationship is drawn from a source package to a package whose contents will be imported. Private members of a target package cannot be imported.

package merge

A package merge indicates a relationship between two packages whereby the contents of the target package are merged with those of the source package. Private contents of a target package are not merged.

parameter

The specification of a variable that can be changed, passed, or returned. A parameter may include a name, type, and direction. Parameters are used for operations, messages, and events.

Synonym: formal parameter

Contrast: argument

parameterized element

The descriptor for a class with one or more unbound parameters.

Synonym: template, parameterized class

parent

In a generalization relationship, the generalization of another element, the child.

See also: subclass, subtype

Contrast: child

part

Parts are run-time instances of classes or interfaces.

participate

The connection of a model element to a relationship or to a reified relationship. For example, a class participates in an association, an actor participates in a use case.

partition

1. activity graphs: A portion of an activity graphs that organizes the responsibilities for actions.

See also: swim lane

2. architecture: A set of related classifiers or packages at the same level of abstraction or across layers in a layered architecture. A partition represents a vertical slice through an architecture, whereas a layer represents a horizontal slice.

Contrast: layer

pattern

A template collaboration.

persistent object

An object that exists after the process or thread that created it has ceased to exist.

physical system

1. The subject of a model.
2. A collection of connected physical units, which can include software, hardware and people, that are organized to accomplish a specific purpose. A physical system can be described by one or more models, possibly from different viewpoints.

Contrast: system

port

Ports define the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted by using the "Expose Interface" toolbox element.

postcondition

A constraint that must be true at the completion of an operation.

precondition

A constraint that must be true when an operation is invoked.

primitive type

A pre-defined basic datatype without any substructure, such as an integer or a string.

process

1. A heavyweight unit of concurrency and execution in an operating system.

Contrast: thread, which includes heavyweight and lightweight processes. If necessary, an implementation distinction can be made using stereotypes.

2. A software development process - the steps and guidelines by which to develop a system.
3. To execute an algorithm or otherwise handle something dynamically.

profile

A profile is a stereotyped package that contains model elements which have been customized for a specific domain or purpose using extension mechanisms, such as stereotypes, tagged definitions and constraints. A profile may also specify model libraries on which it depends and the metamodel subset that it extends.

project browser

The workspace window (top left) where the model contents are displayed in 'tree' format. Displays packages, diagrams, model elements, etc.

projection

A mapping from a set to a subset of it.

property

A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are predefined in the UML; others may be user defined.

See also: tagged value

pseudo-state

A vertex in a state machine that has the form of a state, but doesn't behave as a state. Pseudo-states include initial and history vertices.

published model [MOF]

A model that has been frozen, and becomes available for instantiating repositories and for the support in defining other models. A frozen model's model elements cannot be hanged.

16.16 Q (Glossary)

~Q~

qualifier

An association attribute or tuple of attributes whose values partition the set of objects related to an object across an association.



16.17 R (Glossary)

~R~

realize

A source object realizes the destination object. Realize is used to express traceability and completeness in the model – a business process or requirement is realized by one or more use cases which are in turn realized by some classes which in turn are realized by a component, etc.



receive [a message]

The handling of a stimulus passed from a sender instance.

See also: sender, receiver

receive

A Receive element is used to define the acceptance or receipt of a request. Movement on to next action does occur until it has received what is defined.

receiver [object]

The object handling a stimulus passed from a sender object.

Contrast: sender

reception

A declaration that a classifier is prepared to react to the receipt of a signal.

recursion

A recursion is a type of message used in sequence diagrams to indicate a recursive function.

reference

1. A denotation of a model element.
2. A named slot within a classifier that facilitates navigation to other classifiers.

Synonym: pointer

region

UML 2 supports both expansion regions and interruptible activity regions. An Expansion Region defines the bounds of an region consisting of one or more sets of input collections, where an input collection is a set of elements of the same type. An interruptible region contains activity nodes - when a token leaves an interruptible region, this terminates all of the regions tokens and behaviors.

refinement

A relationship that represents a fuller specification of something that has already been specified at a certain level of detail. For example, a design class is a refinement of an analysis class.

relationship

A semantic connection among model elements. Examples of relationships include associations and generalizations.

repository

A facility for storing object models, interfaces, and implementations.

represents

The Represents connector indicates a collaboration is used in a classifier. The connector is drawn from the collaboration to its owning classifier.

requirement

A desired feature, property, or behavior of a system.

responsibility

A contract or obligation of a classifier.

reuse

The use of a pre-existing artifact.

reverse engineering

The process of importing source code into the model as standard UML model elements (classes, attributes, operations, etc.).

rich text format

A standard mark-up language for creating word processor documents, frequently associated with Microsoft Word.

robustness diagram

Enterprise Architect supports business process modeling extensions from the UML business process model profile. Robustness diagrams are used in the Iconix Process - you can read more about this at www.sparxsystems.com.au/iconix/iconixsw.htm.

role

1. The named detail and rules associated with one end of an association. May indicate name, constraints, multiplicity and collection details.
2. The named specific behavior of an entity participating in a particular context. A role may be static (e.g., an association end) or dynamic (eg. a collaboration role).

role binding

Role Binding is the mapping between a collaboration occurrence's internal roles and the respective parts needed to implement a specific situation. The associated parts can have properties defined to enable the binding to occur, and the collaboration to take place.

run time

The period of time during which a computer program executes.

Contrast: modeling time

16.18 S (Glossary)

~S~**scenario**

A specific sequence of actions that illustrates behaviors. A scenario may be used to illustrate an interaction or the execution of a use case instance.

See also: interaction.

scenario

A sequence of operations carried out in some order to produce a known result. May apply to use cases where it is the equivalent of a sequence diagram, or to other objects to describe how they are used at run-time.

schema [MOF]

In the context of the MOF, a schema is analogous to a package which is a container of model elements. Schema corresponds to an MOF package.

Contrast: metamodel, package

self-message

A self-message reflects a new process or method invoked within the calling lifeline's operation. It is a specification of a message.

semantic variation point

A point of variation in the semantics of a metamodel. It provides an intentional degree of freedom for the interpretation of the metamodel semantics.

send [a message]

The passing of a stimulus from a sender instance to a receiver instance.

See also: sender, receiver

sender [object]

The object passing a stimulus to a receiver object.

Contrast: receiver

sequence diagram

A diagram that shows object interactions arranged in time sequence. In particular, it shows the objects

participating in the interaction and the sequence of messages exchanged. Unlike a collaboration diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and collaboration diagrams express similar information, but show it in different ways.

See also: collaboration diagram

signal

The specification of an asynchronous stimulus communicated between instances. Signals may have parameters.

signature

The name and parameters of a behavioral feature. A signature may include an optional returned parameter.

single inheritance

A semantic variation of generalization in which a type may have only one supertype.

Synonym: multiple inheritance [OMA]

Contrast: multiple inheritance

single valued [MOF]

A model element with multiplicity defined is single valued when its Multiplicity Type: upper attribute is set to one. The term single-valued does not pertain to the number of values held by an attribute, parameter, etc., at any point in time, since a single-valued attribute (for instance, with a multiplicity lower bound of zero) may have no value.

Contrast: multi-valued

specification

A declarative description of what something is or does.

Contrast: implementation

state

A condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.

Contrast: state [OMA]

state invariant

A State Invariant is a condition applied to a lifeline, which must be fulfilled for the lifeline to exist.

state machine

A behavior that specifies the sequences of states that an object or an interaction goes through during its life in response to events, together with its responses and actions.

state machine diagram

A State Machine diagram illustrates how an element, often a class, can move between states classifying its behavior, according to transition triggers, constraining guards, and other aspects of state machine diagrams that depict and explain movement and behavior.

state chart

diagram A diagram that shows a state machine.

See also: state machine

state continuation

The State/Continuation symbol serves two different purposes for interaction diagrams, as state invariants and as continuations. A State Invariant is a condition applied to a lifeline, which must be fulfilled for the lifeline to exist. A Continuation is used in seq and alt combined fragments, to indicate the branches of continuation an operand follows.

state lifeline

A State Lifeline follows discrete transitions between states, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations.

static classification

A semantic variation of generalization in which an object may not change classifier.

Contrast: dynamic classification

stereotype

A new type of modeling element that extends the semantics of the metamodel. Stereotypes must be based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user defined. Stereotypes are one of three extensibility mechanisms in UML.

See also: constraint, tagged value

stimulus

The passing of information from one instance to another, such as raising a signal or invoking an operation. The receipt of a signal is normally considered an event.

See also: message

string

A sequence of text characters. The details of string representation depend on implementation, and may include character sets that support international characters and graphics.

structural diagram

Structural diagrams depict the structural elements composing a system or function. These diagrams can reflect the static relationships of a structure, as do class or package diagrams, or run-time architectures, such as object or composite structure diagrams. Structural diagrams include Class diagrams, Composite Structure diagrams, Component diagrams, Deployment diagrams, Object diagrams and Package diagrams.

structural feature

A static feature of a model element, such as an attribute.

structural model aspect

A model aspect that emphasizes the structure of the objects in a system, including their types, classes, relationships, attributes, and operations.

subactivity state

A state in an activity graph that represents the execution of a non-atomic sequence of steps that has some duration.

subclass

In a generalization relationship, the specialization of another class; the superclass.

See also: generalization

Contrast: superclass

submachine state

A state in a state machine which is equivalent to a composite state but its contents is described by another state machine.

subpackage

A package that is contained in another package.

substate

A state that is part of a composite state.

See also: concurrent state, disjoint state

subsystem

A grouping of model elements that represents a behavioral unit in a physical system. A subsystem offers interfaces and has operations. In addition, the model elements of a subsystem can be partitioned into specification and realization elements.

See also: package, physical system

subtype

In a generalization relationship, the specialization of another type; the supertype.

See also: generalization

Contrast: supertype

superclass

In a generalization relationship, the generalization of another class; the subclass.

See also: generalization

Contrast: subclass

supertype

In a generalization relationship, the generalization of another type; the subtype.

See also: generalization

Contrast: subtype

supplier

A classifier that provides services that can be invoked by others.

Contrast: client

swimlane

A partition on a activity diagram for organizing the responsibilities for actions. Swimlanes typically correspond to organizational units in a business model.

See also: partition

synch

A synch state is useful for indicating concurrent paths of a state machine will be synchronized. After bringing the paths to a synch state, the emerging transition will indicate unison.

synchronize code

The process of importing and exporting code changes to ensure the model and source code match

system

A top-level subsystem in a model.

Contrast: physical system

system boundary

A System Boundary element is used to delineate a particular part of the system.

16.19 T (Glossary)

~T~**table**

A relational table (composed of columns).

tagged value

The explicit definition of a property as a name-value pair. In a tagged value, the name is referred as the tag. Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML.

See also: constraint, stereotype

template

Synonym: parameterized element

terminate

The terminate pseudostate indicates that upon entry of its pseudostate, the state machine's execution will end.

thread [of control]

A single path of execution through a program, a dynamic model, or some other representation of control flow. Also, a stereotype for the implementation of an active object as lightweight process.

See also: process



time event

An event that denotes the time elapsed since the current state was entered.

See also: event

time expression

An expression that resolves to an absolute or relative value of time.

timing diagram

The Timing diagram defines the behavior of different objects within a time-scale, with visual depictions of those objects changing state and interacting over time.

toolbox

The main toolbar running down the centre of EA from which you can select model elements to insert into diagrams. This is also known as the UML Toolbox and the Object Toolbar.

top level

A stereotype of package denoting the top-most package in a containment hierarchy. The topLevel stereotype defines the outer limit for looking up names, as namespaces "see" outwards. For example, opTopLevelSubsystem represents the top of the subsystem containment hierarchy.

trace

A dependency that indicates a historical or process relationship between two elements that represent the same concept without specific rules for deriving one from the other.

transient object

An object that exists only during the execution of the process or thread that created it.

transition

A relationship between two states indicating that an object in the first state will perform certain specified actions and enter the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to fire.

type

type A stereotyped class that specifies a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects. A type may not contain any methods, maintain its own thread of control, or be nested. However, it may have attributes and associations. Although an object may have at most one implementation class, it may conform to multiple different types.

See also: implementation class

Contrast: interface

type expression

An expression that evaluates to a reference to one or more types.

16.20 U (Glossary)

~U~



UML

The Unified Modeling Language, a notation and specification for modeling software systems in an Object-Oriented manner. You can read more about UML at the OMG home page or at our UML Tutorial

UML diagrams

UML diagrams are used to model different aspects of the system under development. They include various elements and connectors, all of which have their own meanings and purposes. UML 2.0 includes 13 diagrams: Use Case diagram, Activity diagram, State Machine diagram, Timing diagram, Sequence diagram, Interaction Overview diagram, Communication diagram, Package diagram, Class diagram, Object diagram, Composite Structure diagram, Component diagram and Deployment diagram.

UML toolbox

The main toolbar running down the centre of EA from which you can select model elements to insert into diagrams. This is also known as the Toolbox and the Object Toolbar.

uninterpreted

A placeholder for a type or types whose implementation is not specified by the UML. Every uninterpreted value has a corresponding string representation.

See also: any [CORBA]

usage

A dependency in which one element (the client) requires the presence of another element (the supplier) for its correct functioning or implementation.

use

A Use link indicates that one element requires another to perform some interaction. The Usage relationship does not specify how the target supplier is used, other than that the source client uses it in definition or implementation.

use case [class]

A Use Case is a UML model element that describes how a user of the proposed system will interact with the system to perform a discrete unit of work. It describes and signifies a single interaction over time that has meaning for the end user (person, machine or other system), and is required to leave the system in a complete state: either the interaction completed or was rolled back to the initial state.

See also: use case instance

use case diagram

A Use Case diagram captures Use Cases and actor interactions. It describes the functional requirements of the system, the manner that outside things (actors) interact at the system boundary, and the response of the system.

use case estimation

The technique of estimating project size and complexity based on the number of use cases and their difficulty.

use case instance

The performance of a sequence of actions being specified in a use case. An instance of a use case.

See also: use case class

use case model

A model that describes a system's functional requirements in terms of use cases.

utility

A stereotype that groups global variables and procedures in the form of a class declaration. The utility attributes and operations become global variables and global procedures, respectively. A utility is not a fundamental modeling construct, but a programming convenience.

16.21 V (Glossary)

~V~

value

An element of a type domain.

**value lifeline**

The Value lifeline shows the lifeline's state across the diagram, within parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

vertex

A source or a target for a transition in a state machine. A vertex can be either a state or a pseudo-state.

See also: state, pseudo-state

view

A projection of a model, which is seen from a given perspective or vantage point and omits entities that are not relevant to this perspective.

view element

A view element is a textual and/or graphical projection of a collection of model elements.

view projection

A projection of model elements onto view elements. A view projection provides a location and a style for each view element.

visibility

An enumeration whose value (public, protected, package or private) denotes how the model element to which it refers may be seen outside its enclosing namespace.

Visual Basic

A rapid application development programming language. Windows only scripting language based on COM.

Part

17

17 Acknowledgements

Some parts of this application include code originally written by various authors and modified for use in Enterprise Architect.

Marquet Mike

Print listview contents

mike.marquet@altavista.net

Davide Pizzolato

CXImage Library

© 7-Aug-2001

ing.davide.pizzolato@libero.it

Also, many thanks to all those who have made suggestions, reported bugs, offered feedback and helped with the beta testing of Enterprise Architect. Their help has been invaluable.

Index

- . -

.NET - Setting up a Debug session. 1008

- A -

About EA 38

Abstract 274

Abstract XSD models 1048

Acceptance Testing 887

Acknowledgements 1356

Action - Element 296

Action - Local Pre/Post Conditions 300

Action Expansion Node 298

Action Notation 296

Action Pin 299

ActionScript Options 949

Active Classes 311

Active state configuration 357

Activity - Element 301

Activity Diagram 248

Activity Diagrams - Object Flows 421

Activity Final - Element 330

Activity Group - UML Toolbox 172

Activity Notation 303

Activity Parameter Nodes 303

Activity Partition 306

Actor - Element 307

Add a Diagram 187

Add a Pattern to Diagram 618

Add a Task 901

Add a UML Pattern to Diagram 618

Add Additional Views 750

Add an Element 199

Add an Instance Variable 346

Add Code Modules in MDG Technology Wizard 585

Add Diagram Properties Note 444

Add Enumeration tags to Stereotypes 601

Add Expansion Region 329

Add Images in MDG Technology Wizard 587

Add Ins - Automation and Scripting 1277

Add Interruptible Activity Region 341

Add Key 34

Add License Key 34

Add MDG Technologies to UML Toolbox 591

Add MS Word Table of Contents 1145

Add MS Word Table of Figures 1146

Add Note to Connection 538

Add Note to Link 538

Add or Delete Line Points 542

Add Packages to Your Document Object 1161

Add Pattern in MDG Technology Wizard 583

Add Profile in MDG Technology Wizard 583

Add Properties Note 444

Add Tagged Value types in MDG Technology Wizard 584

Add Tagged Values 520

Add to Project Clipboard 62

Add UML Profile Connector to Diagram 609

Add-In Manager 240

Add-In Tasks - Automation and Scripting 1278

Adding a Port to an Element 351

Adding Content 1107

Adding New Code Sections to Existing Features 987

Adding New Features and Elements 987

Adding New Stereotyped Templates 984

Adding Packages 185

Add-Ins Events - Automation and Scripting 1284

Advanced Settings 511

Advanced Tag Management 518

Aggregate - Connection 390

Aggregation Link Form 391

All Permissions 827

Allow Duplicate Tagged Values 157

Allow Duplicate Tags 157

Alternate Image 452

Analysis Diagram 284

Analysis Group - UML Toolbox 166

Analysis Stereotypes 372

App - Automation and Scripting - Automation Interface 1177

Appearance 482

Appearance - Element Context Menu 466

Appearance Menu Section 537

Application Workspace 41

Apply a Filter 1131

Apply a User Lock 836

Applying a Stereotype 399

Applying Stereotypes 559

Argument 501

- Arranging Connections 539
- Artifact -Element 308
- Assembly - Connection 391
- Assign People to Defects or Changes 899
- Assigning Information to tagged Values 156
- Assigning Tagged Values to an Item 154
- Associate - Connection 392
- Associated Files 516
- Association - Connection 393
- Association - Diagonal Representation 393
- Association - Link a New Class to an Existing Association 394
- Association Class 381, 393
- Association Details 552
- Association Properties 552
- Association Type - Change 540
- Attach a Note to a Link 538
- Attribute Constraints 490
- Attribute Detail 490
- Attribute Tagged Values 491
- Attributes 487
- Attributes and Operations 487
- Attributes Main Page 489
- Authors 853
- Auto Routing Connector Style 542
- Autohide Windows 59
- Autolayout 188
- Automation - Properties 1227
- Automation - Property 1227
- Automation and Scripting 1167
- Automation and Scripting - Add Ins 1277
- Automation and Scripting - Add-In Tasks 1278
- Automation and Scripting - Add-Ins - Overview 1278
- Automation and Scripting - Add-Ins Events 1284
- Automation and Scripting - Add-Ins Events - EA_Connect 1284
- Automation and Scripting - Add-Ins Events - EA_Disconnect 1285
- Automation and Scripting - Add-Ins Events - EA_GetMenuItems 1285
- Automation and Scripting - Add-Ins Events - EA_GetMenuState 1286
- Automation and Scripting - Add-Ins Events - EA_MenuClick 1287
- Automation and Scripting - Add-Ins Events - EA_OnOutputItemClicked 1287
- Automation and Scripting - Add-Ins Events - EA_OnOutputItemDoubleClicked 1288
- Automation and Scripting - Add-Ins Events - EA_ShowHelp 1289
- Automation and Scripting - Automation Interface 1167
- Automation and Scripting - Automation Interface - App 1177
- Automation and Scripting - Automation Interface - Available Resources 1173
- Automation and Scripting - Automation Interface - Call from EA 1172
- Automation and Scripting - Automation Interface - Connecting to the Interface 1168
- Automation and Scripting - Automation Interface - Connector 1228
- Automation and Scripting - Automation Interface - Connector - Connector 1229
- Automation and Scripting - Automation Interface - Connector - Connector Constraints 1230
- Automation and Scripting - Automation Interface - Connector - Connector End 1231
- Automation and Scripting - Automation Interface - Connector - Connector Tag 1232
- Automation and Scripting - Automation Interface - Connector - Role Tag 1233
- Automation and Scripting - Automation Interface - ConstLayoutStyles Enum 1178
- Automation and Scripting - Automation Interface - Diagram 1234
- Automation and Scripting - Automation Interface - Diagram - Diagram Links 1236
- Automation and Scripting - Automation Interface - Diagram - Diagram Objects 1237
- Automation and Scripting - Automation Interface - Diagram- Diagram 1235
- Automation and Scripting - Automation Interface - Element 1204
- Automation and Scripting - Automation Interface - Element - Constraint 1206
- Automation and Scripting - Automation Interface - Element - Effort 1206
- Automation and Scripting - Automation Interface - Element - Element 1207
- Automation and Scripting - Automation Interface - Element - File 1211
- Automation and Scripting - Automation Interface - Element - Issue 1212
- Automation and Scripting - Automation Interface - Element - Metric 1213
- Automation and Scripting - Automation Interface - Element - Requirement 1213
- Automation and Scripting - Automation Interface - Element - Resource 1214

- Automation and Scripting - Automation Interface - Element - Risk 1215
- Automation and Scripting - Automation Interface - Element - Scenario 1216
- Automation and Scripting - Automation Interface - Element - Tagged Value 1216
- Automation and Scripting - Automation Interface - Element - Test 1217
- Automation and Scripting - Automation Interface - Element Features 1218
- Automation and Scripting - Automation Interface - Element FeaturesAttribute 1219
- Automation and Scripting - Automation Interface - Element FeaturesAttribute Constraint 1220
- Automation and Scripting - Automation Interface - Element FeaturesAttribute Tag 1221
- Automation and Scripting - Automation Interface - Element FeaturesCustom Properties 1227
- Automation and Scripting - Automation Interface - Element FeaturesEmbedded Elements 1225
- Automation and Scripting - Automation Interface - Element FeaturesMethod 1221
- Automation and Scripting - Automation Interface - Element FeaturesMethod Constraint 1223
- Automation and Scripting - Automation Interface - Element FeaturesMethod Tag 1223
- Automation and Scripting - Automation Interface - Element FeaturesParameter 1224
- Automation and Scripting - Automation Interface - Element FeaturesPartitions 1225
- Automation and Scripting - Automation Interface - Element FeaturesTransitions 1226
- Automation and Scripting - Automation Interface - Enumerations 1178
- Automation and Scripting - Automation Interface - EnumRelationSetType Enum 1178
- Automation and Scripting - Automation Interface - Examples and Tips 1171
- Automation and Scripting - Automation Interface - MDG Menus Enumeration 1178
- Automation and Scripting - Automation Interface - Model and Package 1174
- Automation and Scripting - Automation Interface - Object TypeEnum 1179
- Automation and Scripting - Automation Interface - Project Interface 1239
- Automation and Scripting - Automation Interface - Project Interface - Automation Interfaces 1259
- Automation and Scripting - Automation Interface - Project Interface - Enumerating 1262
- Automation and Scripting - Automation Interface - Project Interface - Enumerating Diagrams 1262
- Automation and Scripting - Automation Interface - Project Interface - Enumerating Elements 1265
- Automation and Scripting - Automation Interface - Project Interface - Enumerating Links 1274
- Automation and Scripting - Automation Interface - Project Interface - Enumerating Views 1260
- Automation and Scripting - Automation Interface - Project Interface - Get Diagram Image 1277
- Automation and Scripting - Automation Interface - Project Interface - Get Diagrams 1263
- Automation and Scripting - Automation Interface - Project Interface - Get Element 1266
- Automation and Scripting - Automation Interface - Project Interface - Get Link 1275
- Automation and Scripting - Automation Interface - Project Interface - Opening a Project 1260
- Automation and Scripting - Automation Interface - Project Interface - Project 1240
- Automation and Scripting - Automation Interface - Project Interface - Read Only Interface 1259
- Automation and Scripting - Automation Interface - Reference 1173
- Automation and Scripting - Automation Interface - Repository 1180, 1182
- Automation and Scripting - Automation Interface - Repository - Author 1190
- Automation and Scripting - Automation Interface - Repository - Client 1191
- Automation and Scripting - Automation Interface - Repository - Collection 1191
- Automation and Scripting - Automation Interface - Repository - Datatype 1193
- Automation and Scripting - Automation Interface - Repository - EventProperties 1194
- Automation and Scripting - Automation Interface - Repository - EventProperty 1194
- Automation and Scripting - Automation Interface - Repository - Package 1194
- Automation and Scripting - Automation Interface - Repository - Project Issues 1198
- Automation and Scripting - Automation Interface - Repository - Project Resource 1199
- Automation and Scripting - Automation Interface - Repository - Property Type 1200
- Automation and Scripting - Automation Interface - Repository - Reference 1201
- Automation and Scripting - Automation Interface - Repository - Stereotype 1201
- Automation and Scripting - Automation Interface - Repository - Task 1202
- Automation and Scripting - Automation Interface - Repository - Term 1203

- Automation and Scripting - Automation Interface - Set Up VB 1169
- Automation and Scripting - Automation Interface - The Project Interface (XML) 1258
- Automation and Scripting - Automation Interface - Using the Automation Interface 1168
- Automation and Scripting - Automation Interface - XMI Type Enum 1179
- Automation and Scripting - Automation Interface -Code Example - Add a Connector 1249
- Automation and Scripting - Automation Interface -Code Example - Add and Manage Diagrams 1250
- Automation and Scripting - Automation Interface -Code Example - Add and Manage Elements 1249
- Automation and Scripting - Automation Interface -Code Example - Add and Manage Packages 1248
- Automation and Scripting - Automation Interface -Code Example - Adding and Deleting Attributes and Methods 1251
- Automation and Scripting - Automation Interface -Code Example - Element Extras 1252
- Automation and Scripting - Automation Interface -Code Example - Iterate Through an EAP File 1248
- Automation and Scripting - Automation Interface -Code Example - Open the Repository 1247
- Automation and Scripting - Automation Interface -Code Example - RepositoryExtras 1254
- Automation and Scripting - Automation Interface -Code Example - Stereotypes 1256
- Automation and Scripting - Automation Interface -Code Example - Working with Attributes 1256
- Automation and Scripting - Automation Interface -Code Example - Working with Methods 1257
- Automation and Scripting - Automation Interface -Code Examples 1247
- Automation and Scripting - Broadcast Events 1290
- Automation and Scripting - Creating Add-In 1279
- Automation and Scripting - Creating Add-In - Defining Menu Items 1279
- Automation and Scripting - Creating Add-In - Deploying 1280
- Automation and Scripting - Creating Add-In - Tricks and Traps 1281
- Automation and Scripting - Creating a Custom View 1309
- Automation and Scripting - Custom View 1308
- Automation and Scripting - EA_FileOpen 1290
- Automation and Scripting - EA_OnDeleteTechnology 1304
- Automation and Scripting - EA_OnImportTechnology 1304
- Automation and Scripting - EA_OnPostNewConnector 1300
- Automation and Scripting - EA_OnPreNewConnector 1296
- Automation and Scripting - EA_OnPreNewElement 1295
- Automation and Scripting - EA_PostNewElement 1300
- Automation and Scripting - MDG Add-Ins 1309
- Automation and Scripting - MDG Add-Ins - MDG Events 1310
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG Add-In 1316
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG BuildProject 1310
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_Connect 1311
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_Disconnect 1312
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_GetConnectedPackages 1313
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_GetProperty 1313
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_Merge 1314
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_PostGenerate 1317
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_PostMerge 1317
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_PreGenerate 1318
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_PreMerge 1319
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_PreReverse 1320
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_RunExe 1320
- Automation and Scripting - MDG Add-Ins - MDG Events - MDG_View 1321
- Automation Interface 1167, 1194
- Automation Interface - Repository 1198
- Automation Interfaces - Project Interface 1259
- Autosize Elements 445
- Available Resources - Automation and Scripting - Automation Interface 1173

- B -

- Base Project 728
- Baselines 845
- Baselines - Creating 847
- Baselines - Managing 846
- Baselines and Differences 845

- Basic Element 295
 - Basic UML Elements 295
 - Batch XMI Export 774
 - Batch XMI Import 775
 - Behavioral Diagrams 246
 - Bend Line at Cursor 542
 - Bitmap Images in Diagrams 452
 - Bookmark Selected 62
 - Bookmarks 918
 - Boundary 373
 - Boundary - Create 373
 - Boundary - Element 365
 - Boundary Object Settings 523
 - Boundary Properties 523
 - Breakpoints 1023
 - States 1023
 - Broadcast Events - Automation and Scripting 1290
 - Build and Run 998
 - Build and Run - Build Scripts 1001
 - Build and Run - Creating a New Script 1003, 1004
 - Build and Run - Debugging with EA 1015
 - Build and Run - Deploy Scripts 1005
 - Build and Run - Managing Compile Scripts 1000
 - Build and Run - Recording a Debug Session 1025
 - Build and Run - Sequence Diagrams 1025
 - Build and Run - Source Code Configuration 999
 - Build and Run - xUnit Testing 1034
 - Build and Run Sub Menu 129
 - Build Scripts - Build and Run 1001
 - Business Analysis 48
 - Business Analyst - Project Roles and EA 13
 - Business Model 48
 - Business Model Pattern 48
 - Business Modeling 48
 - Business Modeling Stereotypes 1 374
 - Business Modelling 637
 - Business Process 48
 - Business Process Model 48
 - Business Process Model Pattern 48
 - Business Processes 637
- C -**
- C Conventions 988
 - C# Options 950
 - C# Transformation 648
 - C++ Options 951
 - Call 412
 - Call from EA - Automation and Scripting - Automation Interface 1172
 - Change Aggregation Link Form 391
 - Change Connector Type 540
 - Change Diagram Type 440
 - Change the Source or Target Object 540
 - Changes 895, 898
 - Changes and Defects 896
 - Check Constraint 1083
 - Checking in a Package 811
 - Child Sections 1109
 - Choice - Element 308
 - Class 274
 - Class - Active Classes 311
 - Class - Element 309
 - Class - Parameterized Classes (Templates) 311
 - Class Diagram 274
 - Class Group - UML Toolbox 168
 - Class Model 51
 - Class Model Pattern 51
 - Class Modeling 51
 - Class Pattern 51
 - Classes - Parameterized (Templates) 311
 - Clear All Bookmarks 62
 - Clear Project Clipboard 62
 - Clear Selection 62
 - Clients 854
 - Close Project 61
 - Code Engineering 920
 - Code Engineering - Build and Run 998
 - Code Engineering - Build and Run - BuildScripts 1001
 - Code Engineering - Build and Run - Creating a New Script 1003, 1004
 - Code Engineering - Build and Run - Debugging with EA 1015
 - Code Engineering - Build and Run - Deploy Scripts 1005
 - Code Engineering - Build and Run - Managing Compile Scripts 1000
 - Code Engineering - Build and Run - Recording a Debug Session 1025
 - Code Engineering - Build and Run - Setting up a Debug session 1005
 - Code Engineering - Build and Run - Source Code Configuration 999
 - Code Engineering - Build and Run - xUnit Testing 1034
 - Code Engineering - Code - Reverse Engineer 920
 - Code Engineering - Eclipse 926

- Code Engineering - Element Context Menu 465
- Code Engineering - Generate Source Code 929
- Code Engineering - Generate Source Code - Code Generation Dialog 932
- Code Engineering - Generate Source Code - Generate Package Dialog 935
- Code Engineering - Generate Source Code - Generate a Group of Classes 933
- Code Engineering - Generate Source Code - Generate a Package 933
- Code Engineering - Generate Source Code - Generate a Single Class 930
- Code Engineering - Generate Source Code - Generate Code 930
- Code Engineering - Generate Source Code - Generate Package Source Code 933
- Code Engineering - Generate Source Code - Namespaces 936
- Code Engineering - Generate Source Code - Package Contents - Update 935
- Code Engineering - Generate Source Code - Synchronize Package Tree 935
- Code Engineering - Generate Source Code - Update Package Contents 935
- Code Engineering - Import Source Code 926
- Code Engineering - MDG Link 926
- Code Engineering - MDG Link Code Engineering 926
- Code Engineering - Reverse Engineer Source Code 920
- Code Engineering - Reverse Engineer Source Code - Import a Directory Structure 921
- Code Engineering - Reverse Engineer Source Code - Import Action Scripts 922
- Code Engineering - Reverse Engineer Source Code - Import Binary Module 925
- Code Engineering - Reverse Engineer Source Code - Import C# 923
- Code Engineering - Reverse Engineer Source Code - Import C++ 922
- Code Engineering - Reverse Engineer Source Code - Import Delphi 923
- Code Engineering - Reverse Engineer Source Code - Import Java 923
- Code Engineering - Reverse Engineer Source Code - Import PHP 924
- Code Engineering - Reverse Engineer Source Code - Import Python 924
- Code Engineering - Reverse Engineer Source Code - Import VB.Net 924
- Code Engineering - Reverse Engineer Source Code - Import Visual Basic 924
- Code Engineering - Reverse Engineer Source Code - Reverse Engineer a Directory Structure 921
- Code Engineering - Synchronize Model and Code 928
- Code Engineering - Visual Studio.NET 926
- Code Engineering - XML Schema (XSD) 1039
- Code Engineering - XML Schema (XSD) - Abstract XSD models 1048
- Code Engineering - XML Schema (XSD) - Default UML to XSD mappings 1050
- Code Engineering - XML Schema (XSD) - Generate XSD 1051
- Code Engineering - XML Schema (XSD) - Import XSD 1051
- Code Engineering - XML Schema (XSD) - Model XSD 1039
- Code Engineering - XML Schema (XSD) - UML Profile for XSD 1041
- Code Engineering - XML Schema (XSD) - XSD Datatype Packages 1048
- Code Engineering Settings 937
- Code Engineering Settings - Code Page for Source Editing 942
- Code Engineering Settings - General Code Options 937
- Code Engineering Settings - Import Component Types 939
- Code Engineering Settings - Options - Code Generation - Attributes 941
- Code Engineering Settings - Options - Code Generation Constructor/Destructor 940
- Code Engineering Settings - Options - Default Editors 939
- Code Engineering Settings Options 938
- Code Engineering Sub-Menu 129
- Code Generation 930
- Code Generation - Toolbar 112
- Code Generation Options 236
- Code Template Editor 982
- Code Templates 960
- Code Templates - Base Templates 961
- Code Templates - Custom Templates 964
- Code Templates - Execution 965
- Code Templates - Literal Text 966
- Code Templates - Macros 966
- Code Templates - Overview 960
- Code Templates - Variables 980
- Code Templates Syntax 965
- CodeTemplate Framework 960
- Collaboration - Element 313
- Collaboration Diagram 266

- Collaboration Diagrams in Color 268
- Collaboration Group - UML Toolbox 170
- Collaboration Messages 408
- Collaboration Occurrence - Element 314
- Collisions 839
- Color Coded External Requirements 623
- Combined Fragment - Create a Combined Fragment 318
- Combined Fragment - Element 316
- Combined Fragment - Interaction Operators 318
- Common Actions - Element Context Menu 466
- Common Actions Menu Section 536
- Common Connection Tasks 537
- Common Element Tasks - 468
- Common Element Tasks - Align Elements 477
- Common Element Tasks - Changing Object Type 475
- Common Element Tasks - Configure Default Appearance 482
- Common Element Tasks - Copying Attributes and Operations Between Elements 472
- Common Element Tasks - Creating Elements 468
- Common Element Tasks - Creating Notes and Text 480
- Common Element Tasks - Customize Visible Elements 479
- Common Element Tasks - Deleting Elements 478
- Common Element Tasks - Get/Set Project Custom Colors 484
- Common Element Tasks - Moving Attributes and Operations between Elements 473
- Common Element Tasks - Moving Elements in Diagrams 476
- Common Element Tasks - New Elements 468
- Common Element Tasks - Set Element Parent 469
- Common Element Tasks - Set Up References 471
- Common Element Tasks - Show Element Usage 470
- Common Element Tasks - Size Elements 477
- Common Usage 410
- Communication Colors 233
- Communication Diagram 266
- Communication Diagrams in Color 268
- Communication Message 407
- Communication Message Creation 407
- Communication Message Properties 407, 408
- Compact a Project 747
- Compare Data 734
- Comparing Models 734
- Compartments 524
- Compile Scripts - Managing - Build and Run 1000
- Component - Element 321
- Component Diagram 281
- Component Group - UML Toolbox 173
- Component Model 52
- Component Model Pattern 52
- Component Modeling 52
- Component Pattern 52
- Compose - Connection 396
- Composite Elements 375
- Composite Group - UML Toolbox 169
- Composite state 357
- Composite State Regions 253
- Composite Structure Diagram 278
- Composition 628
- Configure Controlled Packages with XMI 770
- Configure Local Options 223
- Configure Packages 770
- Connect to a MSDE Server Data Repository 714
- Connect to a MySQL Data Repository 702
- Connect to a Oracle9i Data Repository 708
- Connect to a PostgreSQL Data Repository 710
- Connect to a SQL Server Data Repository 705
- Connect to an Adaptive Server Anywhere Data Repository 712
- Connecting Elements 200, 541
- Connecting Objects 541
- Connecting to a Data Repository 702
- Connecting to the Interface - Automation and Scripting - Automation Interface 1168
- Connection - Add a Note 538
- Connection Constraints 553
- Connection Context Menu 535
- Connection Details 552
- Connection Properties 552
- Connections 388
- Connections - Aggregate 390
- Connections - Assembly 391
- Connections - Associate 392
- Connections - Association 393
- Connections - Compose 396
- Connections - Connector 396
- Connections - Control Flow 397
- Connections - Delegate 398
- Connections - Dependency 399
- Connections - Deployment 400
- Connections - Extend 401
- Connections - Generalize 402

- Connections - Include 403
- Connections - Interrupt Flow 405
- Connections - Manifest 406
- Connections - Message 406
- Connections - Nesting 420
- Connections - Object Flow 420
- Connections - Occurrence 422
- Connections - Package Import 423
- Connections - Package Merge 423
- Connections - Realize 424
- Connections - Represents 425
- Connections - Role Binding 425
- Connections - Trace 426
- Connections - Transition 427
- Connections - Use 429
- Connector - Tagged Value 599
- Connector Styles 542
- Connector Type - Change 540
- Connector Visibility 546, 549
- ConstLayoutStyles Enum - Automation and Scripting
- Automation Interface 1178
- Constraint Compartment 525
- Constraint Status Types 860
- Constraint Types 860
- Constraints 513, 600
- Contents Sub-Menu 131
- Context Element Highlight 187
- Continuation 361
- Continuation - Element 360
- Control 376
- Control Element - Create 376
- Control Flow - Connection 397
- Control Macros 977
- Controlled Package 772
- Controlled Packages Menu -XMI 769
- Controlled Packages with XMI 769
- Convert Linked Element to Local Copy 439
- Copy a Base Project 728
- Copy Diagram Element 438
- Copy Diagram Image to Clipboard 440
- Copy Image to Clipboard 440
- Copy Image to Disk 440
- Copy RTF Bookmark to Clipboard 128, 133
- Copyright Notice 5
- Correcting Words 850
- Corrupt .EAP file 748
- Create a Boundary 373
- Create a Control Element 376
- Create a Document Object 1161
- Create a Model 688
- Create a New Adaptive Server Anywhere Repository
724
- Create a New MSDE Server Repository 726
- Create a New MySQL Repository 716
- Create a New Oracle9i Server Repository 721
- Create a New PostgreSQL Repository 721
- Create a New SQL Server Repository 719
- Create a Rich Text Document 1099
- Create a Timing Message 416
- Create a Virtual Report 1161
- Create an Entity 377
- Create and Open Model Files Discussion 727
- Create Link 543
- Create MDG Technologies 579
- Create Messages 407
- Create UML Patterns 616
- Creating a Custom View - Automation and Scripting
1309
- Creating a New Script - Build and Run 1003, 1004
- Creating a Repository 716
- Creating Add-Ins 1279
- Creating an HTML Report 1155
- Creating Connections Between Elements 163
- Creating Documents 1098
- Creating Elements 468
- Creating New Elements Using the Quick Linker
162
- Creating Notes 480
- Creating Patterns 616
- Creating Profiles 594
- Creating Properties 492
- Creating Replicas 841
- Creating Requirements 622
- Creating Templates For Custom Languages 985
- Creating Text 480
- Creating UML Profiles 594
- Cross References 471
- CSV Export 779
- CSV Import 780
- CSV Import and Export 777
- CSV Specifications 777
- CTF 960
- Current Connector Toolbar 117
- Current Element Toolbar 116
- Custom Connector Style 542
- Custom Diagram 285, 286, 287, 288
- Custom Group - UML Toolbox 175

Custom Language Settings 1102
 Custom Layouts 236
 Custom Properties Dialog 461
 Custom Stereotypes 560
 Custom Tagged Values 153
 Custom Toolbars 79
 Custom Tools 81
 Custom Views - Automation and Scripting 1308
 Customize Commands 78
 Customize Keyboard Shortcuts 85
 Customize Menu 87
 Customize Options 88
 Customize Toolbar 119
 Customize Window 77
 CVS 792

- D -

Data and Model Integrity 730
 Data and Model Integrity - Checking Data Integrity 730
 Data and Model Integrity - Running SQL Patches 732
 Data Compare 734
 Data Integrity 730
 Data Model 52
 Data Model Pattern 52
 Data Modeling 52, 1064
 Data Modeling - Foreign Keys 1075
 Data Modeling - Create a Table 1065
 Data Modeling - Create Columns 1071
 Data Modeling - Data Model Diagram 1065
 Data Modeling - Data Type Conversion Procedures 1088
 Data Modeling - DBMS Conversion Procedure Package 1089
 Data Modeling - DBMS Datatypes 1091
 Data Modeling - DDL 1085
 Data Modeling - Generate DDL 1085
 Data Modeling - Generate DDL for a Package 1087
 Data Modeling - Index, Trigger, Check Constraint 1083
 Data Modeling - ODBC 1085
 Data Modeling - Primary Key 1073
 Data Modeling - Primary Key - Primary Key Extended Properties 1074
 Data Modeling - Set MySQL Table Type 1069
 Data Modeling - Set Table Owner 1068
 Data Modeling - Set Table Properties 1066
 Data Modeling - SQL 1085
 Data Modeling - Stored Procedures 1079
 Data Pattern 52
 Data Transfer 732
 Data Transfer - Comparing Models 734
 Data Transfer - Copy Packages from One Project to Another 735
 Data Transfer - Perform a Data Transfer 733
 Data Transfer - Why Compare Models? 734
 Data Types 865
 Database Administrator 25
 Database Model 52
 Database Model Pattern 52
 Database Modeling 52
 Database Pattern 52
 Database Schema 290
 Datastore - Element 321
 Datatype - Data Type Conversion Procedures 1088
 Datatypes - DBMS 1091
 DBMS - Data Type Conversion Procedures 1088
 DBMS Conversion Procedure 1088
 DDL - Data Modeling 1085
 DDL Transformation 650
 Debug History 1022
 Debug Toolbar 1017
 Debug Workbench 1028
 Debugger 1016
 Debugger to Sequence 1025
 Debugging - Invoking Methods 1033
 Debugging - Local Variables 1020
 Debugging - Output 1021
 Debugging - Stack 1021
 Debugging - Variables 1029
 Debugging Another Process 1013
 Debugging ASP .NET 1009
 Debugging Assemblies 1012
 Debugging Java Web Servers 1008
 Debugging with EA 1015
 Decision Element 322
 Default Diagram 435
 Default Hours 874
 Default Model Diagram 435
 Default UML to XSD mappings 1050
 Defaults and User Settings 223
 Defects 895
 Defects (Issues) 897
 Define a Run time Variable 346

- Define Stereotype Constraints 600
- Defining Menu Items - Automation and Scripting - Creating Add-In 1279
- Delegate - Connection 398
- Delete a Package 186
- Delete a Package From Document Object 1163
- Delete a Task 901
- Delete an Instance Variable 347
- Delete Selected Element(s) 62
- Delete Views 752
- Deleting Connections 545
- Deleting Diagrams 437
- Delphi Options 952
- Delphi Properties 953
- Denote Lifecycle of an Element 258
- Dependency - Connection 399
- Dependency Report 629, 1151
- Deploy Scripts - Build and Run 1005
- Deploying - Automation and Scripting - Creating Add-In 1280
- Deployment - Connection 400
- Deployment and Rollout 22
- Deployment Diagram 282
- Deployment Group - UML Toolbox 174
- Deployment Model 53
- Deployment Model Pattern 53
- Deployment Modeling 53
- Deployment Pattern 53
- Deployment Spec - Element 323
- Design Master 730, 839
- Design Masters 841
- Destination Role 556
- Developer - Project Roles and EA 17
- Diagram Behavior 228
- Diagram Context Menu 434
- Diagram Context Menu - Project Browser 133
- Diagram Gate - Element 324
- Diagram Menu 70
- Diagram Navigation 439
- Diagram Navigation Hotkeys 439
- Diagram Notes 444
- Diagram Only RTF Report 1137
- Diagram Properties 437
- Diagram Properties Note 444
- Diagram Settings 227
- Diagram Tabs Toolbar 120
- Diagram Tasks 187, 436
- Diagram Toolbar 115
- Diagram View 94
- Diagrams - Analysis 284
- Diagrams - Collaboration 266
- Diagrams - Communication 266
- Diagrams - Component 281
- Diagrams - Composite Structure 278
- Diagrams - Custom 285, 286, 287, 288
- Diagrams - Database Schema 290
- Diagrams - Deployment 282
- Diagrams - Interaction Overview 269
- Diagrams - Robustnes 290
- Diagrams - Sequence 256
- Diagrams - Structural Diagrams 271
- Direct Connector Style 542
- Disconnect Controlled Package 772
- Discussion Forum 676
- Discussion Forum - Adding a New Category 677
- Discussion Forum - Adding a New Post 679
- Discussion Forum - Adding a New Topic 678
- Discussion Forum - Adding Element Links 683
- Discussion Forum - Copy Path to Clipboard 684
- Discussion Forum - Deleting a Category 681
- Discussion Forum - Editing a Post 680
- Discussion Forum - Message Dialog 683
- Discussion Forum - Posting and Editing Messages 677
- Discussion Forum - Replying to a Post 679
- Displaying Inherited Attributes 494
- Displaying Inherited Operations 507
- Displaying Tagged Values 443
- Displaying Tags 443
- Distributed Development 819
- Dockable Windows 136
- Dockable Windows - Arranging Windows and Menus 56
- Dockable Windows 2005 Style 57
- Document a Single Element 1129
- Document Artifact - Element 325
- Document Object 1161
- Document Object - Adding Packages 1161
- Document Object - Delete a Package 1163
- Document Object - Generate 1164
- Document Object - Rearrange Package Order 1162
- Document Options 457
- Documentation - HTML 1155
- Documentation - RTF 1098
- Documentation and Web 1158
- Documentation Sub-Menu 128

Documenting Projects 1098
 Documents - HTML 1098
 Documents - RTF 1098
 Domain Model 50
 Domain Model Pattern 50
 Domain Modeling 50
 Domain Pattern 50
 Drag 445
 Drag a Package onto a Diagram 186
 Drop Elements from the Project Browser 445
 Drop Objects from Tree 445
 Duplicate a Diagram 442

- E -

EA Project Files - Open a Project 687
 EA_Connect - Automation and Scripting - Add-Ins Events 1284
 EA_Disconnect - Automation and Scripting - Add-Ins Events 1285
 EA_FileClose - Automation and Scripting - Broadcast Events 1291
 EA_FileOpen - Automation and Scripting - Broadcast Events 1290
 EA_GetMenuItems - Automation and Scripting - Add-Ins Events 1285
 EA_GetMenuState - Automation and Scripting - Add-Ins Events 1286
 EA_MenuClick - Automation and Scripting - Add-Ins Events 1287
 EA_OnContextItemChanged - Automation and Scripting - Broadcast Events 1305
 EA_OnContextItemDoubleClicked - Automation and Scripting - Broadcast Events 1306
 EA_OnDeleteTechnology - Automation and Scripting - Broadcast Events 1304
 EA_OnImportTechnology - Automation and Scripting - Broadcast Events 1304
 EA_OnNotifyContextItemModified - Automation and Scripting - Broadcast Events 1307
 EA_OnOutputItemClicked - Automation and Scripting - Add-Ins Events 1287
 EA_OnOutputItemDoubleClicked - Automation and Scripting - Add-Ins Events 1288
 EA_OnPostNewConnector - Automation and Scripting - Broadcast Events 1300
 EA_OnPreDeleteConnector - Automation and Scripting - Broadcast Events 1292
 EA_OnPreDeleteDiagram - Automation and Scripting - Broadcast Events 1293
 EA_OnPreDeleteElement - Automation and Scripting - Broadcast Events 1291
 EA_OnPreDeletePackage - Automation and Scripting - Broadcast Events 1294
 EA_OnPreNewConnector - Automation and Scripting - Broadcast Events 1296
 EA_OnPreNewElement - Automation and Scripting - Broadcast Events 1295
 EA_PostNewElement - Automation and Scripting - Broadcast Events 1300
 EA_ShowHelp - Automation and Scripting - Add-Ins Events 1289
 ECF Dialog 871
 Edit Menu 62
 Editor Commands 1111
 Effort and Risk Management 876
 Effort Types 880
 EJB Transformations 652
 Element Appearance 482
 Element Browser 144
 Element Context Menu 460
 Element Context Menu - Appearance 466
 Element Context Menu - Code Engineering 465
 Element Context Menu - Common Actions 466
 Element Context Menu - Embedding and Features 463
 Element Context Menu - Multiple Selection 467
 Element Context Menu - Project Browser 131
 Element Context Menu - Properties 461
 Element Context Menu - Type 462
 Element Lifecycle 258
 Element Menu 72
 Element Properties 509, 899
 Element Relationship Matrix 630
 Element Tasks 199
 Element Tasks - Add an Object 199
 Element Tasks - Auto Counters 201
 Element Tasks - Auto Element Naming 201
 Element Tasks - Connecting Objects 200
 Element Tasks - Default Element Template 202
 Element Tasks - Default Element Templates 202
 Element Tasks - Element Template 202
 Element Tasks - Element Template Package 202
 Element Tasks - Moving Objects and Packages 200
 Element Tasks - Quick Links 161
 Element Tasks - Set Element Template Package 202
 Element Tasks - Template - default element 202
 Element Template 202

| | | | |
|--|-----|--|------|
| Element Toolbar | 114 | Elements - Region - Interruptible Activity | 340 |
| Element Visibility | 231 | Elements - Send | 355 |
| Elements - | 291 | Elements - State | 356 |
| Elements - Action | 296 | Elements - State Invariant | 360 |
| Elements - Activity | 301 | Elements - State Lifeline | 359 |
| Elements - Activity Final | 330 | Elements - State/Continuation | 360 |
| Elements - Actor | 307 | Elements - SubActivity | 363 |
| Elements - Artifact | 308 | Elements - Sub-Activity | 301 |
| Elements - Basic Elements | 295 | Elements - SubMachine | 364 |
| Elements - Boundary | 365 | Elements - Sub-Machine | 356 |
| Elements - Choice | 308 | Elements - Synch | 365 |
| Elements - Class | 309 | Elements - System Boundary | 365 |
| Elements - Collaboration | 313 | Elements - Terminate | 367 |
| Elements - Collaboration Occurrence | 314 | Elements - Use Case | 367 |
| Elements - Combined Fragment | 316 | Elements - Value Lifeline | 370 |
| Elements - Component | 321 | Embedded Elements | 463 |
| Elements - Continuation | 360 | Embedding and Features - Element Context Menu | 463 |
| Elements - Datastore | 321 | Enabling Security | 822 |
| Elements - Decision | 322 | Encrypt Password | 832 |
| Elements - Deployment Spec | 323 | Endpoint - Element | 325 |
| Elements - Diagram Gate | 324 | Enterprise Architect - What is EA | 3 |
| Elements - Document Artifact | 325 | Enterprise Architect Connections | 388 |
| Elements - Endpoint | 325 | Enterprise Architect Glossary | 1324 |
| Elements - Entry Point | 326 | Entity | 377 |
| Elements - Exception | 327 | Entity - Create | 377 |
| Elements - Exit Point | 329 | Entry Point - Element | 326 |
| Elements - Expansion Region | 327 | Enumerating - Project Interface | 1262 |
| Elements - Flow Final | 331 | Enumerating Diagrams - Project Interface | 1262 |
| Elements - Fork/Join | 332 | Enumerating Elements - Project Interface | 1265 |
| Elements - Fragment | 316 | Enumerating Links - Project Interface | 1274 |
| Elements - History | 335 | Enumerating Views - Project Interface | 1260 |
| Elements - Initial | 337 | Enumeration | 601 |
| Elements - Instance | 344 | Enumerations - Automation and Scripting - | |
| Elements - Interaction Occurrence | 338 | Automation Interface | 1178 |
| Elements - Interface | 339 | EnumRelationSetType Enum - Automation and | |
| Elements - Interruptible Activity Region | 340 | Scripting - Automation Interface | 1178 |
| Elements - Junction | 341 | Environment Complexity Factors | 871 |
| Elements - Lifeline | 343 | Estimating Project Size | 873 |
| Elements - Node | 343 | Estimation | 870 |
| Elements - Object | 344 | Estimation Hour Rate | 874 |
| Elements - Occurrence | 338 | Event | 377 |
| Elements - Package | 348 | Events | 377 |
| Elements - Part | 348 | Example UML Profile File | 614 |
| Elements - Partition | 349 | Examples and Tips - Automation and Scripting - | |
| Elements - Port | 351 | Automation Interface | 1171 |
| Elements - Receive | 354 | Exception - Element | 327 |
| Elements - Region | 355 | Exclude Elements in RTF | 1132 |
| Elements - Region - Expansion | 327 | Exclude Objects in RTF | 1132 |

exclusion 1098
Exit 61
Exit Point - Element 329
Expansion Region - Add 329
Expansion Region - Element 327
Export Perspectives 184
Export Profile 603
Export Reference Data 867
Expose Interface 401
Extend - Connection 401
Extended Class 285
Extended Stereotypes 371
Extension Points 368
External Requirements 622
External Responsibilities 512

- F -

Favorites 140
Features of EA 4
Feedback 11
Field Substitution Macros 967
File Menu 61
Filters 102
Find 101
Find in Project 62
Flow Final - Element 331
Foreign Keys 1075
Fork 333
Fork/Join - Element 332
Format Toolbar 118
Forum Options 685
Fragment - Element 316
Free Sorting 134
Function Macros 974

- G -

General Properties 510
General Settings 225, 510
General Types 857
Generalization Sets 545
Generalize - Connection 402
Generate RTF Report 1135
Generate Sequence Diagram 1024
Generate Virtual Documents 1164
Generate WSDL 1061

Generate XSD 1051
Get Diagram Image - Project Interface 1277
Get Diagrams - Project Interface 1263
Get Element - Project Interface 1266
Get Link - Project Interface 1275
Getting Started 28
Getting Started - About EA 38
Getting Started - Add License Key 34
Getting Started - Installing Enterprise Architect 28
Getting Started - Licence Management 33
Getting Started - License Information 38
Getting Started - Register Enterprise Architect 35
Getting Started - Registration Key 38
Getting Started - Upgrade an Existing License 37
Getting Started Starting Enterprise Architect 32
Glossary - A 1324
Glossary - B 1326
Glossary - C 1327
Glossary - D 1331
Glossary - E 1333
Glossary - F 1334
Glossary - G 1335
Glossary - H 1336
Glossary - I 1336
Glossary - J 1338
Glossary - L 1338
Glossary - M 1338
Glossary - N 1341
Glossary - O 1341, 1342
Glossary - Q 1345
Glossary - R 1345
Glossary - S 1347
Glossary - T 1351
Glossary - U 1353
Glossary - V 1354
Glossary of Terms 1324
Glossary Report Sample 915

- H -

Headers and Footers 1147
Help for Tagged Values 156
Help Menu 93
Hide Attributes 197
Hide Operations 197
Hide/Show Connectors 546
Hide/Show Labels 547
Hierarchy 146

- Hierarchy Window 146
 - Highlight Context Element 187
 - History - Element 335
 - Hotkeys 439
 - HTML and the Web 1158
 - HTML Documentation 1155
 - HTML Documents 1098
 - HTML Reports 1098, 1155
 - Hyperlinking diagrams 378
 - Hyperlinks 378
- | -**
- Image Handling 1142
 - Image Manager 452
 - Images 452
 - Implementation 628
 - Implementation Report 1152
 - Implementation Report - Target Types 1154
 - Import a Pattern 618
 - Import a UML Pattern 618
 - Import Action Scripts - Reverse Engineer Source Code 922
 - Import and Export Reference Data 866
 - Import Binary Module - Reverse Engineer Source Code 925
 - Import C 922
 - Import C# - Reverse Engineer Source Code 923
 - Import C++ - Reverse Engineer Source Code 922
 - Import Database Schema from ODBC 1091
 - Import DDL Schema from ODBC 1091
 - Import Delphi - Reverse Engineer Source Code 923
 - Import Java - Reverse Engineer Source Code 923
 - Import MDG Technologies 590
 - Import Perspectives 184
 - Import PHP - Reverse Engineer Source Code 924
 - Import Python - Reverse Engineer Source Code 924
 - Import Reference Data 867
 - Import Scenario as Test 889
 - Import Test 890
 - Import VB.Net - Reverse Engineer Source Code 924
 - Import Visual Basic - Reverse Engineer Source Code 924
 - Import WSDL 1061
 - Import XSD 1051
 - Import/Export Sub-Menu 130
 - Imported Class Elements 1095
 - Importing and Exporting Code Templates 986
 - Importing Transforms 646
 - Importing UML Profiles 605
 - Inbuilt and Extended Stereotypes 371
 - Include - Connection 403
 - Include or Exclude a Package from RTF Report 1138
 - Index 1083
 - Initial - Element 337
 - Initial Code 500
 - In-place Editor - 204
 - In-place Editor - Edit Attribute Keyword 209
 - In-place Editor - Edit Attribute Keyword and Operation Scope 208
 - In-place Editor - Edit Attribute or Operation Stereotype 207
 - In-place Editor - Edit Operation Parameter Keyword 211
 - In-place Editor - Edit Parameter Kind 212
 - In-place Editor - Element Item Tasks 205
 - In-place Editor - Insert Attribute or Operation 214
 - In-place Editor - Insert Maintenance Feature 217
 - In-place Editor - Insert Operation Parameter 215
 - In-place Editor - Insert Testing Feature 220
 - In-place Editor - Edit Element Name 206
 - Insert Diagram Properties Note 444
 - Insert Element at Cursor - Boundary 523
 - Insert Properties Note 444
 - Insert Related Elements 464
 - Installation 28
 - Installing Enterprise Architect 28
 - Instance Classifier 345
 - Instance Runtime State 347
 - Integration Testing 886
 - Integrity of Data 730
 - Interaction Diagrams 255
 - Interaction Occurrence - Element 338
 - Interaction Operators 318
 - Interaction Overview Diagram 269
 - Interface - Element 339
 - Internal Requirements 624
 - Interrupt Flow - Connection 405
 - Interruptible Activity Region - Add 341
 - Interruptible Activity Region - Element 340
 - Introducing Diagrams 432
 - Introducing UML Objects 291
 - Introduction 3
 - Introduction - Available Help File Formats 10

Introduction - Copyright Notice 5
Introduction - Differences Between Desktop 6
Introduction - EA Features 4
Introduction - EA for Power Users 11
Introduction - ICONIX 11
Introduction - License Agreement 7
Introduction - Ordering Enterprise Architect 10
Introduction - Professional and Corporate Editions 6
Introduction - Support 10
Introduction - Trademarks 5
Introduction - Your Comments and Feedback are Welcome 11
Introduction to Enterprise Architect 3
Invoking Methods - Debugging 1033
Issues - Adding
 Modifying 905
 Modifying (Model Issues Tab) 906
 Modifying (Project Issues Dialog) 905
Issues - Report Sample Output 909
Issues Report 907
Issues Report (Model Issues Tab) 908
Issues Report (Project Issues Dialog) 907

- J -

Java Options 956
Java Setup - Setting up a Debug session. 1006
Java Transformation 655
Join 334
Junciton - Element 341
JUnit Transformation 657

- K -

Keyboard Shortcuts 240

- L -

Label Visibility 547
Language Macros 945
Language Options 948
Language - Spelling 851
Layout a Diagram 188
Layout of Sequence Diagrams 259
Layouts 236
Legacy Document Options 457

Legend 190
Licence Management 33
License Information 38
Life Cycle 894
Lifecycle of an Element 258
Lifeline - Element 343
Lifeline Self-Message Hierarchy 261
Lifeline Termination 258
Line Angles - Tidy 542
Line Points - Add or Delete 542
Line Style 542
Link - Add a Note 538
Link a Note to an Element 526
Linked Documents 527
Linking Documents 527
Linking Documents - Creating Document Artifacts 529
Linking Documents - Creating Linked Document Templates 532
Linking Documents - Creating Linked Documents 529
Linking Documents - Editing Linked Document Templates 533
Linking Documents - Editing Linked Documents 531
Linking Documents - Linked Document Templates 532
Linking Documents - Linking Document to UML Elements (Corporate Edition) 530
Linking Notes 525
Linking Notes to Internal Documentation 525
Links 514
Load a Package 773
Local Directories 944
Local Options 223
Local Path Dialog 944
Local Paths 943
Local Pre/Post Conditions 300
Local Variables - Debugging 1020
Lock Diagram 192
Locking Model Elements 833
Locking Packages 835
Logical Diagram 274
Logical Model 51

- M -

Macros 945
Main Menu 60

| | | | |
|---|----------|--|-------------------------|
| Maintaining Users | 823 | MDA-Style Transforms - Java Transformation | 655 |
| Maintenance | 895 | MDA-Style Transforms - JUnit Transformation | 657 |
| Maintenance - Problem Types | 862 | MDA-Style Transforms - NUnit Transformation | 659 |
| Maintenance - Testing Types | 863 | MDA-Style Transforms - Transformation Templates | 646 |
| Maintenance Diagram | 287 | MDA-Style Transforms - Transforming Connectors | 670 |
| Maintenance Element | 894 | MDA-Style Transforms - Transforming Duplicate Information | 672 |
| Maintenance Element Properties | 902 | MDA-Style Transforms - Transforming Elements | 644 |
| Maintenance Group - UML Toolbox | 179 | MDA-Style Transforms - Writing Transformations | 665 |
| Maintenance Model | 55 | MDA-Style Transforms - WSDL Transformation | 661 |
| Maintenance Model Pattern | 55 | MDA-Style Transforms - XSD Transformation | 662 |
| Maintenance Modeling | 55 | MDG - Automation and Scripting | 1309 |
| Maintenance Pattern | 55 | MDG Add-In - Automation and Scripting - MDG Add-Ins - MDG Events | 1316 |
| Maintenance Script | 895 | MDG Events - Automation and Scripting - MDG Add-Ins | 1310 |
| Maintenance Support | 902 | MDG Menus Enumeration - Automation and Scripting - Automation Interface | 1178 |
| Maintenance Workspace | 895 | MDG Technologies | 579 |
| Making the Output Available on the Web | 1158 | MDG Technology | 579, 583, 584, 585, 587 |
| Manage Bookmarks | 918 | MDG_BuildProject - Automation and Scripting - MDG Add-Ins - MDG Events | 1310 |
| Manage Locks | 830 | MDG_Connect - Automation and Scripting - MDG Add-Ins - MDG Events | 1311 |
| Manage Views | 749 | MDG_Disconnect - Automation and Scripting - MDG Add-Ins - MDG Events | 1312 |
| Manage Views - Add Additional Views | 750 | MDG_GetConnectedPackages - Automation and Scripting - MDG Add-Ins - MDG Events | 1313 |
| Manage Views - Delete Views | 752 | MDG_GetProperty - Automation and Scripting - MDG Add-Ins - MDG Events | 1313 |
| Manage Views - Rename Views | 751 | MDG_Merge - Automation and Scripting - MDG Add-Ins - MDG Events | 1314 |
| Managing Baselines | 846 | MDG_PostGenerate - Automation and Scripting - MDG Add-Ins - MDG Events | 1317 |
| Managing Groups | 827 | MDG_PostMerge - Automation and Scripting - MDG Add-Ins - MDG Events | 1317 |
| Managing Inherited and Redefined Ports | 352 | MDG_PreGenerate - Automation and Scripting - MDG Add-Ins - MDG Events | 1318 |
| Managing Models | 675 | MDG_PreMerge - Automation and Scripting - MDG Add-Ins - MDG Events | 1319 |
| Managing Your Locks | 838 | MDG_PreReverse - Automation and Scripting - MDG Add-Ins - MDG Events | 1320 |
| Manifest - Connection | 406 | MDG_RunExe - Automation and Scripting - MDG Add-Ins - MDG Events | 1320 |
| Manual Version Control with XMI | 776 | MDG_View - Automation and Scripting - MDG Add-Ins - MDG Events | 1321 |
| Mapper - Data Type Conversion Procedures | 1088 | Mentor | 238 |
| Master | 839 | | |
| Matrix | 629 | | |
| Matrix Options | 633 | | |
| Matrix Profiles | 637 | | |
| MDA | 643, 646 | | |
| MDA-Style Transforms | 643 | | |
| MDA-Style Transforms - Build in Transformations | 648 | | |
| MDA-Style Transforms - C# Transformation | 648 | | |
| MDA-Style Transforms - Changing Transformations | 646 | | |
| MDA-Style Transforms - Converting Names | 672 | | |
| MDA-Style Transforms - Converting Types | 672 | | |
| MDA-Style Transforms - Cross References | 673 | | |
| MDA-Style Transforms - DDL Transformation | 650 | | |
| MDA-Style Transforms - EJB Transformations | 652 | | |
| MDA-Style Transforms - Importing Transforms | 646 | | |
| MDA-Style Transforms - Intermediary Language | 666 | | |

- Merge Element 322
- Message 410
- Message - Connection 406
- Message (Communication) 407
- Message (Sequence Diagram) 410
- Message (Timing Diagram) 415
- Message (Timing Diagram) Creation 416
- Message Creation 407
- Message Endpoint 418
- Message Label 419
- Message Scope 557
- Message Sequencing 407, 408
- Message Source and Target 557
- Metamodel Group - UML Toolbox 177
- Metric Types 881
- Metrics 877
- Metrics and Estimation 864
- Microsoft Word 1142
- Microsoft Word - Special Considerations 1145
- Model and Package - Automation and Scripting - Automation Interface 1174
- Model and Project Issues 903
- Model Authors 853
- Model Context Menu 123
- Model Context Menu - Root Node Package Control Submenu 125
- Model Driven Architecture 643
- Model Element Cross References 471
- Model Files - Connect to a MSDE Server Data Repository 714
- Model Files - Connect to a MySQL Data Repository 702
- Model Files - Connect to a Oracle9i Data Repository 708
- Model Files - Connect to a PostgreSQL Data Repository 710
- Model Files - Connect to a SQL Server Data Repository 705
- Model Files - Connect to an Adaptive Server Anywhere Data Repository 712
- Model Files - Connecting to a Data Repository 702
- Model Files - Create a Model 688
- Model Files - Create a New Adaptive Server Anywhere Repository 724
- Model Files - Create a New MSDE Server Repository 726
- Model Files - Create a New MySQL Repository 716
- Model Files - Create a New Oracle9i Server Repository 721
- Model Files - Create a New PostgreSQL Repository 721
- Model Files - Create a New SQL Server Repository 719
- Model Files - Create and Open Model Files Discussion 727
- Model Files - Creating a Repository 716
- Model Files - EA Project Files 686
- Model Files - Setting Up an ODBC Driver 690
- Model Files - Setting Up Database Model Files 689
- Model Files - Setup a MySQL ODBC Driver 690
- Model Files - Setup a PostgreSQL ODBC Driver 693
- Model Files - Setup an Adaptive Server Anywhere ODBC Driver 696
- Model Glossary 910
- Model Glossary - Glossary Dialog 911
- Model Glossary - Glossary Report 914
- Model Glossary - Using the Glossary Dialog 912
- Model Glossary - Using the Glossary Tab 913
- Model Glossary Glossary Tab 911
- Model Integrity 730
- Model Issues Tab 904
- Model Maintenance 747
- Model Maintenance - Compact a Project 747
- Model Maintenance - Rename a Project 747
- Model Maintenance - Repair a Project 748
- Model Management 675
- Model Pattern 47
- Model Patterns 688
- Model Search 102
- Model Task List 900
- Model Task Tab 900
- Model Validation 753
- Model Validation - Configuring Model Validation 755
- Model Validation - Rules Reference 755
- Model Validation Rule - Element Composition 756
- Model Validation Rule - OCL Conformance (Element, Relationship, Feature) 757
- Model Validation Rule - Property Validity (Element, Relationship, Feature) 757
- Model Validation Rule - Well Formedness (Element, Relationship, Feature, Diagram) 756
- Model Wizard 29, 688
- Modifying Tagged Values 154
- Modify a Task 901
- Modify Glossary Entries - Add 912
- Modify Glossary Entries - Delete 912
- Modify Glossary Entries - Modify 912

Modifying Relationships in Matrix 633
 MOF 760
 MOF - Export MOF to XMI 763
 MOF - Getting Started 762
 MOF - Import MOF from XMI 764
 Move Down 134
 Move Internal Responsibility to External Requirement 625
 Move Up 134
 Moving Elements between Packages 475
 Moving Objects between Packages 475
 MS Word tables 1148
 Multiple Select 447
 My 45
 MySQL Data Repository 702
 MySQL ODBC Driver 690
 MySQL Repository 716

- N -

N-Ary Association 381
 Nesting - Connection 420
 New Base Project 728
 New Diagrams 436
 New Element 199
 New Elements 468
 New Project... 61
 New RTF Style Template Editor 1104
 Node - Element 343
 Note - Add to Link/Connection 538
 Notes 141
 Notes - Creating Notes and Text 480
 Notes Window 141
 NUnit Transformation 659

- O -

Object - Element 344
 Object - Instance 344
 Object Appearance 230
 Object Classifiers 521
 Object Connections 514
 Object Diagram 276
 Object Flow - Connection 420
 Object Flows in Activity Diagrams 421
 Object Links 514
 Object Properties 509

Object Runtime State 347
 Object Scenarios 515
 Object Tasks 199
 Object TypeEnum - Automation and Scripting - Automation Interface 1179
 Occurrence - Connection 422
 Occurrence - Element 338
 ODBC - Data Modeling 1085
 Open a Package 184, 441
 Open a Report in Microsoft Word 1142
 Open Project... 61
 Open Source File... 61
 Open the Relationship Matrix 630
 Opening a Project - Project Interface 1260
 Opening External Tools 83
 Operation Detail 499
 Operation Parameters 501
 Operation Parameters by Reference 503
 Operation Tagged Values 505
 Operations 496
 Operations Constraints 505
 Operations Main Page 497
 Options 223
 Options - ActionScript 949
 Options - C# 950
 Options - C++ 951
 Options - Code Generation 236
 Options - Communication Colors 233
 Options - Delphi 952
 Options - Diagram Behavior 228
 Options - Diagram Settings 227
 Options - Element Visibility 231
 Options - General Settings 225
 Options - Java 956
 Options - Local 223
 Options - Object Appearance 230
 Options - PHP 956
 Options - Python 957
 Options - Sequence Diagrams 229
 Options - Standard Colors 226
 Options - UML Element Toolbox 235
 Options - VB.Net 958
 Options - Visual Basic 958
 Options - Visual Styles 237
 Options - XML Specifications 234
 Oracle9i Data Repository 708
 Oracle9i Server Repository 721
 Order Package Contents 134

Ordering Enterprise Architect 10
Other Documents 1151
Other Elements 371, 381
Other UML Elements 371
Output - Debugging 1021
Override Implementation 506
Override Parent Operations 506
Overriding Default Templates 984

- P -

Package - Element 348
Package Context Menu 125
Package Control Sub-Menu 127
Package Diagram 272
Package Import - Connection 423
Package Merge - Connection 423
Package Tasks 184
Page Setup... 61
Pan & Zoom 161
Pan a Diagram 192
Parameter 501
Parents and Interfaces 469
Part - Element 348
Partition - Element 349
Passing Parameters to External Tools 84
Password Encryption 832
Paste 447, 449
Paste Elements 62, 448
Paste Object as Link 446
Paste Object as New 446
Pasting Activities 449
Pasting Composite Elements 448
Pasting from the Project Browser 446
Pasting Multiple Items from the Project Browser 447
Patches - SQL 732
Patterns 616
People 853
Permission List 829
Perspectives 182
PHP Options 956
Pin End Point 117
Pin Start Point 117
Pkg Import - Connection 423
Pkg Merge - Connection 423
Place Related Elements on Current Diagram 450
Port - Element 351
Predefined Reference Data 150
Pre-defined Search Definitions 107
Predefined Tagged Value Types 148
Primary Key 1073
Print Preview 61
Print Setup... 61
Print... 61
Printing the Matrix 636
Problem Types 862
Process 382
Process Model 48
Process Modeling 48
Profile 45, 601, 602
Profile Constraints 600
Profile Group - UML Toolbox 176
Profile References 611
Profile Tags 596
Profiles Add Elements 595
Profiles Add Metaclass 595
Programming Languages Data Types 865
Project and Model Issues 903
Project Browser 122
Project Clients 854
Project Discussion Forum 676
Project Explorer 122
Project Interface (XML) 1258
Project Issues 903
Project Management 55, 870
Project Manager - Project Roles and EA 19
Project Menu 66
Project Model 55
Project Model Pattern 55
Project Modeling 55
Project Pattern 55
Project Roles 856
Project Roles and EA 13
Project Roles and EA - Business Analyst 13
Project Roles and EA - Database Administrator 25
Project Roles and EA - Deployment and Rollout 22
Project Roles and EA - Developer 17
Project Roles and EA - Project Manager 19
Project Roles and EA - Software Architects 15
Project Roles and EA - Software Engineer 16
Project Roles and EA - Technology Developer 23
Project Roles and EA - Testers 21
Project View 122
Properties 137
Properties - Element 509

Properties - Element Context Menu 461
 Properties - Object 509
 Properties Menu Section 535
 Properties Note 444
 Properties Window 137
 Property 279
 PropType Enum 1180
 Pseudo-States 254
 Python Conventions 994
 Python Options 957

- Q -

Qualifiers 353
 Quick Add of Tagged Values 520
 Quick Linker 161
 Quick Start 29

- R -

Read Only Interface - Project Interface 1259
 Realize - Connection 424
 Rearrange Package Order 1162
 Receive - Element 354
 rectangle notation 370, 462
 Recursion 382
 Redefine 602
 Redefinition 602
 Reference - Automation and Scripting - Automation Interface 1173
 Reference Data 852
 Reference Data - Clients 854
 Reference Data - Import/Export 866
 Reference Data - Model Authors 853
 Reference Data - People 853
 Reference Data - Resources 855
 Reference Data - Roles 856
 Region 357
 Region - Element 355
 Region - Expansion - Element 327
 Region - Interruptible Activity - Element 340
 Regions 253
 Register Add-In 237
 Register Enterprise Architect 35
 Register Zicom Mentor 238
 Registration Key 38
 Relation Visibility 549

Relationship Matrix 629
 Exporting to CSV 635
 Relationships Window 145
 Reload Current Project 61
 Remove a Defined Variable 347
 Remove Replication 843
 Rename a Project 747
 Rename Views 751
 Renaming Diagrams 438
 Renaming Packages 185
 Re-Order Messages 408
 Repair a Project 748
 Replication 839
 Report Templates 1138
 Report View 95
 Reports - HTML 1098, 1155
 Reports - RTF 1098
 Repository - Automation and Scripting - Automation Interface 1180
 Represents - Connection 425
 Requirement Properties 627
 Requirement Types 857
 Requirements 286, 382
 Requirements Analysis 48
 Requirements Diagram 286
 Requirements Group - UML Toolbox 178
 Requirements Hierarchy 628
 Requirements Management 48, 621
 Requirements Model Pattern 48
 Requirements Modeling 48
 Requirements Pattern 48
 Reset Options 959
 Resolving Conflicts 844
 Resource Allocation 875
 Resource Management 876
 Resource Report 878
 Resource View 140
 Resource Window 139
 Resources 139, 855
 Responsibilities 511
 Responsibility Compartment 525
 Reverse Connector 548
 Reverse Engineer ODBC data sources 1091
 Reverse Engineer Source Code - Handling of Classes not found during Import 926
 Rich Text File 1099
 Risk Management 877
 Risk Types 882

Robustness Diagram 290
Role Binding - Connection 425
Role Tagged Values 556
Roles 856, 879
Root Node 125
RTF Bookmarks 1143
RTF Custom Language Settings 1140
RTF Diagram Format 1132
RTF Dialog Options 1100
RTF Document Options 1102
RTF Documentation 1098
RTF Documents 1098
RTF Model Include 1133
RTF Options 1133
RTF Properties 1130
RTF Report Wizard 1129
RTF Reports 1098
RTF Selections 1134
RTF Templates 1135
RTF Templates Dialog 1104
Rule Window 145
Running SQL Patches 732
Runtime State 345
Runtime State - Add an Instance Variable 346
Runtime State - Delete an Instance Variable 347
Run-time Variable - Defining 346

- S -

Save a Package with XMI 772
Save as Document 1139
Save as UML Pattern 616
Save Image to File 440
Save Project As... 61
Save UML Pattern 616
Scale Image to Page Size 192
SCC 788
Scenario Testing 888
Scenario Types 861
Scenarios 515
Schema 290
Screen 383
Search Definitions 102
Search Filters 102
Search View 98
Searching a Project 101
Searching a Project - Adding Filters 107
Searching a Project - Creating a Search Definition 105
Searching a Project - Fields and Conditions 109
Searching a Project - Search Definitions 102
Searching a Project - Searching the Search View 101
Searching the model 102
Security Group Permissions 828
Security Policy 821
Security Users 823
Select a Data Source 1093
Select a Diagram 193
Select All 62
Select Alternate Image 452
Select By Type 62
Select Spelling Language 851
Select Tables 1094
Selecting Model Elements for Documentation 1105
Selecting Stereotypes 561
Self-Message 411
Self-Message Hierarchy 261
Send Element 355
Sequence Communication Messages 407, 408
Sequence Diagram 256
Sequence Diagram Layout 259
Sequence Diagram Top Margin 264
Sequence Diagrams 229
Sequence Element Activation 260
Sequence Elements 263
Sequence Group - UML Toolbox 171
Sequence Message 410
Set Appearance Options 194
Set Appearance Options - Visible Class Members 195
Set as Model Default 435
Set Association Specialization 549
Set Connector Style 542
Set Connector Visibility 549
Set Diagram Page Size 196
Set Feature Visibility 443
Set Group Permissions 828
Set Instance State 347
Set Message Source and Target 557
Set Object State 347
Set Parents and Interfaces 469
Set Relation Visibility 549
Set the Default Diagram 435
Set the Main RTF Properties 1130
Set Up Single Permissions 826

- Set Up User Groups 825
- Setting Collection Classes 946
- Setting Default Tree Behavior 134
- Setting Element Type 632
- Setting Source and Target Package 631
- Setting the Link Type and Direction 632
- Setting up a Debug session 1005
- Setting up a Debug session - .NET Setup 1008
- Setting up a Debug session - Debugging with EA 1005
- Setting up a Debug session - Java Setup 1006
- Setting Up an ODBC Driver 690
- Setting Up Database Model Files 689
- Settings Menu 89
- Setup a MySQL ODBC Driver 690
- Setup a PostgreSQL ODBC Driver 693
- Setup an Adaptive Server Anywhere ODBC Driver 696
- Shape Editor 578
- Shape Scripts 565
- Shape Scripts - Example Shape Scripts 575
- Shape Scripts - Getting Started 566
- Shape Scripts - Writing Scripts 568
- Shared Keys 34
- Sharing a Project 819
- Sharing an Enterprise Architect Project 818
- Shortcut menu - UML Toolbox 166
- Show or Hide Attributes and Operations 197
- Show or Hide Package Contents 186
- Show Other References 471
- Show/Hide Connectors 546
- Show/Hide Labels 547
- Single Permissions 826
- Single User 944
- Software Architects - Project Roles and EA 15
- Software Engineer - Project Roles and EA 16
- Source and Target - Set for Message 557
- Source Code 63
- Source Code Configuration - Build and Run 999
- Source Code Control 781
- Source Code Viewer 142
- Source Editor 142
- Source Role 554
- Specialize Association 549
- Spell Checking 849
- Spelling 849
- Spelling Language 851
- SQL - Data Modeling 1085
- SQL Patches 732
- SQL Server Data Repository 705
- SQL Server Repository 719
- Stack - Debugging 1021
- Standard Colors 226
- Standard Element Stereotypes 563, 864
- Start Page 29, 43
- Start Page Options 44
- Starting Enterprise Architect 32
- State - Element 356
- State Diagram 251
- State Group - UML Toolbox 172
- State Invariant 362
- State Invariant - Element 360
- State Lifeline - Element 359
- State Machine Diagram 251
- State Machine Regions 253
- State/Continuation - Element 360
- Status Bar 121
- Status Types 858
- Stereotype 596
- Stereotype - Applying 399
- Stereotype Dialog 560
- Stereotype Selector 561
- Stereotype Visibility 562
- Stereotypes 134, 557
- Stereotypes with Alternate Images 565
- Straighten Line at Cursor 542
- Structural Diagrams 271
- Style Menu Section 537
- SubActivity - Element 363
- Sub-Activity -Element 301
- SubMachine - Element 364
- Sub-Machine - Element 356
- Sub-states 357
- Subversion 800
- Support 10
- Supported Attributes in UML Profiles 613
- Supported Tags in UML Profiles 611
- Swimlanes 450
- Synch - Element 365
- Synchronization 839
- Synchronize UML Profile Tags and Constraints 610
- Synchronizing Code 986
- Synchronizing Existing Code Sections 986
- Synchronizing Replicas 842
- System 142
- System Boundary - Element 365

System Clients 880
System Tab 142
System Testing 886
System Users 823

- T -

Table 385
Tabular Sections 1107
Tag Compartment 524
Tagged Value Connector 599
Tagged Value Macros 974
Tagged Values 146, 150, 157, 517, 865
Tagged Values in UML Profiles 608
Tags 150, 596
Target Types 1154
Task Details 901
Task List 900
TCF Dialog 870
Team Development 817
Tear Off Menus 59
Technical Complexity Factors 870
Technology Developer 23
Template Editor 982
Template Package 202
Template Substitution Macros 967
Templates - Parameterized Classes 311
Terminate - Element 367
Test Details 892
Test Details Dialog 884
Test Documentation 893
Test Model 54
Test Model Pattern 54
Test Modeling 54
Test Pattern 54
Test Scripts 893
Testers 21
Testing 884
Testing Model 54
Testing Model Pattern 54
Testing Modeling 54
Testing Pattern 54
Testing Report 1154
Testing Support 883
Testing Types 863
Testing Workspace 884
Text - Creating Notes and Text 480
TFS Version Control 805
The Automation Interface 1167
The Code Template Editor 982
The Debug Toolbar 1017
The Debug Workbench 1028
The Generate HTML Report Dialog 1156
The Legacy RTF Dialog 1128
The Project Interface (XML) 1258
The RTF Dialog 1100
The Start Page 43
The Start Page Options 44
The UML Language 244
The UML Toolbox 164
The Upgrade Wizard 729
Tidy Line Angles 542
Time Event 354
Timing Diagram 255
Timing Group - UML Toolbox 171
Timing Message 415
Timing Message Creation 416
Toggle Line Points 542
Toolbars 110
Toolbars - Code Generation 112
Toolbars - Customize 119
Toolbars - Default Tools 111
Toolbars - Diagram 115
Toolbars - Diagram Tabs 120
Toolbars - Element 114
Toolbars - Project 111
Toolbars - Status Bar 121
Toolbars - UML Elements 114
Toolbars - Workspace Views 119
Toolbox 164
Toolbox Options 235
Toolbox Shortcut menu 166
Tools Window 76
Top Margin 264
Trace - Connection 426
Trademarks 5
Transformation 643, 646
Transformation Templates 646
Transformations 643
Transformations - Build in - MDA-Style Transforms 648
Transforming Elements - MDA-Style Transforms 644
Transforms 646
Transition - Connection 427
Tree Style Hierarchy 551

Triangle 918
 Tricks and Traps - Automation and Scripting -
 Creating Add-In 1281
 Trigger 1083
 Type - Element Context Menu 462
 Type Hierarchy 469
 Type Specific Menu Section 536

- U -

UI Element 385
 UML 864
 UML - What is UML 244
 UML Behavioral Diagrams 246
 UML Connections 388
 UML Diagrams 245, 284, 432
 UML Element Toolbar 114
 UML Element Toolbox 235
 UML Elements 291, 295
 UML Elements - Behavioral Diagram Elements 291
 UML Elements - Structural Diagram Elements 294
 UML Language 244
 UML Mentor 238
 UML Pattern - Add to Diagram 618
 UML Patterns 616
 UML Profile for XSD 1041
 UML Profile Structure 612
 UML Profiles 592
 UML Stereotypes 557
 UML Structural Diagrams 271
 UML Tagged Values 865
 UML Toolbox 164
 UML Toolbox - Activity Group 172
 UML Toolbox - Analysis Group 166
 UML Toolbox - Class Group 168
 UML Toolbox - Collaboration Group 170
 UML Toolbox - Component Group 173
 UML Toolbox - Composite Group 169
 UML Toolbox - Custom Group 175
 UML Toolbox - Deployment Group 174
 UML Toolbox - Maintenance Group 179
 UML Toolbox - Metamodel Group 177
 UML Toolbox - Profile Group 176
 UML Toolbox - Requirements Group 178
 UML Toolbox - Sequence Group 171
 UML Toolbox - State Group 172
 UML Toolbox - Timing Group 171
 UML Toolbox - Toolbox Shortcut menu 166

UML Toolbox - Use Case Group 167
 UML Toolbox - User Interface Group 179
 UML Toolbox - WSDL Group 177
 UML Toolbox - XML Schema Group 178
 Undo (Edit Menu) 62
 Undo Last Action 198
 Unit Testing 885
 Update links in Word 1150
 Update Package Status 916
 Upgrade an Existing License 37
 Upgrade Replicas 730
 Upgrade Wizard 729
 Upgrading Models 729
 Upgrading Models - Upgrade Replicas 730
 Upgrading Models - Upgrade Wizard 729
 Upgrading Projects 729
 Upgrading Replicas 843
 Upsizing 736
 Upsizing Models 736
 Upsizing to MySQL 737
 Upsizing to Oracle 740
 Upsizing to PostgreSQL 742
 Upsizing to SQL Server 738
 Upsizing to SQL Server Desktop Engine (MSDE)
 745
 Upsizing to Sybase Adaptive Server Anywhere (ASA)
 743
 Use - Connection 429
 Use a Pattern 618
 Use Case 49
 Use Case - Element 367
 Use Case Arrowhead 550
 Use Case Diagram 250
 Use Case Extension Points 368
 Use Case Group - UML Toolbox 167
 Use Case Metrics 873
 Use Case Model 49
 Use Case Model Pattern 49
 Use Case Modeling 49
 Use Case Pattern 49
 User Dictionaries 851
 User Groups 823
 User Interface 288
 User Interface Diagram 288
 User Interface Group - UML Toolbox 179
 User Security 820
 User Security Groups 825
 User Settings 944

- Using Classifiers 521
 - Using Enterprise Architect 41
 - Using Enterprise Architect - Application Workspace 41
 - Using Enterprise Architect - Arranging Windows and Menus 56
 - Using Enterprise Architect - Arranging Windows and Menus - Autohide Windows 59
 - Using Enterprise Architect - Arranging Windows and Menus - Dockable Windows 56
 - Using Enterprise Architect - Arranging Windows and Menus - Dockable Windows 2005 Style 57
 - Using Enterprise Architect - Arranging Windows and Menus - Tear Off Menus 59
 - Using Enterprise Architect - Manage My Profile 45
 - Using Enterprise Architect - Removing Recent Models 44
 - Using Enterprise Architect - Start Page 43
 - Using Enterprise Architect - Start Page Options 44
 - Using MS Word 1142
 - Using Object Flows in Activity Diagrams 421
 - Using Profiles 605
 - Using Rectangle Notation 370
 - Using the Debugger 1016
 - Using the Spell Checker 849
 - Using UML 431
- V -**
- Value Lifeline - Element 370
 - Variable Definitions 980
 - Variable References 981
 - Variables - Debugging 1029
 - VB - Set up - Automation and Scripting - Automation Interface 1169
 - VB.Net Options 958
 - Version Control 781
 - Version Control - Add Previously defined Version Control Configuration 814
 - Version Control - Checking In and Out 809
 - Version Control - Configure Package for Version Control 807
 - Version Control - CVS 792
 - Version Control - CVS Options 797
 - Version Control - CVS Remote Repository 793
 - Version Control - Menu 787
 - Version Control - Offline Version Control 810
 - Version Control - Options Dialog 785
 - Version Control - Other users packages 815
 - Version Control - Reference 784
 - Version Control - Remove Package from Version Control 813
 - Version Control - Review version control history 812
 - Version Control - SCC 788
 - Version Control - SCC - Providers Dialog 791
 - Version Control - SCC - Options SCC 792
 - Version Control - Set Up Version Control 783
 - Version Control - Setup Menu 784
 - Version Control - Specifying Private or Shared Models 816
 - Version Control - Subversion 800
 - Version Control - Subversion - Configure Version Control 802
 - Version Control - Subversion - Create a Local Working Copy 801
 - Version Control - Subversion - Creating a new Repository Sub-tree 801
 - Version Control - Subversion - Setting up Subversion 800
 - Version Control - Subversion - TortoiseSVN 804
 - Version Control - Upgrade SCC for Version 4.5 814
 - Version Control - Using Nested Version Control Packages 817
 - Version Control - Using Version Control 783
 - Version Control Overview 782
 - Version Control with TFS 805
 - View and Manage Locks 830
 - View Last and Next Diagram 198
 - View Menu 63
 - View Options 94
 - View Options - Diagram View 94
 - View Options - Report View 95
 - View Options - Search View 98
 - Views 749
 - Virtual Documents 1160
 - Virtual Documents - Adding Packages 1161
 - Virtual Documents - Delete a Package 1163
 - Virtual Documents - Generate the Document 1164
 - Virtual Documents - Rearranging Package Order 1162
 - Virtual Report - Document Object 1161
 - Visible Class Members 195
 - Visual Basic - Automation and Scripting - Automation Interface 1169
 - Visual Basic Options 958
 - Visual Layouts 236
 - Visual Styles 237

- W -

Web Documentation 1158
 Web Services (WSDL) 1052
 Web Services (WSDL) - Generate WSDL 1061
 Web Services (WSDL) - Import WSDL 1061
 Web Services (WSDL) - Model WSDL 1052
 Web Services (WSDL) - Model WSDL - Binding 1058
 Web Services (WSDL) - Model WSDL - Document 1054
 Web Services (WSDL) - Model WSDL - Message 1057
 Web Services (WSDL) - Model WSDL - Message Part 1061
 Web Services (WSDL) - Model WSDL - Port Type 1057
 Web Services (WSDL) - Model WSDL - Port Type Operation 1059
 Web Services (WSDL) - Model WSDL - Service 1055
 Web Services (WSDL) - Model WSDL Namepaces 1053
 Web Stereotypes 387
 Web Style Templates 1158
 Web Templates 1158
 What is UML 244
 Window Menu 92
 Windows and Menus 56
 Windows Authentication 823
 Word - Special Considerations 1145
 Word Substitution 1102
 Workbench 1028
 Workbench Variables 1029
 Worker 388
 Working with MDG Technologies 588
 Working with UML Connections 535
 Working with UML Elements 458
 Working with UML Profiles 594
 Workspace Toolbars 110
 WSDL 1052
 WSDL Group - UML Toolbox 177
 WSDL Transformation 661

- X -

XMI - Export to XMI 765
 XMI - Import from XMI 766

XMI - Limitations of XMI 768
 XMI - UML DTD 768
 XMI Import and Export 764
 XMI Type Enum - Automation and Scripting - Automation Interface 1179
 XML Schema (XSD) 1039
 XML Schema Group - UML Toolbox 178
 XML Specifications 234
 XML Technologies 1039
 XSD - XML Schema 1039
 XSD Datatype Packages 1048
 XSD Transformation 662

- Z -

Z Order Elements 439
 Zicom UML Mentor 238
 Zoom a Diagram View 198

Enterprise Architect User Guide
www.sparxsystems.com