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

© 1998-2008 Sparx Systems Pty Ltd

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: October 2008

Special thanks to:

All the people who have contributed suggestions, examples, bug reports and assistance in the development of Enterprise Architect. The task of developing and maintaining this tool has been greatly enhanced by their contribution.
# Table of Contents

## Foreword

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

## Part I  Enterprise Architect UML Tool

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is Enterprise Architect?</td>
<td>5</td>
</tr>
<tr>
<td>Uses of Enterprise Architect</td>
<td>6</td>
</tr>
<tr>
<td>Key Features</td>
<td>8</td>
</tr>
<tr>
<td>Differences Between Editions</td>
<td>9</td>
</tr>
<tr>
<td>Sparx Systems MDG Add-Ins</td>
<td>12</td>
</tr>
<tr>
<td>Help and Support</td>
<td>13</td>
</tr>
<tr>
<td>Available Helpfile Formats</td>
<td>14</td>
</tr>
<tr>
<td>Support</td>
<td>15</td>
</tr>
<tr>
<td>Formal Statements</td>
<td>16</td>
</tr>
<tr>
<td>Copyright Notice</td>
<td>17</td>
</tr>
<tr>
<td>End User License Agreement</td>
<td>18</td>
</tr>
<tr>
<td>Trademarks</td>
<td>21</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>22</td>
</tr>
<tr>
<td>If You Have the Trial Version</td>
<td>23</td>
</tr>
<tr>
<td>Order Enterprise Architect</td>
<td>24</td>
</tr>
<tr>
<td>Installation</td>
<td>25</td>
</tr>
<tr>
<td>Register a Full License</td>
<td>26</td>
</tr>
</tbody>
</table>

## Part II  Start UML Modeling

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work with Enterprise Architect</td>
<td>30</td>
</tr>
<tr>
<td>Modeling With UML</td>
<td>32</td>
</tr>
<tr>
<td>Manage UML Models</td>
<td>33</td>
</tr>
<tr>
<td>Code Engineering</td>
<td>34</td>
</tr>
<tr>
<td>Quick Start - Create a Project</td>
<td>35</td>
</tr>
<tr>
<td>Add a View to a Model</td>
<td>37</td>
</tr>
<tr>
<td>Add a Package To a Model</td>
<td>38</td>
</tr>
<tr>
<td>Add a Diagram to a Package</td>
<td>39</td>
</tr>
<tr>
<td>Add Elements</td>
<td>40</td>
</tr>
<tr>
<td>Add Connectors</td>
<td>41</td>
</tr>
<tr>
<td>Define Properties</td>
<td>42</td>
</tr>
<tr>
<td>Move Components</td>
<td>44</td>
</tr>
<tr>
<td>Delete Components</td>
<td>46</td>
</tr>
<tr>
<td>Save Changes</td>
<td>47</td>
</tr>
<tr>
<td>Quick Start - Project Tasks</td>
<td>48</td>
</tr>
</tbody>
</table>

## Part III  UML Modeling Tool Features

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Enterprise Architect</td>
<td>52</td>
</tr>
<tr>
<td>The User Interface</td>
<td>53</td>
</tr>
<tr>
<td>The Start Page</td>
<td>55</td>
</tr>
<tr>
<td>Remove Recent Projects</td>
<td>57</td>
</tr>
<tr>
<td>Model Templates</td>
<td>58</td>
</tr>
<tr>
<td>Business Process Model Template</td>
<td>59</td>
</tr>
<tr>
<td>Requirements Model Template</td>
<td>60</td>
</tr>
<tr>
<td>Use Case Model Template</td>
<td>61</td>
</tr>
<tr>
<td>Domain Model Template</td>
<td>62</td>
</tr>
<tr>
<td>Class Model Template</td>
<td>63</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Database Model Template</td>
<td>64</td>
</tr>
<tr>
<td>Component Model Template</td>
<td>65</td>
</tr>
<tr>
<td>Deployment Model Template</td>
<td>66</td>
</tr>
<tr>
<td>Testing Model Template</td>
<td>67</td>
</tr>
<tr>
<td>Maintenance Model Template</td>
<td>68</td>
</tr>
<tr>
<td>Project Model Template</td>
<td>69</td>
</tr>
<tr>
<td>The Project Browser</td>
<td>70</td>
</tr>
<tr>
<td>Order Package Contents</td>
<td>72</td>
</tr>
<tr>
<td>Set Default Behavior</td>
<td>73</td>
</tr>
<tr>
<td>Project Browser Toolbar</td>
<td>75</td>
</tr>
<tr>
<td>Project Browser Icon Overlays</td>
<td>76</td>
</tr>
<tr>
<td>Model (Root Node) Context Menu</td>
<td>77</td>
</tr>
<tr>
<td>Package Control Sub-Menu</td>
<td>78</td>
</tr>
<tr>
<td>Package Menu</td>
<td>79</td>
</tr>
<tr>
<td>Add Sub-Menu</td>
<td>80</td>
</tr>
<tr>
<td>Documentation Sub-Menu</td>
<td>80</td>
</tr>
<tr>
<td>Code Engineering Sub-Menu</td>
<td>80</td>
</tr>
<tr>
<td>Build and Run Sub-Menu</td>
<td>81</td>
</tr>
<tr>
<td>Import/Export Sub-Menu</td>
<td>81</td>
</tr>
<tr>
<td>Contents Sub-Menu</td>
<td>81</td>
</tr>
<tr>
<td>The Element Menu - Project Browser</td>
<td>83</td>
</tr>
<tr>
<td>Add Sub Menu</td>
<td>83</td>
</tr>
<tr>
<td>The Diagram Menu - Project Browser</td>
<td>84</td>
</tr>
<tr>
<td>Operation Menu - Project Browser</td>
<td>85</td>
</tr>
<tr>
<td>The Main Menu</td>
<td>86</td>
</tr>
<tr>
<td>The File Menu</td>
<td>87</td>
</tr>
<tr>
<td>Print Preview</td>
<td>87</td>
</tr>
<tr>
<td>Save Model Copy or Shortcut</td>
<td>88</td>
</tr>
<tr>
<td>Create Copy Or Shortcut</td>
<td>89</td>
</tr>
<tr>
<td>Capture Current Work Environment</td>
<td>90</td>
</tr>
<tr>
<td>Encrypt Repository Password</td>
<td>92</td>
</tr>
<tr>
<td>The Edit Menu</td>
<td>93</td>
</tr>
<tr>
<td>Paste Elements Submenu</td>
<td>93</td>
</tr>
<tr>
<td>The View Menu</td>
<td>95</td>
</tr>
<tr>
<td>View Submenus</td>
<td>96</td>
</tr>
<tr>
<td>The Project Menu</td>
<td>98</td>
</tr>
<tr>
<td>Documentation Submenu</td>
<td>98</td>
</tr>
<tr>
<td>Source Code Engineering Submenu</td>
<td>99</td>
</tr>
<tr>
<td>Build and Run Submenu</td>
<td>99</td>
</tr>
<tr>
<td>Database Engineering Submenu</td>
<td>100</td>
</tr>
<tr>
<td>Model Transformations Submenu</td>
<td>100</td>
</tr>
<tr>
<td>Model Validation Submenu</td>
<td>101</td>
</tr>
<tr>
<td>Web Services Submenu</td>
<td>101</td>
</tr>
<tr>
<td>XML Schema Submenu</td>
<td>101</td>
</tr>
<tr>
<td>Security Submenu</td>
<td>101</td>
</tr>
<tr>
<td>Version Control Submenu</td>
<td>102</td>
</tr>
<tr>
<td>Import/Export Submenu</td>
<td>102</td>
</tr>
<tr>
<td>The Diagram Menu</td>
<td>103</td>
</tr>
<tr>
<td>The Element Menu</td>
<td>105</td>
</tr>
<tr>
<td>Inline Features Submenu</td>
<td>105</td>
</tr>
<tr>
<td>Advanced Submenu</td>
<td>106</td>
</tr>
<tr>
<td>Source Code Engineering Submenu</td>
<td>106</td>
</tr>
<tr>
<td>Appearance Submenu</td>
<td>107</td>
</tr>
<tr>
<td>Position Submenus</td>
<td>107</td>
</tr>
<tr>
<td>The Tools Menu</td>
<td>109</td>
</tr>
<tr>
<td>Data Management Submenu</td>
<td>109</td>
</tr>
<tr>
<td>Manage .EAP File Submenu</td>
<td>109</td>
</tr>
<tr>
<td>The Customize Dialog</td>
<td>110</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Customize Commands</td>
<td>111</td>
</tr>
<tr>
<td>Customize Toolbars</td>
<td>111</td>
</tr>
<tr>
<td>Custom Tools</td>
<td>114</td>
</tr>
<tr>
<td>Open External Tools</td>
<td>116</td>
</tr>
<tr>
<td>Pass Parameters to Applications</td>
<td>117</td>
</tr>
<tr>
<td>Customize Keyboard</td>
<td>118</td>
</tr>
<tr>
<td>Customize Menu</td>
<td>120</td>
</tr>
<tr>
<td>Customize Options</td>
<td>121</td>
</tr>
<tr>
<td>The Add-Ins Menu</td>
<td>122</td>
</tr>
<tr>
<td>The Settings Menu</td>
<td>123</td>
</tr>
<tr>
<td>The Window Menu</td>
<td>124</td>
</tr>
<tr>
<td>The Help Menu</td>
<td>125</td>
</tr>
<tr>
<td>The Enterprise Architect Toolbox</td>
<td>126</td>
</tr>
<tr>
<td>UML Toolbox Appearance Options</td>
<td>128</td>
</tr>
<tr>
<td>UML Toolbox Shortcut Menu</td>
<td>131</td>
</tr>
<tr>
<td>Common Group</td>
<td>133</td>
</tr>
<tr>
<td>Use Case Group</td>
<td>134</td>
</tr>
<tr>
<td>Class Group</td>
<td>135</td>
</tr>
<tr>
<td>Object Group</td>
<td>136</td>
</tr>
<tr>
<td>Composite Group</td>
<td>137</td>
</tr>
<tr>
<td>Communication Group</td>
<td>138</td>
</tr>
<tr>
<td>Interaction Group</td>
<td>139</td>
</tr>
<tr>
<td>Timing Group</td>
<td>140</td>
</tr>
<tr>
<td>State Group</td>
<td>141</td>
</tr>
<tr>
<td>Activity Group</td>
<td>142</td>
</tr>
<tr>
<td>Component Group</td>
<td>143</td>
</tr>
<tr>
<td>Deployment Group</td>
<td>144</td>
</tr>
<tr>
<td>Profile Group</td>
<td>145</td>
</tr>
<tr>
<td>Metamodel Group</td>
<td>146</td>
</tr>
<tr>
<td>Analysis Group</td>
<td>147</td>
</tr>
<tr>
<td>Custom Group</td>
<td>148</td>
</tr>
<tr>
<td>Requirement Group</td>
<td>149</td>
</tr>
<tr>
<td>Maintenance Group</td>
<td>150</td>
</tr>
<tr>
<td>User Interface Group</td>
<td>151</td>
</tr>
<tr>
<td>WSDL Group</td>
<td>153</td>
</tr>
<tr>
<td>XML Schema Group</td>
<td>154</td>
</tr>
<tr>
<td>Data Modeling Group</td>
<td>155</td>
</tr>
<tr>
<td>Workspace Toolbars</td>
<td>156</td>
</tr>
<tr>
<td>Default Tools Toolbar</td>
<td>158</td>
</tr>
<tr>
<td>Project Toolbar</td>
<td>159</td>
</tr>
<tr>
<td>Code Generation Toolbar</td>
<td>160</td>
</tr>
<tr>
<td>UML Elements Toolbar</td>
<td>162</td>
</tr>
<tr>
<td>Diagram Toolbar</td>
<td>163</td>
</tr>
<tr>
<td>Current Element Toolbar</td>
<td>164</td>
</tr>
<tr>
<td>Current Connector Toolbar</td>
<td>165</td>
</tr>
<tr>
<td>Format Toolbar</td>
<td>166</td>
</tr>
<tr>
<td>Workspace Views</td>
<td>167</td>
</tr>
<tr>
<td>Other Views Toolbar</td>
<td>168</td>
</tr>
<tr>
<td>Status Bar</td>
<td>169</td>
</tr>
<tr>
<td>Rich Text Notes Toolbar</td>
<td>170</td>
</tr>
<tr>
<td>Diagram Tabs</td>
<td>171</td>
</tr>
<tr>
<td>View Options</td>
<td>172</td>
</tr>
<tr>
<td>Diagram View</td>
<td>173</td>
</tr>
<tr>
<td>Element List</td>
<td>174</td>
</tr>
<tr>
<td>Element List Options</td>
<td>175</td>
</tr>
<tr>
<td>Model Views</td>
<td>177</td>
</tr>
<tr>
<td>Model Views Toolbar</td>
<td>178</td>
</tr>
</tbody>
</table>
Part IV UML Tool Project Roles

Business Analysts .......................................................... 271
Software Architects .......................................................... 273
Software Engineers .......................................................... 275
Developers ........................................................................ 276
Project Managers .............................................................. 278
Testers ............................................................................. 279
Implementation Manager .................................................. 280
Technology Developers .................................................... 281
Database Administrators .................................................. 283

Part V What Is UML Modeling?

Work With Packages ......................................................... 287
Open Package in the Project Browser ................................ 288
Add a Package ................................................................... 289
Rename a Package ............................................................. 290
Drag a Package Onto a Diagram ......................................... 291
Show or Hide Package Contents ....................................... 292
Delete a Package ................................................................ 293

Work With Diagrams ........................................................ 294
Diagram Context Menu ..................................................... 297
Diagram Tasks .................................................................. 299
Add New Diagrams ............................................................ 299
Lay Out a Diagram .............................................................. 300
Delete Diagram ................................................................ 303
Rename Diagram ............................................................... 304
Copy And Paste Diagram Element ..................................... 304
Diagram Navigation Hotkeys .......................................... 304
Copy Image to Disk .......................................................... 304
Copy Image to Clipboard ................................................ 305
Change Diagram Type ...................................................... 305
Z Order Elements ............................................................ 306
Copy Diagram .................................................................. 306
Open Package From Diagram ......................................... 307
Feature Visibility ............................................................... 307
Insert Diagram Properties Note ....................................... 309
Autosize Elements ............................................................ 310
Paste from Project Browser ............................................. 310
Work With Elements......................................................................................................................................................... 338
Element Context Menu ......................................................................................................................................................... 341
Properties Menu Section ......................................................................................................................................................... 342
Advanced Submenu......................................................................................................................................................... 343
Custom Properties Dialog......................................................................................................................................................... 344
Add Submenu......................................................................................................................................................... 345
Inserted Related Elements......................................................................................................................................................... 346
Find Submenu......................................................................................................................................................... 347
Embedded Elements Submenu......................................................................................................................................................... 348
Embedded Elements Window......................................................................................................................................................... 349
Features Menu Section......................................................................................................................................................... 350
Code Engineering Menu Section......................................................................................................................................................... 351
Appearance Menu Section......................................................................................................................................................... 352
Set Element Font......................................................................................................................................................... 353
Element Multiple Selection Menu......................................................................................................................................................... 354
Element Tasks......................................................................................................................................................... 355
Create Elements......................................................................................................................................................... 356
Add Elements Directly To Packages......................................................................................................................................................... 357
Use Auto Naming and Auto Counters......................................................................................................................................................... 358
Set Element Parent......................................................................................................................................................... 359
Show Element Usage......................................................................................................................................................... 360
Set Up Cross References......................................................................................................................................................... 361
Move Elements Within Diagrams......................................................................................................................................................... 362
Move Elements Between Packages......................................................................................................................................................... 363
Change Element Type......................................................................................................................................................... 364
Align Elements......................................................................................................................................................... 365
Resize Elements......................................................................................................................................................... 366
Delete Elements......................................................................................................................................................... 367
Customize Visibility of Elements......................................................................................................................................................... 368
Create Notes and Text......................................................................................................................................................... 369
Resize Elements......................................................................................................................................................... 370
Move Elements Between Packages......................................................................................................................................................... 371
Set Up Cross References......................................................................................................................................................... 372
Move Elements Within Diagrams......................................................................................................................................................... 373
Change Element Type......................................................................................................................................................... 374
Align Elements......................................................................................................................................................... 375
Resize Elements......................................................................................................................................................... 376
Delete Elements......................................................................................................................................................... 377
Customize Visibility of Elements......................................................................................................................................................... 378
Create Notes and Text......................................................................................................................................................... 379
Paste Multiple Items......................................................................................................................................................... 380
Paste Composite Elements......................................................................................................................................................... 381
Paste Activities......................................................................................................................................................... 382
Drop Elements from Project Browser......................................................................................................................................................... 383
Place Related Elements on Diagram......................................................................................................................................................... 384
Swimlanes......................................................................................................................................................... 385
Swimlanes Matrix......................................................................................................................................................... 386
Using the Image Manager......................................................................................................................................................... 387
Create Custom Diagram Background......................................................................................................................................................... 388
Import Image Library......................................................................................................................................................... 389
Show Realized Interfaces of Class......................................................................................................................................................... 390
Label Menu Section......................................................................................................................................................... 391
Lock Diagram......................................................................................................................................................... 392
Undo Last Action......................................................................................................................................................... 393
Redo Last Action......................................................................................................................................................... 394
View Last and Next Diagram......................................................................................................................................................... 395
Diagram Properties......................................................................................................................................................... 396
General Tab......................................................................................................................................................... 397
Diagram Tab......................................................................................................................................................... 398
Elements Tab......................................................................................................................................................... 399
Features Tab......................................................................................................................................................... 400
Connectors Tab......................................................................................................................................................... 401
Visible Class Members......................................................................................................................................................... 402
Set the Default Diagram......................................................................................................................................................... 403
Create Legends......................................................................................................................................................... 404
Scale Image to Page Size......................................................................................................................................................... 405
Set Diagram Page Size......................................................................................................................................................... 406
Pan and Zoom a Diagram......................................................................................................................................................... 407
Using the Image Manager......................................................................................................................................................... 408
Swimlanes......................................................................................................................................................... 409
Drop Elements from Project Browser......................................................................................................................................................... 410
Set Element Font......................................................................................................................................................... 411
Embedded Elements Window......................................................................................................................................................... 412
Insert Related Elements......................................................................................................................................................... 413
Custom Properties Dialog......................................................................................................................................................... 414
Visible Class Members......................................................................................................................................................... 415
Connectors Tab......................................................................................................................................................... 416
Elements Tab......................................................................................................................................................... 417
General Tab......................................................................................................................................................... 418
Features Tab......................................................................................................................................................... 419
Elements Tab......................................................................................................................................................... 420
General Tab......................................................................................................................................................... 421
Features Tab......................................................................................................................................................... 422
Elements Tab......................................................................................................................................................... 423
Set an Element's Default Appearance ................................................................. 365
Get/Set Project Custom Colors ........................................................................ 367
Set Element Templates Package ........................................................................ 369
Highlight Context Element .................................................................................. 370
Make Linked Element a Local Copy .................................................................... 371
Copy Features Between Elements ....................................................................... 371
Move Features Between Elements ...................................................................... 372

Attributes ........................................................................................................... 374
Attributes Dialog - General Tab ............................................................................ 376
Attributes Dialog - Detail ..................................................................................... 378
Attributes Dialog - Constraints .......................................................................... 379
Attribute Tagged Values ....................................................................................... 379
Create Properties ................................................................................................ 381
Display Inherited Attributes ............................................................................... 383
Create Object From Attribute ............................................................................ 384

Operations ........................................................................................................... 385
Operations Dialog - General ................................................................................ 386
  Operation Parameters ......................................................................................... 389
    Operation Parameter Tagged Values .............................................................. 390
    Operation Parameters by Reference ............................................................. 391
  Operations Dialog - Behavior ............................................................................ 391
    Initial Code ...................................................................................................... 393
  Operations Dialog - Constraints ...................................................................... 393
  Operation Tagged Values ............................................................................... 394
  Override Parent Operations ............................................................................. 395
  Display Inherited Operations ........................................................................... 396

Element In-place Editing Options ........................................................................ 398
  In-place Element Item Tasks ............................................................................. 398
    Edit Element Item Name .................................................................................. 399
    Edit Feature Stereotype ................................................................................... 399
    Edit Feature Scope ......................................................................................... 400
    Edit Attribute Keyword .................................................................................. 401
    Edit Operation Parameter Keyword ............................................................ 402
    Edit Parameter Kind ....................................................................................... 403
    Insert New Feature ......................................................................................... 403
    Insert Operation Parameter .......................................................................... 404
    Insert Maintenance Feature ......................................................................... 405
    Insert Testing Features .................................................................................. 407

Properties ............................................................................................................ 409
  General Settings ............................................................................................... 410
    Advanced Settings .......................................................................................... 411
  Details ................................................................................................................ 412
  Requirements .................................................................................................... 413
    External Requirements .................................................................................... 414
  Constraints ......................................................................................................... 415
  Links ................................................................................................................... 417
  Scenarios .......................................................................................................... 418
  Associated Files ................................................................................................ 418
  Tagged Values ................................................................................................... 419
    Advanced Tag Management ............................................................................ 420
    Quick Add of Tagged Values .......................................................................... 421
  Object Classifiers .............................................................................................. 422
    Using Classifiers ............................................................................................. 423
    Set Element Classifier Dialog ...................................................................... 424
  Boundary Element Settings .............................................................................. 424

Compartments ...................................................................................................... 426
Link Note to Internal Documentation ................................................................... 428
Linked Documents ............................................................................................... 430
Work With Connectors

Connector Context Menu ................................................................. 439
Properties Menu Section............................................................... 439
Type-Specific Menu Section ......................................................... 440
Advanced Menu Section ............................................................... 440
Style Menu Section ...................................................................... 441
Appearance Menu Section ............................................................ 441
Connector Tasks .......................................................................... 442
Connect Elements......................................................................... 442
Add a Note to a Connector ............................................................ 443
Arrange Connectors...................................................................... 444
Change Connector Type ............................................................... 445
Change the Source or Target Element ......................................... 445
Connector Styles .......................................................................... 446
Create Connector in Project Browser .......................................... 446
Delete Connectors ....................................................................... 449
Generalization Sets ................................................................... 450
Hide/Show Connectors ............................................................... 451
Hide/Show Labels ...................................................................... 452
Connector In-place Editing Options ............................................. 453
Reverse Connector...................................................................... 453
Set Association Specializations .................................................. 453
Show Uses Arrow Head ............................................................... 454
Relationship Visibility ............................................................... 454
Tree Style Hierarchy ................................................................... 455
Connector Properties ................................................................... 457
Connector Constraints ............................................................... 458
Source Role ............................................................................... 459
Target Role ............................................................................... 461
Role Tagged Values .................................................................... 462
Message Scope ........................................................................... 463
Requirements Management ....................................................... 464
Create Requirements ................................................................ 465
Requirement Elements ............................................................... 466
Color Code External Requirements ............................................ 468
Internal Requirements ............................................................... 469
Make Internal Requirement External ........................................... 470
Requirement Properties ............................................................. 472
Implementation .......................................................................... 473
Composition ............................................................................. 474
Requirements Hierarchy ............................................................. 475
Relationship Matrix .................................................................... 476
Open the Relationship Matrix ..................................................... 478
Set Element Type ....................................................................... 479
Set Connector Type and Direction .............................................. 480
Set Source and Target Package ................................................ 481
Relationship Matrix Options ..................................................... 482
Modify Relationships in Matrix ................................................ 483
Export to CSV ........................................................................... 484
## Contents

### UML Model Management

#### Part VI

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Architect Project Files</td>
<td>545</td>
</tr>
<tr>
<td>What is a Project?</td>
<td>547</td>
</tr>
<tr>
<td>Open a Project</td>
<td>549</td>
</tr>
<tr>
<td>Create a New Project</td>
<td>551</td>
</tr>
<tr>
<td>Model Wizard</td>
<td>552</td>
</tr>
<tr>
<td>Set Up a Database Repository</td>
<td>553</td>
</tr>
<tr>
<td>Upsize to Sybase ASA</td>
<td>554</td>
</tr>
<tr>
<td>Upsize to Progress OpenEdge</td>
<td>555</td>
</tr>
<tr>
<td>Upsize to MSDE</td>
<td>557</td>
</tr>
<tr>
<td>Upsize to PostgreSQL</td>
<td>557</td>
</tr>
<tr>
<td>Upsize to Oracle 9i, 10g or 11g</td>
<td>559</td>
</tr>
<tr>
<td>Upsize to SQL Server</td>
<td>560</td>
</tr>
<tr>
<td>Upsize to MySQL</td>
<td>562</td>
</tr>
<tr>
<td>Outputs</td>
<td>539</td>
</tr>
<tr>
<td>Events</td>
<td>537</td>
</tr>
<tr>
<td>Goals</td>
<td>539</td>
</tr>
<tr>
<td>A Complete Business Process</td>
<td>540</td>
</tr>
<tr>
<td>Use Profiles</td>
<td>490</td>
</tr>
<tr>
<td>Import a UML Profile</td>
<td>490</td>
</tr>
<tr>
<td>Tagged Values in Profiles</td>
<td>491</td>
</tr>
<tr>
<td>Add Profile Connector to Diagram</td>
<td>492</td>
</tr>
<tr>
<td>Synchronize Tags and Constraints</td>
<td>492</td>
</tr>
<tr>
<td>Profile References</td>
<td>494</td>
</tr>
<tr>
<td>Supported Types</td>
<td>494</td>
</tr>
<tr>
<td>Profile Structure</td>
<td>495</td>
</tr>
<tr>
<td>Attributes Supported in XML Profile</td>
<td>496</td>
</tr>
<tr>
<td>Example Profile</td>
<td>496</td>
</tr>
<tr>
<td>UML Patterns</td>
<td>507</td>
</tr>
<tr>
<td>Create a Pattern</td>
<td>508</td>
</tr>
<tr>
<td>Import a Pattern</td>
<td>511</td>
</tr>
<tr>
<td>Use a Pattern</td>
<td>512</td>
</tr>
<tr>
<td>MDG Technologies</td>
<td>514</td>
</tr>
<tr>
<td>Import MDG Technologies</td>
<td>515</td>
</tr>
<tr>
<td>Manage MDG Technologies</td>
<td>517</td>
</tr>
<tr>
<td>Access Remote MDG Technologies</td>
<td>518</td>
</tr>
<tr>
<td>Work with MDG Technologies</td>
<td>520</td>
</tr>
<tr>
<td>Mind Mapping</td>
<td>522</td>
</tr>
<tr>
<td>Data Flow Diagrams</td>
<td>524</td>
</tr>
<tr>
<td>ICONIX Process</td>
<td>526</td>
</tr>
<tr>
<td>BPMN 1.4</td>
<td>529</td>
</tr>
<tr>
<td>Change Element Appearance</td>
<td>531</td>
</tr>
<tr>
<td>Business Modeling</td>
<td>533</td>
</tr>
<tr>
<td>Process Modeling Notation</td>
<td>535</td>
</tr>
<tr>
<td>Inputs, Resources and Information</td>
<td>536</td>
</tr>
<tr>
<td>Events</td>
<td>537</td>
</tr>
<tr>
<td>Outputs</td>
<td>538</td>
</tr>
<tr>
<td>Goals</td>
<td>539</td>
</tr>
<tr>
<td>A Complete Business Process</td>
<td>540</td>
</tr>
</tbody>
</table>

---

Print the Matrix .................................................................................................................. 485
Matrix Profiles ..................................................................................................................... 486
Review Source and Target Elements ...................................................................................... 487
UML Profiles ........................................................................................................................... 488
Use Profiles ............................................................................................................................ 490
Import a UML Profile ............................................................................................................. 490
Tagged Values in Profiles ...................................................................................................... 491
Add Profile Connector to Diagram ......................................................................................... 492
Synchronize Tags and Constraints ............................................................................................. 492
Profile References .................................................................................................................. 494
Supported Types ...................................................................................................................... 494
Profile Structure ..................................................................................................................... 495
Attributes Supported in XML Profile ....................................................................................... 496
Example Profile .......................................................................................................................... 496
UML Stereotypes ......................................................................................................................... 499
Apply Stereotypes .................................................................................................................... 501
Stereotype Selector .................................................................................................................. 502
Stereotype Visibility ................................................................................................................. 503
Standard Stereotypes .............................................................................................................. 504
Stereotypes with Alternative Images ....................................................................................... 506
Profile References .................................................................................................................. 494
Supported Types ...................................................................................................................... 494
Profile Structure ..................................................................................................................... 495
Attributes Supported in XML Profile ....................................................................................... 496
Example Profile .......................................................................................................................... 496
UML Patterns ........................................................................................................................... 507
Create a Pattern ...................................................................................................................... 508
Import a Pattern ....................................................................................................................... 511
Use a Pattern ............................................................................................................................ 512
MDG Technologies ..................................................................................................................... 514
Import MDG Technologies ........................................................................................................ 515
Manage MDG Technologies ....................................................................................................... 517
Access Remote MDG Technologies ............................................................................................. 518
Work with MDG Technologies .................................................................................................. 520
Mind Mapping .......................................................................................................................... 522
Data Flow Diagrams .................................................................................................................. 524
ICONIX Process .......................................................................................................................... 526
BPMN 1.4 ..................................................................................................................................... 529
Change Element Appearance ....................................................................................................... 531
Business Modeling ..................................................................................................................... 533
Process Modeling Notation ....................................................................................................... 535
Inputs, Resources and Information ............................................................................................ 536
Events ......................................................................................................................................... 537
Outputs ........................................................................................................................................ 538
Goals .......................................................................................................................................... 539
A Complete Business Process ..................................................................................................... 540
Set Up an ODBC Driver ................................................................................................................. 563
MySQL ODBC Driver ..................................................................................................................... 564
PostgreSQL ODBC Driver ............................................................................................................ 566
ASA ODBC Driver .......................................................................................................................... 569
Progress OpenEdge ODBC Driver ............................................................................................... 573
Create a Repository ....................................................................................................................... 574
MySQL Repository ....................................................................................................................... 574
SQL Server Repository ................................................................................................................ 576
Oracle Server Repository ............................................................................................................. 578
PostgreSQL Repository ............................................................................................................... 579
Adaptive Server Anywhere Repository ......................................................................................... 581
MSDE Server Repository ............................................................................................................ 583
Progress OpenEdge Repository ................................................................................................... 583
Connect to a Data Repository ...................................................................................................... 584
MySQL Data Repository ............................................................................................................. 584
SQL Server Data Repository ...................................................................................................... 587
Oracle Data Repository ................................................................................................................ 589
PostgreSQL Data Repository ...................................................................................................... 591
ASA Data Repository .................................................................................................................. 593
MSDE Server Data Repository ................................................................................................... 595
Progress OpenEdge Repository ................................................................................................... 595
MySQL Limitations and SQL Scripts ........................................................................................... 598
Copy a Base Project ........................................................................................................................ 599
Upgrade Models ............................................................................................................................ 600
The Upgrade Wizard ....................................................................................................................... 601
Upgrade Replicas ........................................................................................................................... 602
Project Data Integrity ..................................................................................................................... 603
Check Project Data Integrity .......................................................................................................... 604
Run SQL Patches ........................................................................................................................... 606
Project Data Transfer ..................................................................................................................... 607
Perform a Project Data Transfer .................................................................................................... 608
Why Compare Projects? ................................................................................................................. 609
Compare Projects .......................................................................................................................... 610
Copy Packages Between Projects ................................................................................................. 611
Model Maintenance ......................................................................................................................... 613
Rename a Project .............................................................................................................................. 614
Compact a Project ........................................................................................................................... 615
Repair a Project ............................................................................................................................... 616
Manage Views ................................................................................................................................. 617
Add Views ....................................................................................................................................... 617
Rename Views ................................................................................................................................. 618
Delete Views ..................................................................................................................................... 619
Model Validation ........................................................................................................................... 620
Configure Model Validation .......................................................................................................... 622
Rules Reference ............................................................................................................................... 623
Well-Formedness ............................................................................................................................ 623
Element Composition ...................................................................................................................... 623
Property Validity .............................................................................................................................. 624
OCL Conformance .......................................................................................................................... 624
Model Sharing and Team Deployment .......................................................................................... 628
Share Enterprise Architect Projects .............................................................................................. 629
Share Projects on Network Drive .................................................................................................. 630
Distributed Development ............................................................................................................... 631
Replication ....................................................................................................................................... 632
Design Masters ............................................................................................................................... 633
Create Replicas ............................................................................................................................... 633
Contents XI

Version Control

- Use Version Control
  - Apply Version Control To Models .......................................................................................................................................................... 671
  - Version Control Basics .......................................................................................................................................................... 670

- XMI Import and Export
  - Export to XMI .......................................................................................................................................................... 638
  - Import from XMI .......................................................................................................................................................... 640
  - Import EMX/UML2 Files .......................................................................................................................................................... 642

- Limitedations of XMI .......................................................................................................................................................... 644

- The UML DTD .......................................................................................................................................................... 645

- Controlled Packages .......................................................................................................................................................... 646
  - Controlled Package Menu......................................................................................................................................................... 646
  - Configure Packages......................................................................................................................................................... 647
  - Remove Package from Control......................................................................................................................................................... 648
  - Save a Package......................................................................................................................................................... 649
  - Load a Package......................................................................................................................................................... 649
  - Batch XMI Export......................................................................................................................................................... 650
  - Batch XMI Import......................................................................................................................................................... 650
  - Manual Version Control with XMI......................................................................................................................................................... 651

CSV Import and Export

- CSV Specifications .......................................................................................................................................................... 654
- CSV Export .......................................................................................................................................................... 657
- CSV Import .......................................................................................................................................................... 659

- MOF

  - Getting Started .......................................................................................................................................................... 664
  - Export MOF to XMI ......................................................................................................................................................... 666

Version Control

- Version Control Basics .......................................................................................................................................................... 669

- Apply Version Control To Models ......................................................................................................................................................... 670

- Version Control & Team Deployment ......................................................................................................................................................... 671

- Version Control Menu .......................................................................................................................................................... 672

- Version Control Setup

  - Version Control Settings Dialog......................................................................................................................................................... 673
  - Version Control Nested Packages......................................................................................................................................................... 675
  - Version Control with SCC......................................................................................................................................................... 675
  - Upgrade at Enterprise Architect 4.5......................................................................................................................................................... 678
  - Version Control with CVS......................................................................................................................................................... 679
  - CVS with Remote Repositories......................................................................................................................................................... 679
  - CVS with Local Repositories......................................................................................................................................................... 683
  - Version Control with Subversion......................................................................................................................................................... 686
  - Set up Subversion......................................................................................................................................................... 686
  - Create a new Repository Sub-tree......................................................................................................................................................... 687
  - Create a Local Working Copy......................................................................................................................................................... 688
  - Subversion Under WINE-Crossover......................................................................................................................................................... 688
  - Version Control Configuration......................................................................................................................................................... 689
  - TortoiseSVN......................................................................................................................................................... 691
  - Version Control with TFS......................................................................................................................................................... 691

- Use Version Control

  - Package Version Control Menu......................................................................................................................................................... 695
  - Configure Controlled Package......................................................................................................................................................... 697
  - Use Existing Configuration......................................................................................................................................................... 698
  - Check In and Check Out Packages......................................................................................................................................................... 699
  - Include Other Users’ Packages......................................................................................................................................................... 700
  - Apply Version Control To Branches......................................................................................................................................................... 701
  - Export Controlled Model Branch......................................................................................................................................................... 702
  - Import Controlled Model Branch......................................................................................................................................................... 703
  - Review Package History......................................................................................................................................................... 704
Baselines, Differencing and Merges

User Security

Enable Security .......................................................................................................................................................... 709
Security Policy .......................................................................................................................................................... 710
Maintain Users .......................................................................................................................................................... 711
Assign User To Groups ............................................................................................................................................ 713
Set Up Single Permissions ....................................................................................................................................... 714
View All User Permissions ....................................................................................................................................... 715
Maintain Groups .......................................................................................................................................................... 716
Set Group Permissions ............................................................................................................................................... 717
List of Available Permissions ...................................................................................................................................... 718
View and Manage Locks ........................................................................................................................................... 720
Password Encryption .................................................................................................................................................. 721
Change Password ..................................................................................................................................................... 723
Lock Model Elements ............................................................................................................................................... 725
Add Connectors To Locked Elements ......................................................................................................................... 726
Lock Packages .......................................................................................................................................................... 727
Apply a User Lock ..................................................................................................................................................... 728
Locked Element Indicators .......................................................................................................................................... 729
Identify Who Has Locked An Object ......................................................................................................................... 730
Manage Your Own Locks ........................................................................................................................................... 731

Auditing ................................................................................................................................................................. 732

Auditing Quickstart ................................................................................................................................................... 733
Auditing Settings ....................................................................................................................................................... 734
Audit Scope............................................................................................................................................................. 734
Audit Logs............................................................................................................................................................... 735
Auditing Level.......................................................................................................................................................... 735
Audit Options.......................................................................................................................................................... 735

The Audit View ......................................................................................................................................................... 737
Audit View Controls .................................................................................................................................................. 738
Audit History Tab ....................................................................................................................................................... 742
Auditing Performance Issues ........................................................................................................................................ 743
Audit View Performance Issues .................................................................................................................................. 744

Baselines, Differencing and Merges ........................................................................................................................ 745

Baselines ................................................................................................................................................................. 746
Manage Baselines..................................................................................................................................................... 746
Create Baselines....................................................................................................................................................... 748

The Compare Utility (Diff) ........................................................................................................................................ 749

Compare Options ..................................................................................................................................................... 750

Example Comparison ................................................................................................................................................ 751
Compare Utility Tab Options ...................................................................................................................................... 752

Traceability ............................................................................................................................................................... 755

Packages and Elements ........................................................................................................................................... 757
Create Traceability Diagrams ...................................................................................................................................... 762
Traceability Tools ..................................................................................................................................................... 763

Reference Data .......................................................................................................................................................... 765

People ....................................................................................................................................................................... 766
Project Authors......................................................................................................................................................... 766
Project Roles.......................................................................................................................................................... 769
Project Resources................................................................................................................................................... 771
Project Clients........................................................................................................................................................ 772

General Types ......................................................................................................................................................... 774

Status Types............................................................................................................................................................ 774
Constraint Types....................................................................................................................................................... 776
Constraint Status Types............................................................................................................................................. 777
Part VII  License Management .......................................................... 794

Finding Your License Information ............................................................ 796
Add License Key ....................................................................................... 797
Keystore Troubleshooting ........................................................................... 799
Upgrade an Existing License ..................................................................... 800
Register Add-In .......................................................................................... 803

Part VIII  Project Management .............................................................. 805

Estimation .................................................................................................. 806
Technical Complexity Factors ...................................................................... 807
Environment Complexity Factors .............................................................. 809
Estimating Project Size .............................................................................. 811
Default Hours ............................................................................................ 813
Resource Management .............................................................................. 814
Resource Allocation .................................................................................... 815
Effort Management ..................................................................................... 816
Risk Management ....................................................................................... 817
Metrics ......................................................................................................... 818
Resource Report ........................................................................................ 819
Effort Types ................................................................................................. 820
Metric Types ............................................................................................... 821
Risk Types ................................................................................................. 822
Testing ......................................................................................................... 823
The Testing Workspace ............................................................................... 824
The Test Details Dialog .............................................................................. 825
Unit Testing ................................................................................................. 826
Integration Testing ...................................................................................... 827
System Testing ............................................................................................ 828
Acceptance Testing .................................................................................... 829
Scenario Testing ........................................................................................ 830
Import Scenario as Test ............................................................................. 831
Import Test From Other Elements ............................................................. 833
Testing Details Report ................................................................................. 834
Show Test Script Compartments ............................................................... 835
Test Documentation ................................................................................... 836
Maintenance ............................................................................................... 837
The Maintenance Workspace ...................................................................... 838
Maintenance Element Properties .............................................................. 839
Part IX Code Engineering ................................................. 867

Reverse Engineering .......................................................... 868
Import Source Code ......................................................... 870
Notes on Source Code Import ........................................... 872
Import a Directory Structure .............................................. 874
Import Binary Module ....................................................... 875
MDG Link and Code Engineering ......................................... 876
Classes Not Found During Import ........................................ 877
Synchronize Model and Code ............................................... 878
Generate Source Code .......................................................... 879
Generate a Single Class ......................................................... 881
Generate a Group of Classes .................................................. 883
Generate a Package ............................................................. 884
Update Package Contents .................................................... 886
Namespaces ........................................................................ 887

Code Engineering Settings .................................................. 888
Source Code Engineering ..................................................... 889
Source Code Options.......................................................... 889
Import Component Types ................................................... 890
Options - Code Editors ...................................................... 890
Options - Object Lifetimes .................................................. 891
Options - Attribute/Operations ............................................ 892
Code Page for Source Editing ............................................. 893
Local Paths ......................................................................... 895
Local Paths Dialog ............................................................. 896
Language Macros .............................................................. 897
Set Collection Classes ........................................................ 899
Language Options .............................................................. 901
ActionScript Options ......................................................... 901
C Options ............................................................................. 902
C# Options ........................................................................... 902
C++ Options ......................................................................... 903
Part X Visual Execution Analyzer

Setup for Build and Run ................................................................................................................................. 941

Managing Scripts .............................................................................................................................................. 943
Build Script Dialog .......................................................................................................................................... 945
Build Command ................................................................................................................................................ 946
Test Command .................................................................................................................................................. 947
Run Command ................................................................................................................................................... 948
Debug Command .............................................................................................................................................. 950

Java .................................................................................................................................................................... 951

Attach to Virtual Machine .............................................................................................................................. 952
.NET ................................................................................................................................................................. 954
Debug Assemblies .......................................................................................................................................... 954
Debug - CLR Versions ................................................................................................................................... 955
Microsoft Native .............................................................................................................................................. 956
Recursive Builds ............................................................................................................................................. 956

Deploy Command ........................................................................................................................................... 958

Sequence Options .......................................................................................................................................... 959

Enable Filter ...................................................................................................................................................... 959
Record Arguments To Function Calls ........................................................................................................... 961
Record Calls To External Modules ................................................................................................................ 961

Debugging ....................................................................................................................................................... 963

System Requirements .................................................................................................................................... 964
Debug ASP .NET .............................................................................................................................................. 965
Debug COM interop ....................................................................................................................................... 968
Part XII  Data Modeling

Import WSDL ............................................................................................................................... 1037

Part XII  Data Modeling  1039

A Data Model Diagram .................................................................................................................. 1041
Create a Table ............................................................................................................................... 1042
Set Table Properties .................................................................................................................... 1043
  Set Table Owner ......................................................................................................................... 1045
  Set MySQL Options .................................................................................................................. 1046
  Set Oracle Table Properties ..................................................................................................... 1047
Create Columns ........................................................................................................................... 1049
Create Oracle Packages .............................................................................................................. 1051
Primary Key ................................................................................................................................. 1052
  SQL Server Non Clustered Keys ............................................................................................... 1054
Foreign Key .................................................................................................................................. 1055
  Create Foreign Key .................................................................................................................... 1056
  Define Foreign Key Name Template .......................................................................................... 1060
Stored Procedures ......................................................................................................................... 1061
  Create Container Class Procedure ........................................................................................... 1062
  Create Individual Class Procedure .......................................................................................... 1065
Views ............................................................................................................................................ 1067
Index, Trigger, Check Constraint .................................................................................................. 1070
Generate DDL For a Table .............................................................................................................. 1072
Generate DDL for a Package .......................................................................................................... 1074
Data Type Conversion Procedure ............................................................................................... 1078
Data Type Conversion for a Package ........................................................................................... 1079
DBMS Datatypes ............................................................................................................................ 1081
Import Database Schema from ODBC ......................................................................................... 1083
  Select a Data Source ................................................................................................................... 1086
  Select Tables .............................................................................................................................. 1087
  The Imported Class Elements .................................................................................................. 1088

Part XIII  MDA Transformations

Transform Elements ....................................................................................................................... 1093
Chaining Transformations .............................................................................................................. 1095
Import Transformations ................................................................................................................. 1096
Transformation Templates ............................................................................................................ 1097
Built-in Transformations ................................................................................................................. 1099
  C# Transformation ..................................................................................................................... 1100
  DDL Transformation ................................................................................................................ 1102
  EJB Transformations ................................................................................................................ 1105
  Java Transformation ................................................................................................................ 1107
  JUnit Transformation ................................................................................................................ 1109
  NUnit Transformation ............................................................................................................... 1111
  WSDL Transformation ............................................................................................................. 1113
  XSD Transformation ................................................................................................................. 1114
Write Transformations .................................................................................................................. 1117
  Default Transformation Templates ............................................................................................ 1118
  Intermediary Language ............................................................................................................. 1119
  Objects ..................................................................................................................................... 1120
  Connectors ............................................................................................................................... 1125
Part XIV  Enterprise Architect Reports  1132

RTF Documents ................................................................. 1133

Generate RTF Documents .......................................................... 1135
Diagram Options................................................................. 1136
Exclude Package from Report................................................... 1137
Generate RTF Documentation Dialog....................................... 1137
RTF Templates Tab............................................................... 1139
  RTF Style Template Editor.................................................. 1141
Select Components for Reporting........................................... 1142
Add Content........................................................................ 1144
Tabular Sections................................................................... 1145
Child Sections...................................................................... 1146
RTF Style Template Editor Options......................................... 1148
Scroll Through Text............................................................ 1148
File and Print Options.......................................................... 1149
Cut and Paste Options.......................................................... 1150
View Options....................................................................... 1151
Image and Object Inserts....................................................... 1152
Character Formatting............................................................ 1153
Paragraph Formatting........................................................... 1154
Tab Support.......................................................................... 1155
Page Breaks and Repagination................................................. 1155
Headers and Footers.............................................................. 1156
Hyperlinks and Bookmarks.................................................... 1157
Table Commands................................................................. 1158
Sections and Columns............................................................ 1160
Stylesheets and Table of Contents.......................................... 1160
User-Defined Section Numbering.......................................... 1161
Frames and Drawing Objects.................................................. 1164
Search and Replace Commands............................................... 1165
Import RTF Template........................................................... 1165
Resource Documents............................................................ 1166
Document Options............................................................... 1167
Word Substitution................................................................. 1171
Language Substitution.......................................................... 1171

The Legacy RTF Report Generator ........................................... 1173
Document a Single Element.................................................... 1174
The RTF Report Dialog........................................................ 1174
Set the Main RTF Properties.................................................. 1175
Apply a Filter ...................................................................... 1175
Exclude Elements............................................................... 1176
RTF Diagram Format............................................................ 1176
Project Include.................................................................... 1177
RTF Report Options.............................................................. 1177
RTF Report Selections........................................................... 1178
Generate the Report............................................................... 1179
Legacy RTF Style Templates................................................. 1179
Save as Document............................................................... 1181
Custom Language Settings.................................................... 1181

Use MS Word ...................................................................... 1183
Open a Report in Microsoft Word........................................... 1183
Change Linked Images to Embedded....................................... 1183
RTF Bookmarks................................................................... 1184
Part XV UML Dictionary

UML Diagrams .................................................................................................................. 1212

Behavioral Diagrams ........................................................................................................ 1213
Activity Diagram ............................................................................................................. 1213
Use Case Diagram .......................................................................................................... 1215
State Machine Diagram .................................................................................................. 1216
Regions ............................................................................................................................. 1219
Pseudo-States .................................................................................................................. 1220
State Machine Table ....................................................................................................... 1220
State Machine Table Options ......................................................................................... 1222
State Machine Table Operations .................................................................................... 1224
Change State Machine Table Position ......................................................................... 1225
Change State Machine Table Size ................................................................................ 1225
Insert New State ............................................................................................................ 1225
Insert Trigger ................................................................................................................. 1226
Insert/Change Transition .............................................................................................. 1226
Reposition State or Trigger Cells ................................................................................. 1227
Add Legend ...................................................................................................................... 1227
Find Cell in State Machine Diagram ............................................................................. 1227
State Machine Table Conventions ................................................................................. 1227
Export State Table To CSV File .................................................................................... 1228
Timing Diagram .............................................................................................................. 1228
Create a Timing Diagram ............................................................................................... 1230
Set a Time Range ............................................................................................................ 1230
Edit a Timing Diagram ................................................................................................... 1230
Add and Edit State Lifeline ............................................................................................. 1231
Edit States In State Lifeline ............................................................................................ 1231
Edit Transitions In State Lifeline ................................................................................... 1232
Add and Edit Value Lifeline ............................................................................................ 1234
Add States In Value Lifeline ........................................................................................... 1234
Edit Transitions In Value Lifeline .................................................................................. 1234
Configure Timeline - States ......................................................................................... 1236
Configure Timeline - Transitions .................................................................................. 1237

Other Documents .......................................................................................................... 1192

Dependency Report .......................................................................................................... 1192
Diagram Only Report ...................................................................................................... 1193
Implementation Report .................................................................................................. 1193
Set Target Types Dialog ............................................................................................... 1195
Testing Report ................................................................................................................ 1195

Virtual Documents ....................................................................................................... 1196

Create Master Document .............................................................................................. 1197
Create Model Document ............................................................................................... 1199
Add Packages to Model Document .............................................................................. 1200
Delete Package in Model Document ............................................................................. 1200
Document Order ............................................................................................................ 1201
Generate the Document ................................................................................................. 1202

HTML Reports .............................................................................................................. 1204

Create an HTML Report ................................................................................................ 1205
The Generate HTML Report Dialog ............................................................................. 1206

Web Style Templates ..................................................................................................... 1207
## Contents

- Final: .......................... 1305
- Flow Final: .......................... 1306
- Fork/Join: .......................... 1307
  - Fork: .......................... 1309
  - Join: .......................... 1310
- History: .......................... 1311
- Initial: .......................... 1312
- Interaction Occurrence: .......................... 1313
- Interruptible Activity Region: .......................... 1313
  - Add Interruptible Activity Region: .......................... 1314
- Junction: .......................... 1314
- Lifeline: .......................... 1315
- Merge: .......................... 1316
- Message Endpoint: .......................... 1316
- Message Label: .......................... 1317
- Note: .......................... 1317
- Partition: .......................... 1318
- Receive: .......................... 1319
- Region: .......................... 1320
- Send: .......................... 1320
- State: .......................... 1321
  - Composite State: .......................... 1322
  - State Lifeline: .......................... 1323
  - State/Continuation: .......................... 1324
  - Continuation: .......................... 1324
  - State Invariant: .......................... 1326
- Structured Activity: .......................... 1326
- State Machine: .......................... 1328
- Synch: .......................... 1329
- System Boundary: .......................... 1329
- Terminate: .......................... 1330
- Trigger: .......................... 1330
- Use Case: .......................... 1331
  - Use Case Extension Points: .......................... 1332
  - Rectangle Notation: .......................... 1333
- Value Lifeline: .......................... 1333
- **Structural Diagram Elements** .......................... 1335
- Artifact: .......................... 1336
- Class: .......................... 1337
  - Active Classes: .......................... 1338
  - Parameterized Classes (Templates): .......................... 1338
- Collaboration: .......................... 1340
- Collaboration Occurrence: .......................... 1341
- Component: .......................... 1342
- Deployment Spec: .......................... 1343
  - Device: .......................... 1344
  - Document Artifact: .......................... 1344
- Enumeration: .......................... 1344
- Execution Environment: .......................... 1345
- Expose Interface: .......................... 1345
- Information Item: .......................... 1346
- Interface: .......................... 1347
- Node: .......................... 1348
- Object: .......................... 1348
  - Run-time State: .......................... 1349
    - Define a Run-time Variable: .......................... 1349
    - Remove a Defined Variable: .......................... 1350
    - Define an Object State: .......................... 1350
Part XVI  SDK for Enterprise Architect  1427

Developing Profiles ................................................................. 1428

Custom Stereotypes ........................................................................ 1429
Create Profiles ............................................................................. 1431
Create a Profile Package................................................................. 1431
Add Stereotypes and Metaclasses................................................... 1431
Define Stereotype Tags................................................................. 1433
With Predefined Tag Types......................................................... 1433
With Supported Attributes......................................................... 1434
Use the Tagged Value Connector................................................ 1435
Define Stereotype Constraints.................................................... 1436
Add Enumeration Elements....................................................... 1438
Add Shape Scripts....................................................................... 1439
Set Default Appearance.............................................................. 1441
Export a UML Profile................................................................. 1441
Save Profile Options.................................................................... 1442
Supported Attributes.................................................................... 1442
Define a Stereotype as a Metatype............................................... 1443
Define Multiple-Stereotype Level................................................ 1444
Define Creation of Instance......................................................... 1445
Create Composite Elements....................................................... 1445
Define Child Diagram Types....................................................... 1446
Stereotypes Profiles..................................................................... 1447
Quick Linker.................................................................................. 1448
Quick Linker Definition Format.................................................. 1448
Quick Linker Example................................................................. 1450
Hide Default Quick Linker Settings............................................ 1451
Quick Linker Object Names......................................................... 1451

MDG Technologies in SDK ......................................................... 1453

Create MDG Technologies........................................................ 1454
Add a Profile............................................................................... 1454
Add a Pattern.............................................................................. 1457
Add Tagged Values................................................................. 1458
Add Code Modules................................................................. 1459
Custom Templates.......................................................................................................................................................... 1527
Override Default Templates............................................................................................................................................... 1528
Add New Stereotyped Templates .................................................................................................................................. 1529
Create Custom Language Template.................................................................................................................................. 1529

Enterprise Architect Add-In Model .................................................................................................................................. 1531

Add-In Tasks ........................................................................................................................................................................ 1532
Create Add-Ins................................................................................................................................................................. 1532
Define Menu Items.............................................................................................................................................................. 1532
Deploy Add-Ins..................................................................................................................................................................... 1533
Tricks and Traps..................................................................................................................................................................... 1534

The Add-In Manager .......................................................................................................................................................... 1537

Add-In Search ................................................................................................................................................................. 1538
XML Format (Search Data).................................................................................................................................................. 1538

Add-In Events ................................................................................................................................................................. 1540
EA_Connect........................................................................................................................................................................... 1540
EA_Disconnect........................................................................................................................................................................ 1540
EA_GetMenuItems................................................................................................................................................................. 1541
EA_GetMenuState................................................................................................................................................................. 1541
EA_MenuClick........................................................................................................................................................................ 1542
EA_OnOutputItemClick.......................................................................................................................................................... 1543
EA_OnOutputItemDoubleClicked................................................................. 1543
EA_ShowHelp......................................................................................................................................................................... 1544

Broadcast Events ............................................................................................................................................................ 1545
EA_FileOpen........................................................................................................................................................................... 1545
EA_FileClose............................................................................................................................................................................ 1545
Pre-Deletion Events ............................................................................................................................................................ 1546
EA_OnPreDeleteElement.................................................................................................................................................... 1546
EA_OnPreDeleteConnector................................................................................................................................................ 1546
EA_OnPreDeleteDiagram................................................................................................................................................ 1547
EA_OnPreDeletePackage.................................................................................................................................................. 1547

Pre-New Events ................................................................................................................................................................. 1548
EA_OnPreNewElement........................................................................................................................................................ 1548
EA_OnPreNewConnector.................................................................................................................................................. 1549
EA_OnPreNewDiagram.................................................................................................................................................... 1549
EA_OnPreNewAttribute.................................................................................................................................................... 1550
EA_OnPreNewMethod.................................................................................................................................................... 1551
EA_OnPreNewPackage................................................................................................................................................... 1551

Post-New Events............................................................................................................................................................... 1552
EA_OnPostNewElement.................................................................................................................................................... 1552
EA_OnPostNewConnector................................................................................................................................................ 1553
EA_OnPostNewDiagram................................................................................................................................................ 1553
EA_OnPostNewAttribute................................................................................................................................................ 1554
EA_OnPostNewMethod................................................................................................................................................... 1554
EA_OnPostNewPackage.................................................................................................................................................. 1555

Technology Events ........................................................................................................................................................... 1555
EA_OnPreDeleteTechnology............................................................................................................................................. 1555
EA_OnDeleteTechnology.................................................................................................................................................. 1556
EA_OnImportTechnology.................................................................................................................................................. 1556

Context Item Events ........................................................................................................................................................ 1557
EA_OnContextItemChanged.............................................................................................................................................. 1557
EA_OnContextItemDoubleClicked.................................................................................................................................. 1558
EA_OnNotifyContextItemModified.................................................................................................................................. 1558

EA_OnPostExecute............................................................................................................................................................... 1559

Compartment Events........................................................................................................................................................ 1560
EA_QueryAvailableCompartments...................................................................................................................................... 1560
EA_GetCompartmentData.................................................................................................................................................. 1560

Model Validation Broadcasts............................................................................................................................................ 1562

Enterprise Architect User Guide
Enterprise Architect Object Model

Using the Automation Interface

Connect to the Interface

Set References In Visual Basic

Examples and Tips

Call from Enterprise Architect

Available Resources

Reference

Interface Overview

App

Enumerations

ConstLayoutStyles Enum

EnumRelationSetType Enum

MDGMenus Enum

ObjectType Enum

PropType Enum

ReloadeType Enum

XMLType Enum

Repository

Repository

Author

Client

Collection

Datatype

EventProperties

EventProperty

ModelError

Package
Part XVII  Glossary of Terms  1678

A ......................................................................................................................... 1679
B ......................................................................................................................... 1681
C ......................................................................................................................... 1682
D ......................................................................................................................... 1685
E ......................................................................................................................... 1687
F ......................................................................................................................... 1688
G ......................................................................................................................... 1689
H ......................................................................................................................... 1690
I ......................................................................................................................... 1691
J ......................................................................................................................... 1693
L ......................................................................................................................... 1694
M ......................................................................................................................... 1695
N ......................................................................................................................... 1697
O ......................................................................................................................... 1698
P ......................................................................................................................... 1699
Q ......................................................................................................................... 1702
R ......................................................................................................................... 1703
S ......................................................................................................................... 1705
T ......................................................................................................................... 1708
U ......................................................................................................................... 1710
V ......................................................................................................................... 1711

Index  1712
This user guide provides an introduction to the features contained in Enterprise Architect - a UML CASE tool for developing and building software systems with UML.
Part I
Welcome to Sparx Systems’ Enterprise Architect, a UML 2.1 based modeling tool for designing and constructing software systems, for business process modeling, and for generalized modeling purposes such as visualizing existing systems and processes.

This user guide provides tutorials, guidance and reference material to help you use Enterprise Architect to perform:

- UML Model Management
- Modeling With Enterprise Architect
- RTF and HTML Document Creation (Reports)
- Project Management
- Code Engineering
- Visualization of code execution (debugging)
- Data Modeling
- MDA Transformations
- XML Technology (SOA) Support

See What is Enterprise Architect? for further details of what Enterprise Architect can do and what you can use it for.

Enterprise Architect makes extensive use of UML, so we provide a UML Dictionary of diagrams, elements and connectors. Enterprise Architect also includes a Software Developers’ Kit (SDK for Enterprise Architect) that enables you to extend and customize the tool.

To Use This Guide

If you are new to modeling and UML as well as Enterprise Architect, or otherwise want a rapid review of the process of modeling with Enterprise Architect, go to the Getting Started with UML Modeling tutorial. This is not just a theoretical description - the first things you do are start Enterprise Architect and immediately create a model project.

Enterprise Architect is very flexible and has lots of features. When working through Getting Started, you will see many links to more extensive descriptions of features, functions, tasks and procedures, in the Using Enterprise Architect section. You can follow any of these immediately if you require more information.

The Using Enterprise Architect section is the first of the main references for working with Enterprise Architect. The documentation includes further sections for each of the subject areas listed at the start of this topic.

You should read the Sparx Systems Formal Statements, including the Copyright Notice and our End User Licensing Agreement.

You can also check the Glossary for definitions of various terms and concepts used in the Enterprise Architect User Guide.

Your Feedback

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

You can access our online feedback pages at:

- www.sparxsystems.com/bug_report.htm
- www.sparxsystems.com/feature_request.htm
Alternatively, you can contact Sparx Systems by email at: support@sparsystems.com.
1.1 What is Enterprise Architect?

Powerful UML Analysis and Design Tool
Enterprise Architect is a comprehensive UML analysis and design tool, covering all aspects of the software development cycle from requirements gathering, through analysis, model design, testing, change control and maintenance to implementation, with full traceability. Enterprise Architect combines the power of the latest UML 2.1 specification with a high performance, intuitive interface, to bring advanced modeling to the whole development team. It is a multi-user, visual tool with a great feature set (see below), helping analysts, testers, project managers, quality control staff and deployment staff around the world to build and document robust, maintainable systems and processes.

The UML Modeling Tool of Choice, Globally
With over 150,000 licenses sold, Enterprise Architect has proven highly popular across a wide range of industries and is used by thousands of companies world-wide. From large, well-known, multi-national organizations to smaller independent companies and consultants, Enterprise Architect has become the UML modeling tool of choice for developers, consultants and analysts in over 60 countries.

Sparx Systems software is used in the development of many kinds of application and system in a wide range of industries, including: aerospace, banking, web development, engineering, finance, medicine, military, research, academia, transport, retail, utilities (such as gas and electricity) and electrical engineering. It is also used effectively for UML and enterprise architecture training in many prominent colleges, training companies and universities around the world.

Now see:
- Uses of Enterprise Architect
- Enterprise Architect Key Features
1.1.1 Uses of Enterprise Architect

Enterprise Architect is a powerful tool for specifying, documenting and building your software and business process projects. Using Enterprise Architect’s support for UML and its related standards, you can model new complex software and business systems, or visualize and maintain existing systems.

Design and Build Diverse Systems Using UML

UML 2.1 is an open standard that provides a rich language for describing, documenting and designing software, business and IT systems in general. Enterprise Architect enables you to leverage the full expressive power of UML 2.1 to model, design and build diverse systems in an open and well understood manner. You can generate code, database structures, documentation and metrics; transform models; or specify behavior and structure as the basis for contractual agreements.

Model and Manage Complexity

Enterprise Architect helps individuals, groups and large organizations model and manage complex information. Often this relates to software development and IT systems design and deployment, but it can also relate to business analysis and business process modeling. Enterprise Architect integrates and connects a wide range of structural and behavioral information, helping to build a coherent and verifiable architectural model, either what-is or what-will-be. Tools to manage version control, track and compare differences, audit changes and enforce security help control project development and enforce compliance to standards.

Share Models

Enterprise Architect enables you to share complete models or specific aspects of a model between members of a team, including (through the ‘Lite’, read-only edition) stakeholders who can study a model but not change or manage it. You can make the project .EAP file available on a shared network drive or replicate the .EAP file for complex distributed development. Alternatively, you can develop the project in one of several shared DBMS repositories, such as SQL Server, MySQL, PostgresSQL, Oracle 9i, 10g or 11g, and Sybase ASA. You can import and export data as XML files to distribute and update frameworks and other package-based model structures. You control changes through the version control repository. Enterprise Architect provides a data transfer wizard that enables you to upsize or downsize the complete model for maximum flexibility, and it enables you to export and import reference data so that you do not have to recreate it for related projects.

Model, Manage and Trace Requirements

Enterprise Architect enables you to capture requirements and use full traceability from base requirements to design, build, deployment and beyond. You can use impact analysis to trace from proposed changes to original requirements, and build the ‘right’ system.

Develop Personal Views and Extracts of the Model

Enterprise Architect enables you to develop any number of different views of your model, or parts of it, either for your personal use or for the use of your team. These Model Views are generated by reports, so they can be set up to always show the current status of the selected view. The facility also enables you to create Favorites folders of hyperlinks to frequently-used data structures.

Track and Trace Model Structures

In even a small model, it can be difficult to locate packages, diagrams and elements, even if you apply a rigorous naming and structure policy. Enterprise Architect has a wealth of facilities that enable you to locate structures quickly and easily, through the Model Search, Element List, Auditing facility, Relationship Matrix and reports. The Element menu, Diagram menu and Project Browser context menus also enable you to locate elements in diagrams and in the Project Browser, and you can store hyperlinks to important or commonly-used elements and diagrams in the Model Views. Finally, having located one element you can import any related elements into a diagram in a single operation.

Generate Documentation

Enterprise Architect provides powerful document generation and reporting tools with a full WYSIWYG template editor for RTF or HTML output. You can generate complex and detailed reports from Enterprise Architect with the information you require in the format your company or client demands.
Generate and Reverse Engineer Source Code

Enterprise Architect supports generation and reverse engineering of source code for many popular languages, including C++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript, Python and PHP. With a built-in 'syntax highlighting' source code editor, Enterprise Architect enables you to quickly navigate and explore your model source code in the same environment. Code generation templates enable you to customize the generated source code to your company specifications.

Visualize, Inspect and Understand Complex Software

Software is complex and often hard to understand. You can use Enterprise Architect to reverse engineer code in a wide range of software development languages and database repository schemas, to understand static structure. To complete the picture, use the unique built-in profiling and debugging tools to capture and visualize executing software at run-time. Create run-time instances of model elements and invoke methods using the built-in Object Workbench.

Perform MDA Transformations

Model Driven Architecture (MDA) is an open standard designed to facilitate rapid application development in a platform independent manner. Models can be built at a high level of abstraction and, using MDA based tools, transformed into models and code targeting a specific platform or domain.

Model Databases

Enterprise Architect enables you to reverse engineer from many popular DBMS systems, including Oracle 9i, 10g or 11g; SQL Server; My SQL; Access and PostgreSQL. You can model database tables, columns, keys, foreign keys and complex relationships using UML and an inbuilt data modeling profile, and forward generate DDL scripts to create target database structures.

Customize Enterprise Architect

Enterprise Architect also includes a Software Developers' Kit that enables experienced tool developers to customize and extend Enterprise Architect to suit the specific requirements of their organization with, for example, in-house UML Profiles, Add-Ins and Code Templates. The very detailed Automation Interface gives you access to most element features, major functions such as XMI import/export, and attached information. Most properties are fully writable from the automation client. The Automation Interface provides great support for plug-ins, with the ability to embed automation client windows in the main diagram view. The Interface is accessible from any automation-aware client language, such as VB, C#, C++ and Delphi.

Link Enterprise Architect to IDEs

Using Sparx Systems Model Driven Generation (MDG) Link plug-ins, you can develop source code in your preferred Integrated Development Environment such as Visual Studio .NET or Eclipse, use Enterprise Architect to locate the source code for Classes, attributes and operations, and to model, navigate, track, reverse engineer, build and run your project.

The MDG Integration products for Eclipse and Visual Studio 2005 provide an even closer, seamless integration of Enterprise Architect and UML 2.1 with your IDE, bringing the functionality required of a fully fledged modeling platform right inside the IDE.
1.1.2 Key Features

Enterprise Architect is renowned for its rich feature set. Some of the key features are highlighted in the following list:

- Model complex information, software and hardware systems using UML-compliant notation (comprehensive UML 2.1 support for all 13 UML diagrams)
- Extended modeling for Requirements, User Interface Design, Mind Mapping, Data Modeling and more
- Built-in Requirements Management enables you to specify, trace and verify requirements directly against the design, right through to the deployed solution
- Comprehensive and flexible MS Word-compatible HTML and RTF report options
- Leverage industry-standard Enterprise Architecture frameworks (Zachman, TOGAF, DoDAF-MODAF)
- Support in forward and reverse code engineering for many languages ‘out of the box’: ActionScript 3.0, Java, C#, C++, VB.Net, Delphi, Visual Basic, Python and PHP
- Ability to perform database modeling, to reverse engineer from a range of DBMSs via ODBC, and to forward generate DDL scripts to create database structures
- Manage, track and control change using baseline model merge and auditing capabilities
- Centralize enterprise-wide documentation of processes and information systems
- Model dependencies between elements, system dynamics and state
- Model class hierarchies, deployment, components and implementation details
- Record project issues, tasks and system glossary
- Assign resources to model elements and track effort expended against required effort
- Testing support for test cases, JUnit and NUnit
- Integrated Debug Workbench for profiling executable Java and .Net applications, instantiating run-time model objects and generating Sequence diagrams from a stack trace
- Manage Version control using any Common Source Code Control (SCC)-compliant tool and XMI
- Inbuilt user and group security and access control management
- Distributed development through shareable files, use of shared repositories in a range of major Database Management Systems, file replication, data transfer, and import and export of reference data
- Share models using the latest XMI 2.1 format
- Import models from other tools in XMI format
- Built-in Model Driven Architecture (MDA) Transformations, and facilities to import or create others
- Facilities to import database schema, XSD and WSDL source, .NET and Java binaries
- Software Developers’ Kit for scripting and customizing Enterprise Architect
- A range of internal and external commercial MDG Add-Ins to integrate the facilities of Enterprise Architect with IDEs and other technologies, and templates to write your own
- Read-only Viewer enables stakeholders to view but not change milestone deliverables
- Price: Enterprise Architect is priced to outfit the entire team, making collaboration and team development a real possibility
- Speed: Enterprise Architect is quick to load and a spectacularly fast performer, even with large models
- Scalability: Enterprise Architect supports single users and the development of small models, or many concurrent users developing extremely large models, with equal ease
- Usability: many of our users agree, Enterprise Architect gets you started and productive quickly, with a rich user interface and the ability to create patterns, templates, model views and ‘favorites’ collections of commonly-used elements and diagrams

For a complete list of the new features of the latest version of Enterprise Architect, click on the Help | Read Me menu option.

Enterprise Architect is available in three editions: Corporate, Professional and Desktop, each of which offers a different range of features. For a comparison of the Enterprise Architect editions, see the Differences Between Editions topic.
1.1.3 Differences Between Editions

Enterprise Architect is available in three editions: Corporate, Professional and Desktop. Functionality for each version is described below:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>.EAP Project Files</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Advanced UML 2.1 Modeling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Business Process Modeling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shared Models</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Shared/Floating License Version</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Automation API &amp; Scripting</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Source Code Engineering</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Database Engineering</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>WSDL Engineering</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>XML Schema/XSD Engineering</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Reverse Engineer Binaries (Java, .NET)</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Microsoft Access Repository</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SQL Server; MySQL; Oracle 9i, 10g and 11g; PostgreSQL; MSDE; Adaptive Server Anywhere Database Repositories</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Version Control</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Replication</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Profile/Metamodel Extensibility</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MDG Technologies (Create and Use)</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>MDG Link for Eclipse and MDG Link for Visual Studio.NET</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Security (Role Based)</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shape Script Customization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Test Management</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Auditing of model changes</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Baselines</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Compare (Diff) Utility</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Relationship / Traceability Matrix</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Functionality

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Management</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Element List (Tabular Editing)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Metadata/Repository Search</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Project Discussion Forum</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Project Data Transfer</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XMI Import and Export, Version 2.1, 1.2, 1.1, 1.0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MDA-Style Transformations</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Visualization (Debug) Of Applications</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Document Generation - RTF &amp; HTML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Report Customization - WYSIWYG Rich-text Templates</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>State Chart Editor</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Link RTF Documents To UML Elements</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### Enterprise Architect 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, 10g or 11g DBMS back ends as the shared repository. This provides additional scalability and improved concurrency over the shared .EAP file approach to model sharing. User security, user logins, user groups and user level locking of elements, user/group based security (with locking at diagram and element levels) are also supported. Security comes in two modes: in the first mode, all elements are considered 'readable' until explicitly locked by a user or group; in the second mode, all elements are considered locked until checked out with a user lock.

The Corporate edition is available in either standalone (fixed license) or Floating License form. The Corporate Floating License arrangement is particularly useful for companies that manage a central store of license keys. Floating license keys can be used by different employees over time, temporarily or permanently.

### Enterprise Architect 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 Enterprise Architect projects and extracting information in XMI format. The Professional edition fully supports code import/export and synchronization of model elements with source code. It enables reverse engineering SQL Server, MS Access and Oracle 9i, 10g or 11g databases. Support for MDG Technologies and MDG Link (sold separately) is included with the Professional version of Enterprise Architect. The shared repository available in the Professional edition is restricted to the .EAP file format (JET database).

### Enterprise Architect Desktop Edition

The Desktop edition is targeted at single developers producing UML analysis and design models.
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 required configuration. When Enterprise Architect starts, select the mode to trial; you can close down Enterprise Architect and restart it in another mode for comparison.

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

More information about Enterprise Architect editions is available on the Sparx Systems website.
1.1.4 Sparx Systems MDG Add-Ins

Enterprise Architect is the core for a range of Model Driven Generation (MDG) Add-Ins that enable you to extend its modeling capabilities to use more specialized, niche frameworks and profiles. Some of these, such as ICONIX, BPMN, Data Flow Diagrams and Mind Mapping, are already provided with the Enterprise Architect installer.

Enterprise Architect provides support for downloading MDG Technologies from external system files or websites, or for creating your own easily with the Enterprise Architect MDG Technology Wizard.

Sparx Systems also market a number of MDG products, as follows:

- **MDG Technology For:**
  - Zachman Framework
  - The Open Group Architecture Framework (TOGAF)
  - Department Of Defense Architecture Framework - Ministry Of Defence Architecture Framework (DoDAF-MODAF)
  - Data Distribution Service (DDS)
  - Systems Modeling Languages (SysML)
  - Python (for Enterprise Architect versions 4.5 to 5.0, integrated in later versions) (* free product! *)
  - CORBA (* free product! *)
  - Java Beans (* free product! *)
  - Testing (* free product! *)

- **MDG Integration For:**
  - Eclipse 3.3
  - Visual Studio 2005 and 2008
  - Siemens PLM Teamcenter Systems Engineering (TcSE)

- **MDG Link For**
  - Eclipse
  - Visual Studio.Net
  - Microsoft Visio (* free product! *)
  - Telelogic DOORS

Over time, this list is being extended to include further products.

**Product Information**

For the latest list of available Add-Ins and an introduction to each product, including details of pricing, purchasing and download options, see the Products Page on the Sparx Systems website. When you purchase one of the Add-Ins, you receive one or more license keys and instructions on obtaining, installing and registering the product.

The information page for most products provides a link to download the product User Guide in .pdf format.

The product User Guide can also be displayed as a .chm file online within the product itself. To access this online help in Enterprise Architect, select the Add-Ins | <productname> | Help menu option.
1.2 Help and Support

Enterprise Architect has three main help and information systems to assist you in using the product:

- **Tasks Pane**
- **Enterprise Architect Help**
- The Sparx Systems website.

In addition Sparx Systems recommend that you fully explore the sample project supplied with Enterprise Architect. It assists you in learning to use Enterprise Architect and offers tips on getting the most out of Enterprise Architect's features. Click on the **EAExample** option on the Enterprise Architect **Start Page** to answer any queries or problems.

If you have purchased Enterprise Architect and are a registered user, you can also contact **Sparx Support** to answer any queries or problems.

**Tasks Pane**

The Enterprise Architect **Tasks Pane** provides context-sensitive guidance, tools, demonstrations and other online resources to help you understand any area of Enterprise Architect that you are interested in. The **Tasks Pane** automatically displays on the right of the screen when you first open Enterprise Architect, showing the **Getting Started** topics. You can select other task areas by clicking on the **More tasks** option in the toolbar.

**Enterprise Architect Help**

Enterprise Architect Help provides comprehensive documentation of Enterprise Architect and covers every aspect and facility of the product. To access Help within Enterprise Architect:

- Click on the Help icon in the various toolbars
- Select the **Help | Help Contents** menu option
- Click on the **Help** button on a dialog (for Help specific to that dialog).

Enterprise Architect Help is extensive; if you cannot quickly locate the topic you require in the online contents list, you can use one of two search facilities:

- Click on the **Index** tab, type in a keyword or key phrase appropriate to the subject you require help for, and press **Enter**; double-click on the appropriate index item
- Click on the **Search** tab, type in a word or phrase to search for, and click on the **List Topics** button; double-click on the required topic.

The Enterprise Architect Help is also available separately from the product, in different formats. See the **Available Helpfile Formats** topic.

**Sparx Systems Website**

The Sparx Systems website is also extensive, and provides information and announcements concerning the company and its full range of products, as well as tutorials, white papers, templates and solutions. It also provides a user forum and support network; Sparx Systems are highly responsive to user feedback and requirements, and the web site enables rapid communication concerning problems, solutions and enhancements.

You can access the web page and user forum within Enterprise Architect from the **View | More Windows | Web Browser** menu option, and through the **Tasks Pane Online Resources** topics.

If you do not have Enterprise Architect open, the Sparx Systems website address is [http://www.sparxsystems.com/](http://www.sparxsystems.com/).

The user forum address is [www.sparxsystems.com/cgi-bin/yabb/YaBB.cgi](http://www.sparxsystems.com/cgi-bin/yabb/YaBB.cgi).
1.2.1 Available Helpfile Formats

You can access the latest Enterprise Architect help files from the following locations:

- **.CHM** format: [www.sparxsystmes.com/bin/EA.chm](http://www.sparxsystmes.com/bin/EA.chm)
- **.CHM** format inside a **.ZIP** file: [www.sparxsystmes.com/bin/EAHelp.zip](http://www.sparxsystmes.com/bin/EAHelp.zip)
- **.PDF** format: [www.sparxsystmes.com/bin/EAUserGuide.pdf](http://www.sparxsystmes.com/bin/EAUserGuide.pdf)

Version and release date information for the help files can be found at:

- [www.sparxsystmes.com/ea_downloads.htm#Helpfiles](http://www.sparxsystmes.com/ea_downloads.htm#Helpfiles), or

1.2.2 Support

Technical support for Enterprise Architect is available to registered users. Responses to support queries are sent by email. Sparx Systems endeavors to provide a rapid response to all product-related questions or concerns.

Registered users can lodge a support request, by visiting:

Trial users can contact Sparx Systems with questions regarding their evaluation at:
support@sparxsystems.com.

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

Please take the time to read the following legal statements concerning Sparx Systems Enterprise Architect:

- Software Copyright Notice
- Enterprise Architect End User Licensing Agreement
- Acknowledgement of Trademarks

Spark Systems would also like to gratefully acknowledge contributions to the development of Enterprise Architect.
1.3.1 Copyright Notice

Copyright © 1998 - 2008 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 can 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
Sales Email: sales@sparxsystems.com
Website: www.sparxsystems.com
1.3.2 End User License Agreement

Enterprise Architect - UML CASE Tool - Desktop, Professional and Corporate editions, Version 7.1

Copyright (C) 1998-2008 Sparx Systems Pty Ltd. All Rights Reserved

IMPORTANT- READ CAREFULLY: This End User License 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 License 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 License Agreement.
- "SPARX" means Sparx Systems Pty Ltd A.B.N 38 085 034 546.
- "Licensee" means YOU, or the organization (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/products/ea/purchase.html, following the thirty day free evaluation period.
- "SOFTWARE PRODUCT" or "SOFTWARE" means Enterprise Architect, UML Case Tool, Desktop, 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 LICENSE

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 want 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 LICENSE

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 license must be purchased (as described at http://www.sparxsystems.com/products/ea/purchase.html). Upon payment of the license 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-license, loan or distribute the SOFTWARE PRODUCT other than as expressly authorized by this EULA.

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

If the Software Product purchased is an Academic Edition, YOU ACKNOWLEDGE THAT the license 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 license 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’ 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
  - 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 licenses.


GOVERNING LAW

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

Trademarks of Microsoft
- Microsoft Word
- Microsoft Office
- Windows®
- ActiveX

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

Trademarks of The OMG
- OMG™
- Object Management Group™
- The CORBA logo
- ORB™
- Object Request Broker™
- The CORBA Academy design
- OMG Interface Definition Language™
- IDL™
- CORBA services™
- CORBA facilities™
- CORBA med™
- CORBA net™
- Unified Modeling Language™
- UML™
- The UML Cube logo
- MOF™
- CWM™
- Model Driven Architecture™
- MDA™
- OMG Model Driven Architecture™
- OMG MDA™
1.3.4 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. Your help has been invaluable.
1.4 If You Have the Trial Version

If you are exploring one of the Enterprise Architect trial versions, note that the software operates for a limited period. To continue using Enterprise Architect when the trial period expires, you can purchase and register a full license as explained in the following topics:

- **Order Enterprise Architect**
- **Installation**
- **Register a Full License**

If you already have a full license edition of Enterprise Architect and want to register Add-Ins or upgrade to the Professional or Corporate editions, see the **License Management** topic.
1.4.1 Order Enterprise Architect

Enterprise Architect is designed, built and published by Sparx Systems and available from Sparx Systems. The trial version of Enterprise Architect is identical to the registered edition with the exception that all diagrams are output to files with an embedded watermark. The trial software stops working after the trial period has elapsed. On purchase of a suitable license or licenses, the registered version is 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.
1.4.2 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, 10g or 11g options (see below). Please note that installation and maintenance of these database management systems is not covered under the support agreement.

The latest evaluation and registered versions of Enterprise Architect are always available from the Sparx Systems website. The registered version is available through the registered user area of the website, which requires a username and password to access. These are provided upon purchase of a license.

System Requirements

The system requirements for installing Enterprise Architect are defined on the Enterprise Architect | System Requirements page of the Sparx Systems website.

Windows Vista

Under Windows Vista (with User Account Control turned on) an application starts with only Standard permissions, regardless of what level of authority the current user has. As a result, an installer run normally with an Admin account under Vista only has Standard privileges and either is not able to write to certain critical areas of the registry/file system, or redirects the write requests to a per-user virtualized registry/file system.

Sparx Systems recommend that if you are installing on Windows Vista, always run the Enterprise Architect installer with Administrator privileges (right-click on the downloaded installer icon and select the Run as administrator menu option).

Install Enterprise Architect

Run the Enterprise Architect setup program. Generally you can accept all the default options without change.

To place Enterprise Architect in a directory other than the default, enter the name of the destination when prompted.

You might be prompted to restart your computer when the installation completes. Although this is not always necessary (if you already have the components Enterprise Architect requires installed on your computer), you should restart just to be certain.

If you intend to run Enterprise Architect on Linux, refer to the Installation and Use page on the Sparx Systems website.

Corporate edition users planning to use SQL Server, MySQL, PostgreSQL, Sybase Adaptive Server Anywhere or Oracle 9i, 10g or 11g as their model repository can access scripts that create the required data structures for the choice of DBMS. You can find these at one of the following pages:

- The Corporate edition Resources page
- The Trial Corporate edition Resources page.

Note: Enterprise Architect requires Read/Write access to the program files directory where Enterprise Architect has been installed.
1.4.3 Register a Full License

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

Register Enterprise Architect

To obtain the full version and complete the registration process, follow the steps below:

1. Purchase one or more licenses.
   Once you have paid for a licensed version of Enterprise Architect, you receive (via email or other suitable means):
   - a license key or keys
   - the address of a web site from which to download the full version.
2. Save the license key and download the latest full install package from the address supplied.
3. Run the setup program to install the full version.
4. Open Enterprise Architect from the Start Menu or desktop icon.
5. Select the Help | Register and Manage License Key(s) menu option. The License Management dialog displays.

6. Click on the Add Key button. The Enter Registration dialog displays.
7. In the Copy registration key... field, copy the license key, including the { and } bracket characters (use Copy and Paste from an email to avoid typing mistakes).
8. Click on the OK button. The full version is now activated on your PC, and Enterprise Architect displays...
the message: *Registration succeeded! Thank you for purchasing Enterprise Architect <type> Edition.*

**See Also**

- Add License Key
- Upgrade an Existing License
- Register Add-In
2 Start UML Modeling

This guide provides two options to help you gain an understanding of how to perform UML modeling with Enterprise Architect:

- A brief outline of the types of work and tasks that Enterprise Architect supports, so that you can quickly locate the more detailed explanations of subjects that interest you.
- A Quick Start tutorial to Enterprise Architect that illustrates how to open and create new projects, navigate Enterprise Architect, and use Enterprise Architect to perform various tasks in system and process modeling. It leads on to examining the Enterprise Architect User Interface and work areas applicable to certain Project Roles.

At various points throughout the Enterprise Architect Help, there are further Quick Start topics and sections to help you use the system immediately to experiment with a feature of Enterprise Architect. Use the Help Index tab and search for Quick Start to locate these topics.
2.1 Work with Enterprise Architect

Enterprise Architect is a powerful CASE tool for specifying, documenting and building software projects. Using Enterprise Architect's support for UML and its related standards, you can model new complex software and business systems, or visualize and maintain existing systems.

This topic introduces the fundamental processes that Enterprise Architect supports.

Modeling with UML

Enterprise Architect is a comprehensive UML analysis and design tool. To create models with Enterprise Architect, you therefore should become familiar with:

- how Enterprise Architect implements the UML standards and
- how you apply UML in Enterprise Architect to develop your models.

For more information, see the Modeling With UML topic.

Managing UML Models

To manage the models in your projects, you both protect and manage the model data itself, and communicate information on the data in the form of RTF and HTML documentation and reports.

For more information, see the Manage UML Models topic.

Code Engineering

In Enterprise Architect, UML modeling both depends on and supports code engineering - you generate and update code from a model, and you create and update models from code. In this broad sense, Enterprise Architect enables you to:

- Forward engineer, reverse engineer, round-trip and synchronize code in a range of programming languages
- Debug and profile code
- Model and generate code for XML Technologies
- Perform database modeling and database design for a range of database management systems
- Convert model components from one domain to another using Model Driven Architecture (MDA) Transformations.

For more information, see the Code Engineering topic in this section.

Managing Projects

Enterprise Architect provides strong support for Project Management, particularly in the following areas:

- Project estimation - working out how much time and effort is required to build and deploy a solution, using the Use Case metrics facility and carefully-calibrated metrics
- Defining, assigning and managing resources
- Monitoring and managing problems, changes, issues and tasks that affect both individual elements and the project as a whole
- Managing the development, execution and results of testing, from Integration through to User Acceptance, and
- Maintaining a project glossary of terms, procedures and policies applied to the project.

For more information, see the Project Management topic.

Project management discussions and decisions can be communicated to the project through the Project Discussion Forum.
The scope of your project management might include upgrades to Enterprise Architect and installation of related technologies. In this case, also see License Management.

**Extending Enterprise Architect Facilities**

Experienced Technology Developers can develop customized additions to the functionality already present within Enterprise Architect. These additions include:

- **UML Profiles** and **Stereotypes**
- **UML Patterns**
- **Code Templates**
- **Tagged Value Types**
- **MDG Technologies** and
- **Enterprise Architect Add-Ins**.

By creating these extensions the Technology Developer can customize the Enterprise Architect modeling process to specific tasks and speed up development.

For more information, see SDK For Enterprise Architect.
2.1.1 Modeling With UML

Enterprise Architect is a comprehensive UML analysis and design tool. Enterprise Architect has a library of UML data structures that you can use and extend to develop your models.

UML Structures

To explain how Enterprise Architect interprets the UML standards and specifications, Sparx Systems provides a UML Dictionary of diagrams, elements and connectors.

- You create your projects and models using the Start Page or File Menu, which provide templates on which to base your models.
- You initially create your packages and diagrams using the Toolbars and Menus, and the elements and connectors using the Enterprise Architect UML Toolbox.
- You can also create new structures through the Project Browser, re-use existing structures using the Project Browser, Model Views, Element List and Model Search.

UML Modeling With Enterprise Architect

Modeling in Enterprise Architect is the process of graphically representing a business process or software system. The resulting model can be used to emphasize a certain aspect of the system being represented and to record and communicate its detail.

Building models requires the use of various UML data structures and Enterprise Architect tools, as above. A further extremely useful tool is the:

- Relationship Matrix, which enables you to visualize and amend the relationships and hence organization of structures within the model.

Enterprise Architect also provides particular support for:

- Requirements Management and
- Modeling the business process, an essential part of any software development process.

You can extend the scope of your models by using:

- UML Stereotypes, Profiles and Patterns, and
- MDG Technologies.

For more information, see the Modeling With Enterprise Architect topic.
2.1.2 Manage UML Models

To manage the UML models in your projects, you both protect and manage the model data itself, and communicate information on the data in the form of documentation and reports.

UML Model Management

In managing models, you control:

- The model files in a Microsoft JET database or (Corporate edition) in one of a range of DBMS repositories
- Model data integrity
- Development of the models in a shared, team environment
- Versions of the model, ensuring that work on different areas of the model is coordinated and synchronous rather than conflicting
- User security
- Transfer of value data and reference data between projects and models
- Changes to model data, using model auditing, Baselines and a differencing utility that enables you to roll back changes to a previous state
- Model upgrades
- Replication of models for parallel development (EAP repositories only)
- Extensions of development with Add-Ins and the Enterprise Architect Automation Interface

You can also have recorded discussion and communication of decisions using the Project Discussion Forum.

For further information, see the Model Management topic.

Generating Model Documentation

You can generate documentation from the components of your model, in RTF or HTML format. You can also generate a range of RTF reports on your model.

For more information, see the Enterprise Architect Reports topic.
2.1.3 Code Engineering

Code Engineering with Enterprise Architect broadly encompasses various processes for generating or transforming code from your UML model and importing code into the model, to support model development in several coding languages, database development and SOA development.

Code Engineering

Enterprise Architect supports:

- **Source code generation** and reverse engineering for many popular languages, including C++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript, Python and PHP.

Enterprise Architect also provides:

- A built in ‘syntax highlighting’ source code editor
- **Code generation templates**, which enable you to customize the generated source code to your company specifications.

For more information, see the Code Engineering topic.

MDA Transformations

Enterprise Architect provides:

- Advanced Model Driven Architecture (MDA) transformations using transformation templates
- Built-in transformations for DDL, C#, Java, EJB and XSD.

One PIM can be used to generate and synchronize multiple PSMs, providing a significant productivity boost.

For more information, see the MDA Transformations topic.

Debug And Profile

Enterprise Architect enables you to:

- Build, test, debug, run and execute deployment scripts
- Integrate UML development and modeling with source development and compilation
- **Generate NUnit and JUnit** test Classes from source Classes using MDA Transformations
- Integrate the test process directly into the Enterprise Architect IDE
- Debug .NET, Java and Microsoft Native (C, C++ and Visual Basic) applications.

For more information, see the Debug and Profile topic.

Database Modeling

Enterprise Architect enables you to:

- Reverse engineer from many popular DBMSs, including SQL Server, My SQL, Access, PostgreSQL and Oracle 9i, 10g or 11g
- Model database tables, columns, keys, foreign keys and complex relationships using UML and an inbuilt data modeling profile
- **Forward generate DDL scripts** to create target database structures.

For more information, see the Data Modeling topic.

XML Technology Engineering

Enterprise Architect enables you to rapidly model, forward engineer and reverse engineer two key W3C XML technologies:

- **XML Schema** (XSD)
- **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 2.1 and XML provides the natural mechanism for implementing XML-based SOA artifacts within an organization.

For more information, see the XML Technologies topic.
2.2 Quick Start - Create a Project

Tutorial
Welcome to Enterprise Architect! This quick-start tutorial helps you start UML modeling with Enterprise Architect.

As you read through the Quick Start sections, have Enterprise Architect open so that you can explore and try out the functions described. By the end of the Quick Start tutorial you should be able to begin modeling your own software projects with Enterprise Architect and UML.

The tutorial guides you through creating a simple project. Throughout the descriptions there are hyperlinks to more detailed information on a range of topics. Follow these links if you would like more information, or ignore them if you want to just follow the steps.

Your task is to create a new *project* and then add a View, package, diagram, elements and connectors.

Create a Project
When you start Enterprise Architect it opens at the **Start Page**:

1. Click on the **Create a New Project...** option. The **New Project** dialog displays.

2. In the **File name** field, type a meaningful name for the project and click on the **Save** button to create the project file. The **Model Wizard** displays.

3. You now select one or more **model templates** (these provide you with the basic structures - packages and diagrams - for your project, as well as references to useful help files to get you started). Select the checkbox of each model that interests you.

4. Click on the **OK** button. Enterprise Architect creates your project and displays it in the **Project Browser** on the right-hand side of the screen.

**Note:**
You could also quickly create a project by copying an existing base project provided with Enterprise Architect; see the **Copy a Base Project** topic.

Expand The Project
To navigate through your project, in the **Project Browser** click on the ‘plus’ icon against each folder or **package** to expand it.
Double-click on the diagram icon displayed underneath a package name. Enterprise Architect displays the sample diagram for that model in the Diagram View, which is in the middle of the screen.

**Add a View To Your Model**

Now that you have created a project containing at least one model, you can add another View to a model, and then add a package with diagram, elements and connectors (relationships).
### 2.2.1 Add a View to a Model

A **View** is the highest-level container, or package, within a model. There are six types of View, five of which represent conventional ways of categorizing the structures or purposes of a model, and one (**Simple View**) for developing your own categorization.

To create a View, follow the steps below:

1. Right-click on your model name in the Project Browser. The context menu displays.
2. Select the **New View** menu option. The Create New View dialog displays.

   ![Create New View Dialog](image)

3. In the **Name** field, type the name of the View.
4. In the **Set View Icon Style** panel, click on the radio button for the type of View to create.
5. If the model root node had been under version control, the **Add to Version Control** checkbox would display, defaulted to selected. Ignore this for now.
6. Click on the **OK** button.

#### Add a Package To Your Model

Now that you have created a View in the model, you can add a **package** and diagram to that View or any other in the model, and then add elements and connectors (relationships).
2.2.2 Add a Package To a Model

A **Package** is a container of model elements, and is displayed in the **Project Browser** as the 'folder' icon familiar to Windows users. Package contents are arranged alphabetically.

In the **Project Browser** click on a package and, in the **Project Browser** toolbar, click on the **New Package** icon.

Enterprise Architect displays a prompt for the package name.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>This prompt also contains the <strong>Automatically add new diagram</strong> option for automatically creating a diagram for the package, which defaults to selected. This is very a useful feature, but for the purposes of this introduction deselect the checkbox against the option.</td>
</tr>
</tbody>
</table>

Type in a name and click on the **OK** button. Enterprise Architect adds the new package subordinate to the package you selected.

Add a Diagram To a Package

Now add a diagram.

Additional Information

For additional information on adding packages and Views (top-level packages), see the **Add a Package** topic and **Add Additional Views** topic.
2.2.3 Add a Diagram to a Package

A diagram is a representation of the components or elements of your model and, depending on the type of diagram, how those elements are connected or how they interact.

When you first create a project, Enterprise Architect provides simple examples of diagrams appropriate to your selected model patterns, with annotations. You can edit these diagrams, but here we create an additional one.

Click on your new package and, in the Project Browser toolbar, click on the New Diagram icon. The New Diagram dialog displays.

**Note:**

When you create a package, if you leave the Automatically add new diagram option selected, the New Diagram dialog displays automatically.

Click on a diagram category in the Select From panel, and a diagram type in the Diagram Types panel, then click on the OK button. Enterprise Architect adds a diagram object to the package, with the same name as the package. It also opens the Diagram View for your diagram, in the center of the screen.

Add Elements to a Diagram and a Package

Now add some elements.

Additional Information

For additional information on adding diagrams to a project, see the Add New Diagrams topic.
2.2.4 Add Elements

You have several options for adding elements (the UML model building units) to a package and/or diagram. The simplest method is to use the Enterprise Architect UML Toolbox to the left of the diagram, which automatically lists the elements applicable to the type of diagram you have created. Just click on the required element and drag it onto your diagram.

Two things might occur before the element displays on the diagram:

- If you have selected an Object element, Enterprise Architect prompts you to define what stereotype the object is based on (an object can represent a wide range of things, and a stereotype helps you define what the object or element is); for now, select any value.
- The element Properties dialog displays. If it does not display, double-click on the element on the diagram.

You can use the Properties dialog to define the characteristics of the element, such as its name. Type a name in the Name field, and click on the OK button. Look at the Project Browser, underneath the package in which you created the diagram. The element is listed.

**Tip:**

Enterprise Architect has two very useful features:

- To find out more about the type of element you have dragged on to a diagram, right-click on the element and select the UML Help menu option. This displays a Help page on the element type.
- If you are creating several elements of one type, after creating the first just press [Shift]+[F3] or [Ctrl]+click to create the next element of that type.

You can also **drag or paste existing elements** onto a diagram from the Project Browser. This enables you to make use of previous work in defining elements.

Add Connectors Between Elements

Now **connect the elements** with relationships.

Additional Information

For additional information on adding elements to a project, including via the Quick Linker, see the Create Elements topic and the Create New Elements topic.
2.2.5 Add Connectors

Connectors define specific relationships between specific elements, so you usually create them directly on the diagram by dragging the required relationship type from the Enterprise Architect UML Toolbox. As for elements, the Toolbox automatically presents the connector or relationship types appropriate to the type of diagram.

Create two elements on the diagram. Click on a connector in the Toolbox, click on the source element in the relationship, then drag across to the target element. This creates the selected connection between the two elements. If you double-click on the connector, the connector Properties dialog displays, and you can define the characteristics of the relationship.

Tip:

Enterprise Architect has three very useful features:

- To find out more about the type of connector you have dragged on to a diagram, right-click on the connector and select the UML Help menu option. This displays a Help page on the connector type.

- If you are creating several connectors of one type, after creating the first just click on the appropriate source element and press [F3] to create the next connector of that type.

- As you drag a connector, you can press [Shift] to create a bend in the connector. If necessary, you can put several bends in the connector line, pressing [Shift] every time you want to change direction. To roll back the bends, keep holding the left mouse button down and press [Backspace] as many times as is necessary.

Moving and Deleting Elements and Connectors

Having created a model with some components, you can move those components around and delete them. You should also know how to save your work.

Additional Information

For additional information on creating a connector through the Project Browser, or with the Quick Linker, see the Create Connector topic and the Create Connections Between Elements topic.
2.2.6 Define Properties

When you create an element and connect it to another element, you usually have to define various characteristics of both the element and the connector to identify the purpose and function they represent. You do this using a Properties dialog.

Enterprise Architect is initially configured to display the Properties dialog automatically when you create an element or connector, but it is easy (and often convenient) to turn the dialog display off. If the default display has been turned off, you can display the dialog by:

- double-clicking on the element or connector in the diagram or
- right-clicking on it in the Project Browser and selecting the Properties menu option.

Properties dialogs vary between element types and between elements and connectors but, as you saw when you created your first element, they look something like this:

![Properties Dialog Example](image)

When you create elements, Enterprise Architect automatically names and numbers them by type - for example, Class1, Class2 - so you should at least change the Name field to more easily identify each element. See the Element Properties topic for a full description of the element Properties dialog.

Enterprise Architect does not automatically name connectors, but for many connector types you should provide a name that describes the purpose of the connection. See the Connector Properties topic for a full description of the connector Properties dialog.

Explore User Interface

So far you have been using the Project Browser and Diagram View to develop your project. At this point you should find out a bit more about the other facilities of the Enterprise Architect User Interface.

When you have finished exploring the User Interface topics, go to Quick Start - Project Tasks to identify...
areas of Enterprise Architect that provide particular support for your job role.
2.2.7 Move Components

You have created a project containing packages, diagrams and elements, and you have connected the elements. You might have arranged your components in the wrong project structure. How do you change where things are?

Note:
Changing names and properties is discussed a little later.

In this topic, the explanations refer to the following example:

![Project Browser diagram](image)

Notes:
- You display and work on your model in the Project Browser, and display and work on a diagram in the Diagram View.
- In the Project Browser, the contents of a package are listed in the order diagrams / child packages / elements. Elements are further arranged in type order. Within their types, components are initially listed in alphabetical or numerical order.
- Moving an element or package has no effect on any relationships that the element, package, or elements within the package have. You have to specifically create, delete or move the relationships themselves.

Move Components Within a Package in the Project Browser
To move a diagram, child package or element within its parent package, click on it in the Project Browser and click on or in the toolbar at the top of the window.

You could move Class3 in the Project Browser above Class1, or move the Actors package underneath Classes A.

To revert to listing components in alphabetical order, right-click on the package and select the Contents | Reset Sort Order menu option.

Move Components Between Packages in the Project Browser
You might have created a diagram, child package or element in the wrong place in the Project Browser. To move a model component to another package, click on the component and drag it to the new package. This can be at either a higher level or a lower level.

You might, for example, drag Class1 from the Use Case Model package into the Business Process Model package. Class1 then is listed in the Business Process Model package in the Project Browser. As a similar example, you could drag Class Diagram into the Business Process Model package.

Moving elements in the Project Browser does not affect the use of elements in diagrams. In our example, Class1 is initially in a diagram in the Use Case Model package. When you move Class1 in the Project Browser
from Use Case Model to Business Process Model, it still shows in the diagram in Use Case Model, and does not display in any diagram in Business Process Model.

**Note:**
Moving a diagram generally does not affect the location of elements in packages. If you move the Class Diagram out of Use Case Model into Business Process Model, all the elements in the diagram remain in the Use Case Model package.

However, elements of certain types might be used only within one diagram, have no meaning outside that diagram, and never be re-used in any other diagram. Such elements include Decision, Initial and Final Node elements. Therefore, if you move a diagram containing these elements, they are moved to the new parent package with the diagram.

To remove Class1 from the Use Case Model diagram, click on it on the diagram and delete it. Nothing happens to the element in the Project Browser. To put Class1 into a diagram in the Business Process Model package, open the diagram in that package and drag the element from the Business Process Model package in the Project Browser onto the diagram.

**Move Elements in a Diagram**
If an element is not in the right position in the diagram, just click on the middle of it and drag it to the correct place. In the diagram above, you might move Class2 below Class 3, and move Class3 to the left. The element brings its connectors with it.

**Move Connectors in a Diagram**
You might have connected the wrong pair of elements. To move the end of a connector to a different element (for example, Class2 instead of Class3), click on the end to display a black 'handle' box and drag the end to its new position. Be aware that the connector does not break from the original target element until the cursor is on the new target.

You can also tidy up a connection by dragging the end of the connector to a better position on the edge of the element, or move both ends at once by dragging the middle of the connector.

**Additional Information**
See the topics on Deleting Components and Saving Changes.

For additional information on moving connectors and elements, see the Arrange Connectors topic and the Order Package Contents topic.
2.2.8 Delete Components

You can delete the components of a model from a diagram or from the Project Browser.

Delete From a Diagram

A diagram can contain elements, connectors, packages and other diagrams. To delete any of these from the diagram, click on it and press [Delete] on the keyboard.

Notes:

- Remember that the contents of the model are listed in the Project Browser. If you delete something from a diagram, it is not deleted from the Project Browser. This is because you can use the same component in several diagrams at once, so you only remove the representation of the component from a diagram.
- To delete a connector from a diagram, click on it and press [Delete]. This time, Enterprise Architect prompts you to select whether to delete the connector or just hide it. Unlike elements, the same connector is not reapplied in several places, so if you delete one it is removed completely from the model.
- Connectors can get confusing on a complex diagram, so it is useful to hide some of them to clarify a specific aspect of a more complex picture. To identify and reveal hidden connectors, see the Connectors topic. However, first you should become more familiar with element and connector properties, through the Quick Start - Define Elements and Relationship Properties topic.

Delete From Project Browser

To delete a package, diagram or element from the Project Browser, right-click on the component and select the Delete <name> menu option.

For a package, this completely removes the package and all its contents - diagrams, child packages and elements - from the model.

For an element, this completely removes the element and its properties, connectors, child elements and child diagrams from the model, and from every diagram that contains it.

For a diagram, this completely removes the diagram and connectors from the model, but not the diagram's component elements. They remain in the parent package.

Additional Information

See the Save Changes topic.

For additional information on deleting elements, connectors and model views in Enterprise Architect, see the Delete Elements and Delete Connectors topics, and the Delete Views topic.
2.2.9 Save Changes

Throughout much of your work in Enterprise Architect, any changes you make are automatically saved when you close the dialog (data entry window) on which you made the changes. In some cases the dialog contains a Save or Apply button, which enables you to save your changes and then keep working on the dialog.

If there is no specific dialog, such as when you create a diagram, you can save your work by:

- Pressing the [Ctrl]+[S] keyboard keys
- Clicking on the Save icon in the Diagram toolbar or
- Selecting the Diagram | Save menu option.

Often, Enterprise Architect does not let you close a screen without confirming that you want to save or discard your changes.
2.3 Quick Start - Project Tasks

Throughout a design and development project there are many different tasks to be performed, which could be carried out either by one person or - more probably - by members of a team with different responsibilities. In either case, Enterprise Architect supports most - if not all - of the responsibilities you might have on your project.

Therefore, the next topics to explore depend on the work you normally do on a project.

The descriptions below cover a number of job roles that Enterprise Architect supports. For those that most resemble your role on a project, follow the job title hyperlink to display a description of how that role might use Enterprise Architect, then follow links within those topics to explore some of the Enterprise Architect features of importance to the role.

Another area of responsibility that Enterprise Architect supports is System Administration - see the Model Management topic.

Most of these roles work with specific types of diagram, so you might want to learn more about diagram types in general and specific types of diagram in particular. See the UML Diagrams topic in the UML Dictionary.

Several types of project team member might want to generate documentation on their work and reports on how the project is developing and changing. Enterprise Architect enables you to generate project documentation in either RTF or HTML format - see the Enterprise Architect Reports topic.

Note:

The Corporate edition of Enterprise Architect has a user security feature that can be applied or turned off. If security is turned on, you require the appropriate access permissions to use many of the Enterprise Architect facilities listed above. For further information, see the List of Available Permissions topic in the User Security topic.
Business Analyst
A Business Analyst might be responsible for modeling:
- Requirements
- High-level business processes
- Business activities
- Work flows
- System behavior.

Software Architect
A Software Architect might be responsible for:
- Mapping functional requirements of the system
- Mapping objects in real time
- Mapping the deployment of objects
- Defining deliverable components.

Software Engineer
A Software Engineer might be responsible for:
- Mapping Use Cases into detailed Classes
- Defining the interaction between Classes
- Defining system deployment
- Defining software packages and the software architecture.

Developer
A Developer might be responsible for:
- Forward, reverse and round-trip engineering
- Visualizing the system states
- Visualizing package arrangements
- Mapping the flow of code.

Technology Developer
A Technology Developer might be responsible for creating or customizing:
- UML Profiles
- UML Patterns
- Code Templates
- Tagged Value types
- MDG Technologies
- Add-Ins.

Database Administrator
A Database Administrator might be responsible for:
- Developing databases
- Modeling database structures
- Creating logical data models
- Generating schema
- Reverse engineering databases.

Tester
A Tester might be responsible for:
- Developing test cases
- Importing requirements, constraints and scenarios
- Creating Quality Test documentation
- Tracking element defects and changes.

Project Manager
A Project Manager might be responsible for:
- Providing project estimates
- Resource Management
- Risk Management
- Maintenance Management.

Implementation Manager
An Implementation Manager might be responsible for:
- Modeling the tasks in rolling-out a project, including network and hardware deployment
- Assigning and tracking maintenance items on elements (issues, changes, defects and tasks).
Part III
This section provides a detailed exploration of the Enterprise Architect UML Modeling tools and features for modeling software systems and business processes, including:

- Starting Enterprise Architect
- The User Interface
- The Start Page
- Model Templates
- The Project Browser
- The Main Menu
- The Enterprise Architect UML Toolbox
- Workspace Toolbars
- Diagram Tabs
- View Options
- Search a Project
- The Web Browser
- Arrange Windows and Menus
- Dockable Windows
- The Quick Linker
- Defaults and User Settings
- Keyboard Shortcuts
- The Project Discussion Forum
- Register Add-In
- The Spell Checker
3.1 Start Enterprise Architect

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

Start Enterprise Architect

You can start Enterprise Architect 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.

After a short pause, the Start Page displays. From this dialog you can:

- 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 Enterprise Architect, 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 you become familiar with UML and software engineering using Enterprise Architect.

To begin a guided exploration of Enterprise Architect immediately, go to the Quick Start - Create a Project topic.

See Also

(For users of the Corporate edition)

- Connect to a MySQL Repository
- Connect to an SQL Server Repository
- Connect to an Oracle Repository
- Connect to a PostgreSQL Repository
- Connect to an Adaptive Server Anywhere Repository
- Connect to an MSDE Server Repository
3.2 The User Interface

The Enterprise Architect Application Workspace consists of a number of windows, menus and toolbars as described below. Together these elements provide a simple and flexible software engineering environment. In concept the Application Workspace is similar to programs such as Microsoft Outlook and the Microsoft Visual Studio application suite; if you have used these applications you should find the Enterprise Architect interface quite familiar.

Enterprise Architect in Action

A demonstration of Enterprise Architect in use is provided on the Sparx Systems website.

Workspace Components

This section outlines the components of the Enterprise Architect Application Workspace. To obtain further information on specific features, follow the hyperlinks in each description.

Main Menu and Toolbars

At the top of the workspace are the Main Menu and Toolbars. The Main Menu provides access to further submenus. There are several toolbars, which you can hide or display as necessary.

Context Menus

Throughout Enterprise Architect, if you right-click on work areas, lists and objects, Enterprise Architect displays a menu of options specific to the work context. For examples, see:

- Package Context Menu (Project Browser)
- Diagram Context Menu (Project Browser)
- Element Context Menu (Project Browser)
- Diagram Context Menu (Diagram)
- Element Context Menu (Diagram).

Key Combinations

Many main menu and context menu options have alternative key combinations to perform the same operation. Instead of displaying a menu and selecting the required option, you can press the key combination. See Keyboard Shortcuts for a full list of key combinations and their functions, or display the Help Keyboard dialog (select the Help | Keyboard Accelerator Map menu option). You can also customize these function keys.

Enterprise Architect UML Toolbox

The Enterprise Architect UML Toolbox is an Outlook-style toolbar from which you can select model elements and relationships to add to your modeling diagrams. This is an important feature of Enterprise Architect, as it provides all the components and connectors that you use to create your models in whatever diagrams are appropriate.

Diagram View

The large central area of the Enterprise Architect display is the Diagram View. This is where you can arrange new model elements and set their characteristics in a model diagram. Note that when you first open Enterprise Architect there is no active diagram; you must create and/or open the required diagram.

Project Browser

The Project Browser is used to navigate your project. Double-click on package icons to open them and display the diagrams and elements they contain. Similarly, double-click on elements to open them, and on diagrams to display them in the Diagram View. You can drag elements from the Project Browser to add them to diagrams.

Model Views

You can set up tailored views of your model, containing sections or organizations of your model that are of particular relevance to you or your team. Model Views are stored in the model and are visible to all users. You can set up Favorites folders to give you easy access (hyperlinks) to commonly-used packages and elements. You also have a My Views model stored locally on your machine and only visible to you, and Technology-defined views that are read only and stored with MDG technologies. You can associate each view with a query-built search that you can run by either double-clicking or expanding it.
Visual Style
You can configure the look and feel of Enterprise Architect to suit your working environment. Options range from a classic windows application to an enhanced XP appearance.

Arranging Windows
You can rearrange windows and some menus to adapt the screen space to your work habits. You can:

- Dock windows against any edge of the workspace, or move them freely (float them) as you work; for a list of dockable windows, see Dockable Windows.
- Autohide windows so that they display only when you are actually using them.
- Tear off submenus so that they stay open in a convenient (docked) location.
3.3 The Start Page

When you start Enterprise Architect, the first page displayed is the Start Page.

This page offers the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>Locate an object in Enterprise Architect. Type the name of the object in this field and click on the [...] button. Enterprise Architect displays the results of the search on the Model Search screen. Click on an item in the search results to highlight it in the Project Browser.</td>
</tr>
<tr>
<td>Getting Started</td>
<td>Opens the Tasks Pane to display useful topics and guides for various areas of work in Enterprise Architect.</td>
</tr>
<tr>
<td>Online Resources &amp; Tutorials</td>
<td>Open the Resources page of the Sparx Systems website, which provides access to a wide range of Enterprise Architect and UML tutorials, demonstrations, examples, Add-Ins and discussions.</td>
</tr>
<tr>
<td>Configure Options</td>
<td>Display the Options dialog which enables you to define how Enterprise Architect displays and processes information.</td>
</tr>
<tr>
<td>Open a Project File</td>
<td>Display the Open Project dialog, which you use to open an existing project (where you have more project files than can be listed in the Recent panel).</td>
</tr>
<tr>
<td>Create a New Project</td>
<td>Save a new project and open the Model Wizard dialog.</td>
</tr>
<tr>
<td>Copy a Base Project</td>
<td>Select a different Base Project to generate a new project from.</td>
</tr>
<tr>
<td>Connect to Server</td>
<td>Specify a Data Source name to connect to. MySQL, SQL Server, Oracle 9i, 10g and 11g, PostgreSQL, MSDE, Adaptive Server Anywhere, and Progress OpenEdge repositories are supported.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Note:</td>
<td>This feature is available in the Corporate edition only.</td>
</tr>
<tr>
<td>Recent</td>
<td>Select from a list of the most recently used Enterprise Architect projects (both .EAP files and DBMS connections). Click on the required project to open it.</td>
</tr>
<tr>
<td></td>
<td>If you have created and used shortcuts to your models, a model might have two entries - one for the model accessed through Enterprise Architect and one for the model accessed through the desktop shortcut. These open the same model, although the shortcut entry also enacts any view profile you have defined.</td>
</tr>
<tr>
<td></td>
<td>To remove a hyperlink to a project from this list, see Remove Recent Projects.</td>
</tr>
</tbody>
</table>

If your model has a default diagram set, the default diagram opens immediately over the top of the Start Page. You can still access the Start Page from the diagram tabs below the diagram. However, if you have set a shortcut view profile, that overrides the default diagram setting.
3.3.1 Remove Recent Projects

To remove a project hyperlink from the Recent list on the Start Page, follow the steps below:

1. Select File | Open Project from the menu bar or press [Ctrl]+[O]. The Open Project dialog displays.

2. In the Recent Projects panel click on the project to remove.

3. Click on the Remove Selection from List button.

**Note:**

Removing the hyperlink to a project from the Start Page only removes the link to the project and does not remove the .EAP file from the file system.
3.4 Model Templates

The model templates contained in Enterprise Architect are designed to assist in the creation of projects and models for both new and experienced users. Each template provides a framework on which you can create your model. You create models based on the selected templates using the Model Wizard.

Template Format
All the model templates provided with Enterprise Architect follow the format described below.

Note
The note introduces you to the model template and outlines its purpose.

Help Links
Help hyperlinks provide further information on how to use the model. Depending on the model template, links to examples and other useful information are also provided.

Template
The Template section in the model template provides a framework for creating your own model.

The topics listed below provide an introduction to the terminology and icons used in the model templates, and give you a quick guide to the UML concepts important to the templates and how they are applied in Enterprise Architect.

- Business Process Model Template
- Requirements Model Template
- Use Case Model Template
- Domain Model Template
- Class Model Template
- Database Model Template
- Component Model Template
- Deployment Model Template
- Testing Model Template

If you are a Technology Developer, you can also create and distribute custom templates as part of your own MDG Technology.
3.4.1 Business Process Model Template

The Business Process Model describes 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
- Business Modeling and Business Interaction Diagrams
### 3.4.2 Requirements Model Template

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, connect Requirements to other model elements, connect Requirements into a hierarchy and report on Requirements.

**Online Resources**
- [Requirements Management in Enterprise Architect](#)

**See Also**
- [Packages](#)
3.4.3 Use Case Model Template

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 user’s perspective of the system and interactions. See Use Case and Use Case Diagram for more information.

Online Resources
- The Use Case Model
### 3.4.4 Domain Model Template

A Domain Model is a high-level conceptual model, defining 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 (that is, classes that represent the concept of a group of things rather than Classes that define a programming object). It is also useful for defining the terms of a domain.

A Domain Model shows:

- The physical and organizational units of the domain; for example, Employee and Flight
- The relationships between these units; for example, Employee is assigned to Flight
- The multiplicity of those relationships; for example, one employee can be assigned to no flights, one flight or many flights (represented by the 1 and the * at the ends of that relationship).
3.4.5 Class Model Template

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 Diagram
3.4.6 Database Model Template

The Database Model describes the data that must be stored and retrieved as part of the overall system design. Typically this means relational database models that describe the tables and data in detail and enable generation of DDL scripts to create and set up databases.

Online Resources
- UML Database Modeling

See Also
- Database Modeling
3.4.7 Component Model Template

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

- Online Resources
  - The Component Model

- See Also
  - Component Diagram
3.4.8 Deployment Model Template

The Deployment Model describes how and where a system is to 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 and rollout, the allocation is guided by the use of deployment specifications.

Online Resources
- The Physical Model

See Also
- Deployment Diagram
- Display Tagged Values
3.4.9 Testing Model Template

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
- Test Case
- Show Test Scripts
3.4.10 Maintenance Model Template

The Maintenance Model enables 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.
3.4.11 Project Model Template

The Project Model details the overall project plan, phases, milestones and resourcing requirements for the current project. Project Managers can use Enterprise Architect to assign resources to elements, measure risk and effort and to estimate project size. Change control and maintenance are also supported.

Online Resources
- Project Manager

See Also
- Project Management
### 3.5 The Project Browser

The **Project Browser** enables 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 drag or paste elements from the Project Browser directly into the current diagram.

If you right-click on an item in the **Project Browser**, you can perform additional actions such as adding new packages, creating diagrams, renaming items, creating documentation and other reports, and deleting model elements. You can also edit the name of any item in the **Project Browser** by selecting the item and pressing `[F2]`. 
Tip:
The Project Browser is the main view of all model elements in your model; use the mouse to navigate the project.

Note:
You can hide and show the Project Browser by pressing [Alt]+[0].

Views
The Project Browser can be divided into Views each of which contains diagrams, packages and other elements. A default View hierarchy is described below, but you can create different Views to suit your requirements:

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case View</td>
<td>The functional and early analysis View. Contains Business Process and Use Case models.</td>
</tr>
<tr>
<td>Dynamic View</td>
<td>Contains State Charts, Activity and Interaction diagrams. The dynamics of your system.</td>
</tr>
<tr>
<td>Logical View</td>
<td>The Class Model and Domain Model View.</td>
</tr>
<tr>
<td>Component View</td>
<td>A View for your system components. The high level view of what software is to be built (such as executables, DLLs and components).</td>
</tr>
<tr>
<td>Deployment View</td>
<td>The physical model; what hardware is to be deployed and what software is to run on it.</td>
</tr>
<tr>
<td>Custom View</td>
<td>A work area for other Views, such as formal requirements, recycle bin, interview notes and non-functional requirements.</td>
</tr>
</tbody>
</table>

Selective Collapse
When you are working on an expanded project in the Project Browser, you might want to locate the parent element or package of an item, and/or collapse the structure under that parent element or package. To do this, follow the steps below:

1. Position the cursor on an item within the element or package.
2. Press [→] on the keyboard to highlight the parent.
3. Press the key again to collapse the structure under that parent element or package.

See Also
- Project Browser Icon Overlays
3.5.1 Order Package Contents

Enterprise Architect enables you to change the order of elements listed in the Project Browser.

Elements by default are first listed in order of type, then of set position, then alphabetically. You can use the context menu options to move an element up or down within its type, but not outside its type. This means you can resequence Packages or Diagrams or Use Cases, but you cannot mix elements up. However, you can change this default behavior to allow elements to be re-ordered regardless of type.

Ordering elements is very important when it comes to structuring your model, especially packages. RTF documents honor any custom ordering when printing documentation.
3.5.2 Set Default Behavior

The General page of the Options dialog provides several options for altering the look and behavior of the Project Browser.

To access the General page, select the Tools | Options | General menu option.

Double-click Behavior

In the Double click on browser panel, select the appropriate radio button.

- **Shows Properties** - double-clicking an item in the Project Browser opens a Property dialog (if available) for that element
- **Opens Branch** - double-clicking an item in the Project Browser expands the tree to show the item's children; if there are no children, nothing happens
- **Opens Branch & Diagram** - as above, but also opens the first diagram beneath the item, if applicable.

Enable Free Sorting

In the Project Browser panel, select the Allow Free Sorting checkbox. This enables free sorting of elements in the Project Browser, regardless of element type.

For example, below, the element Object1 has been moved from its original position with the other Object elements, to a point amongst the Class elements.

You move elements using the **Up-arrow** (moves the element further up the tree) and **Down-arrow** (moves the element further down the tree) icons at the end of the Project Browser toolbar.

Show Stereotypes

1. In the Project Browser panel, select the Show Stereotypes checkbox.
2. When prompted, shut down and restart Enterprise Architect to enable this change to take effect.
As shown in the above screen, when a stereotype is defined for an element, the stereotype name then displays in front of the element name (see Artifact1, Class R and Object2).

You set the stereotype of an element such as Object1 in its Properties dialog.
3.5.3 Project Browser Toolbar

At the top of the Project Browser is a toolbar that enables you to perform a range of operations on your project structures.

The functions of each icon in the toolbar are, from left to right:

- Provide options to generate an **RTF report**, **HTML report**, or **Diagram Only** report on the selected package in the Project Browser.
- Provide options to **generate source code**, **DDL**, **import a source directory**, **binary module**, **database schema**, **generate package contents**, to synchronize with package code, or **reset the source code language** all for the selected package.
- **Create a new child package** under the selected package.
- **Create a new child diagram** under the selected package or element.
- **Create a new child element** under the selected package or element.
- Perform a simple search for a text string in the Project Browser.
- Move the selected package or element further up the Project Browser, within its parent package.
- Move the selected package or element further down the Project Browser, within its parent package.
3.5.4 Project Browser Icon Overlays

The Project Browser displays the status of each package in the model by overlaying status icons on the package icon. The following table describes what each overlaid icon means.

<table>
<thead>
<tr>
<th>Icon Overlay</th>
<th>Indicates that...</th>
</tr>
</thead>
<tbody>
<tr>
<td>![file]</td>
<td>This package is controlled and is represented by an XMI file on disk. Version control either is not being used or is not available. You can edit the package.</td>
</tr>
<tr>
<td>![file]</td>
<td>This package is version controlled and checked out to you, therefore you can edit the package.</td>
</tr>
<tr>
<td>![file]</td>
<td>This package is version controlled and not checked out to you, therefore you cannot edit the package (unless you check the package out).</td>
</tr>
<tr>
<td>![file]</td>
<td>This package is version controlled, but you checked it out whilst not connected to the version control server. You can edit the package but there could be version conflicts when you check the package in again.</td>
</tr>
<tr>
<td>![MDG Add-In]</td>
<td>MDG Add-Ins specify their own icon to denote that this branch of the model belongs to that Add-In. All packages connected to an MDG Add-In correspond to a namespace root, so the namespace root icon is not displayed.</td>
</tr>
</tbody>
</table>

Similarly, the Project Browser indicates attribute and operation scope status with icons. The following table describes what each indicator icon means.

<table>
<thead>
<tr>
<th>Icon Overlay</th>
<th>Indicates that...</th>
</tr>
</thead>
<tbody>
<tr>
<td>![protected]</td>
<td>The attribute or operation is scoped as protected.</td>
</tr>
<tr>
<td>![private]</td>
<td>The attribute or operation is scoped as private.</td>
</tr>
</tbody>
</table>

In the Corporate edition, if Project User Security is on, the Project Browser also has element locking indicators (red and blue exclamation marks) that indicate the lock status of individual elements and packages. The availability of these elements for editing depends on whether user locks are required or not. For further information, see the Locked Element Indicators topic.
3.5.5 Model (Root Node) Context Menu

The Root Node in the Project Browser is the Model element. You can 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 such as Use Case Model and Logical Model.

Right-click on the Root Node to display the Model context menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Control</td>
<td>Display the Package Control submenu.</td>
</tr>
<tr>
<td>Rename Model</td>
<td>Rename the current model.</td>
</tr>
<tr>
<td>New Model (root node)</td>
<td>Create a new model root.</td>
</tr>
<tr>
<td>New View</td>
<td>Create a new View (package).</td>
</tr>
<tr>
<td>Add Model using Wizard</td>
<td>Add further models using the Model Wizard.</td>
</tr>
<tr>
<td>Search in Project Browser [Ctrl]+[Shift]+[F]</td>
<td>Search the Project Browser.</td>
</tr>
<tr>
<td>Expand Branch</td>
<td>Expand all items.</td>
</tr>
<tr>
<td>Collapse Branch</td>
<td>Collapse all items.</td>
</tr>
<tr>
<td>Import Model from XMI [Ctrl]+[Alt]+[I]</td>
<td>Import a model from an XMI file.</td>
</tr>
<tr>
<td>Export Model to XMI [Ctrl]+[Alt]+[E]</td>
<td>Export a model to XMI.</td>
</tr>
<tr>
<td>HTML Report [Shift]+[F8]</td>
<td>Produce HTML documentation for the model.</td>
</tr>
<tr>
<td>Diagrams Only Report [Ctrl]+[Shift]+[F8]</td>
<td>Produce a diagrams only report (in RTF) for the model.</td>
</tr>
<tr>
<td>Delete Project Root</td>
<td>Delete the Model node and all subordinate Views and packages.</td>
</tr>
<tr>
<td>Help</td>
<td>Display the Help topic for the Project Browser.</td>
</tr>
</tbody>
</table>
3.5.5.1 Package Control Sub-Menu

To display the model node Package Control sub-menu, right-click on the node in the Project Browser and on the Package Control menu option.

From this menu you can:

- Configure various settings for the package
- Manage Baselines and compare them with the current package
- Create and work with the package XMI file
- Add model branches (package hierarchies) to version control, and export and import the file that manages changes to branches under version control
- Retrieve the latest version of the package from the repository; available only for packages that are checked in. Get All Latest is not intended for sharing .EAP files and should only be used when people have their own individual databases
- Display the Version Control Options dialog
- Provide a bulk update on the status of a package, this includes status options such as Proposed, Validate and Mandatory
- Set the namespace root for languages that support namespaces; for more information see the Namespaces topic.
### 3.5.6 Package Menu

Right-click on a View or Package in the Project Browser. The context menu displays, providing the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add-In</strong></td>
<td>Connect to external Add-Ins and external projects.</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>Add new packages to the model.</td>
</tr>
<tr>
<td><strong>Package Control</strong></td>
<td>Enables you to submit packages to package control and version control.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Add a new diagram, element or another package to the current package.</td>
</tr>
<tr>
<td><strong>Linked Document</strong></td>
<td>Create or display a linked document for the package or view.</td>
</tr>
<tr>
<td><strong>Paste Diagram</strong></td>
<td>If you have copied a diagram from another package, paste the diagram into the currently-selected package.</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>Produce a variety of reports and documentation in RTF format.</td>
</tr>
<tr>
<td><strong>Code Engineering</strong></td>
<td>Perform Code Engineering functions.</td>
</tr>
<tr>
<td><strong>Build and Run</strong></td>
<td>Build, run and debug functions.</td>
</tr>
<tr>
<td><strong>Import/Export</strong></td>
<td>Import and export using XMI text files.</td>
</tr>
<tr>
<td><strong>Transform Current Package</strong></td>
<td>Perform a model transformation on the selected package.</td>
</tr>
<tr>
<td><strong>Show Level Numbering</strong></td>
<td>Add a sequence number to each element in the package, based on the element's position in the package hierarchy. For nested elements, the numbering indicates level; that is: 3.2 3.2.1 3.2.1.1. This option is only available for packages, and the numbering only applies to the elements in the package, not diagrams.</td>
</tr>
<tr>
<td><strong>Show Element List</strong></td>
<td>Display the Element List showing the elements contained in the selected package.</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>Reorganize the package contents after making changes.</td>
</tr>
<tr>
<td><strong>Bookmarks</strong></td>
<td>Bookmark all elements in the selected folder.</td>
</tr>
<tr>
<td><strong>Find in Project Browser</strong></td>
<td>Search the Project Browser for specific elements.</td>
</tr>
<tr>
<td><strong>Copy Node Path to Clipboard</strong></td>
<td>Copy the selected package hierarchy structure to the clipboard.</td>
</tr>
<tr>
<td><strong>Save Package as UML Profile</strong></td>
<td>Save the selected package as a Profile.</td>
</tr>
<tr>
<td><strong>Set View Icon</strong></td>
<td>Change the display icon for the selected package.</td>
</tr>
<tr>
<td><strong>Move up</strong></td>
<td>Move the package up in the list.</td>
</tr>
<tr>
<td><strong>Move down</strong></td>
<td>Move the package down the list.</td>
</tr>
<tr>
<td><strong>Delete &lt;packagename&gt;</strong></td>
<td>Delete the selected package and its contents.</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td>Display the Help topic for the Project Browser.</td>
</tr>
</tbody>
</table>
### 3.5.6.1 Add Sub-Menu

In the Project Browser, select the **Package | Add** context menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Diagram</td>
</tr>
<tr>
<td>Add Element [Ctrl]+[M]</td>
</tr>
<tr>
<td>Add Package [Ctrl]+[W]</td>
</tr>
<tr>
<td>Add Model using Wizard</td>
</tr>
</tbody>
</table>

### 3.5.6.2 Documentation Sub-Menu

In the Project Browser, select the **Package | Documentation** context menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rich Text Format (RTF) Report [F8]</td>
</tr>
<tr>
<td>HTML Report [Shift]+[F8]</td>
</tr>
<tr>
<td>Diagrams Only Report [Ctrl]+[Shift]+[F8]</td>
</tr>
<tr>
<td>Testing Report [Ctrl]+[Shift]+[F8]</td>
</tr>
<tr>
<td>Open in Relationship Matrix [F8]</td>
</tr>
<tr>
<td>RTF Report Options [F8]</td>
</tr>
<tr>
<td>Copy RTF Bookmark</td>
</tr>
<tr>
<td>Implementation Report</td>
</tr>
<tr>
<td>Dependency Report</td>
</tr>
<tr>
<td>Testing Details</td>
</tr>
<tr>
<td>Resource Allocation</td>
</tr>
<tr>
<td>Package Metrics</td>
</tr>
</tbody>
</table>

### 3.5.6.3 Code Engineering Sub-Menu

In the Project Browser, select the **Package | Code Engineering** context menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Source Code [Ctrl]+[Alt]+[K]</td>
</tr>
<tr>
<td>Import Source Directory [Ctrl]+[Shift]+[U]</td>
</tr>
<tr>
<td>Import Binary Module</td>
</tr>
<tr>
<td>Synchronize Package With Code [Ctrl]+[Alt]+[M]</td>
</tr>
<tr>
<td>Generate DDL</td>
</tr>
<tr>
<td>Import DB schema from ODBC</td>
</tr>
<tr>
<td>Generate XML Schema</td>
</tr>
<tr>
<td>Import XML Schema</td>
</tr>
</tbody>
</table>
### 3.5.6.4 Build and Run Sub-Menu

In the Project Browser, select the Package | Build and Run context menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generate WSDL</strong> [Ctrl]+[F12]</td>
</tr>
<tr>
<td><strong>Import WSDL</strong> [Ctrl]+[F12]</td>
</tr>
<tr>
<td><strong>Reset Options for this Package</strong> [F12]</td>
</tr>
<tr>
<td><strong>Reset DBMS Options</strong></td>
</tr>
<tr>
<td><strong>Set as Namespace Root</strong> / Clear Namespace Root</td>
</tr>
</tbody>
</table>

#### Menu Option & Function Keys

| Package Build Scripts [Ctrl]+[Shift]+[F12] |
| Build [Ctrl]+[Shift]+[F12] |
| Test [Ctrl]+[Alt]+[T] |
| Run [Ctrl]+[Alt]+[N] |
| Deploy [Ctrl]+[Shift]+[Alt]+[F12] |
| **Debug Run** [F6] |
| **Step Into** [Shift]+[F6] |
| **Step Over** [Alt]+[F6] |
| **Step Out** [Ctrl]+[F6] |
| **Debug Stop** [Ctrl]+[Alt]+[F6] |
| **Start Debug Recording** |
| **Stop Debug Recording** |
| **Create Sequence Diagram** |

### 3.5.6.5 Import/Export Sub-Menu

In the Project Browser, select the Package | Import/Export context menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Import package</strong> [Ctrl]+[Alt]+[I] from XMI file</td>
</tr>
<tr>
<td><strong>Export package</strong> [Ctrl]+[Alt]+[E] to XMI file</td>
</tr>
<tr>
<td><strong>CSV Import</strong> / <strong>Export</strong></td>
</tr>
</tbody>
</table>

### 3.5.6.6 Contents Sub-Menu

In the Project Browser, select the Package | Contents context menu option.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expand Branch</strong></td>
<td>Expand all of the items in the Project Browser.</td>
</tr>
<tr>
<td>Menu Option</td>
<td>Use to</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Collapse Branch</td>
<td>Collapse all of the items in the Project Browser.</td>
</tr>
<tr>
<td>Reset Sort Order</td>
<td>Return sorting of package contents to list in alphabetical order.</td>
</tr>
<tr>
<td>Reload current package</td>
<td><a href="#">Refresh the current package</a> in the Project Browser.</td>
</tr>
</tbody>
</table>
3.5.7 Element Menu - Project Browser

Right-click on an element (such as Class, Object, Activity, State) in the Project Browser to open the element's context menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>View and modify the element properties.</td>
</tr>
<tr>
<td>Custom Properties [Ctrl]+[Shift]+[Enter]</td>
<td>Customize the properties.</td>
</tr>
<tr>
<td>Add</td>
<td>Create a child element and diagram (Classifier elements) or a connector to another element.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Display the Attribute dialog ready to create a new attribute.</td>
</tr>
<tr>
<td>Operations</td>
<td>Display the Operations dialog ready to create a new operation.</td>
</tr>
<tr>
<td>Synchronize with Code [F7]</td>
<td>Synchronize the element in the diagram with the source code. See Reverse Engineer and Synchronizing.</td>
</tr>
<tr>
<td>View Source Code [F12]</td>
<td>View the source code files.</td>
</tr>
<tr>
<td>Open Source Directory [Ctrl]+[Alt]+[Y]</td>
<td>Open the source directory.</td>
</tr>
<tr>
<td>In Diagrams [Ctrl]+[U]</td>
<td>Locate the element in all open diagrams.</td>
</tr>
<tr>
<td>Locate in Current Diagram</td>
<td>Select the element in the current visible diagram.</td>
</tr>
<tr>
<td>Copy RTF Bookmark</td>
<td>Copy a bookmark in RTF format to the clipboard.</td>
</tr>
<tr>
<td>Linked Document [Ctrl]+[Alt]+[D]</td>
<td>Create a Linked Document (Corporate edition only). See the Linked Documents topic.</td>
</tr>
<tr>
<td>Add Custom Reference [Ctrl]+[J]</td>
<td>Set up cross references between elements in a diagram and the selected element in the Project Browser.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Move the element up in the list of elements within this package.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Move the element down in the list of elements within this package.</td>
</tr>
<tr>
<td>Delete ’&lt;element Name&gt;’</td>
<td>Delete the element.</td>
</tr>
<tr>
<td>Help</td>
<td>Display the Help topic for the Project Browser.</td>
</tr>
</tbody>
</table>

3.5.7.1 Add Sub Menu

To display the Add submenu, either click on the element and press [Insert], or right-click on the element and select the Add context menu option.

The Add sub-menu enables you to:

- Add a Behavior element (Activity or Interaction or State Machine) and one of its associated diagrams to the selected Classifier element.
- Create a diagram to explain or expand on the selected Classifier element, using the New Diagram dialog, or
- Create a connector to another element.

Elements such as Actors, Classes and Activities can define a large amount of information that can be conveniently represented by or expanded in a child diagram. The Add sub-menu for these elements provides all of the options listed above.

Other elements, such as Timing, Exit and History have much more specific functions that do not require expansion. Therefore, the Add sub-menu for these elements only provides the option to create a connector to another element, and does not offer options for adding child elements and diagrams.
3.5.8 Diagram 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.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties [F5]</td>
<td>View and modify a diagram's properties.</td>
</tr>
<tr>
<td>Open</td>
<td>Open the diagram in the Diagram View.</td>
</tr>
<tr>
<td>Show Element List</td>
<td>Display the Element List listing the elements in the selected diagram.</td>
</tr>
<tr>
<td>Copy Diagram</td>
<td>Copy the diagram for pasting into another location (see Copy a Diagram).</td>
</tr>
<tr>
<td>Copy RTF Bookmark</td>
<td>Copy a bookmark in RTF format to the clipboard.</td>
</tr>
<tr>
<td>Add Custom Reference</td>
<td>Add this diagram as a cross reference to other elements.</td>
</tr>
<tr>
<td>Move up</td>
<td>Move the diagram up in the list of diagrams within this package.</td>
</tr>
<tr>
<td>Move down</td>
<td>Move the diagram down in the list of diagrams within this package.</td>
</tr>
<tr>
<td>Delete '&lt;diagram name&gt;'</td>
<td>Delete the diagram.</td>
</tr>
<tr>
<td>Help</td>
<td>Display the Help topic for the Project Browser.</td>
</tr>
</tbody>
</table>
3.5.9 Operation Menu - Project Browser

To display the Operation (or Method) context menu, right-click on an Operation in the Project Browser.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronize With Code [F7]</td>
<td>Synchronize the operation with the code.</td>
</tr>
<tr>
<td>View Source Code [F12]</td>
<td>Open the Source Code Viewer and display the operation.</td>
</tr>
<tr>
<td>Operation Properties</td>
<td>Display the Properties dialog for the operation.</td>
</tr>
<tr>
<td>Delete Operation</td>
<td>Delete the operation.</td>
</tr>
<tr>
<td>Generate Sequence Diagram [Ctrl]+[Shift]+[F6]</td>
<td>See the Record Debug Session For Method topic.</td>
</tr>
<tr>
<td>Help</td>
<td>Display the Help topic for the Project Browser.</td>
</tr>
</tbody>
</table>
3.6 The Main Menu

The Enterprise Architect 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
- Edit menu
- View menu
- Project menu
- Diagram menu
- Element menu
- Tools menu
- Add-ins menu
- Settings menu
- Window menu
- Help menu.

The above topics provide an overview of the functions available from the Main menu, and their general purposes.
3.6.1 The File Menu

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

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Project [Ctrl]+[N]</td>
<td>Create a new Enterprise Architect project</td>
</tr>
<tr>
<td>Open Project [Ctrl]+[O]</td>
<td>Open a project</td>
</tr>
<tr>
<td>Open Source File [Ctrl]+[Alt]+[O]</td>
<td>Open a source file.</td>
</tr>
<tr>
<td>Close Project</td>
<td>Close the current project.</td>
</tr>
<tr>
<td>Save Project As</td>
<td>Save the current model with a new name, or create a desktop shortcut to the current model. (This option is also active in the 'Lite', read-only version of Enterprise Architect.)</td>
</tr>
<tr>
<td>Reload Current Project [Ctrl]+[Shift]+[F11]</td>
<td>Reload the current project (use in a multi-user environment to refresh the Project Browser).</td>
</tr>
<tr>
<td>Print Setup</td>
<td>Configure your printer's settings.</td>
</tr>
<tr>
<td>Page Setup</td>
<td>Configure the page settings for printing.</td>
</tr>
<tr>
<td>Print Preview</td>
<td>Preview how the currently displayed diagram prints.</td>
</tr>
<tr>
<td>Print [Ctrl]+[P]</td>
<td>Print the currently displayed diagram. Enterprise Architect provides facilities to change the scale of the printed diagram (the number of pages it takes up) and to print or omit page headers and footers.</td>
</tr>
<tr>
<td>Recent Files List</td>
<td>Select from a list of the most recently opened projects.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exit Enterprise Architect.</td>
</tr>
</tbody>
</table>

3.6.1.1 Print Preview

When you select the File | Print Preview menu option, the display initially shows the first two pages on one screen, with no scroll bar. To toggle between this two-page display and a single-page display, click on the icon in the preview screen toolbar. In either mode, you can use the forward and back arrows to scroll through the pages of the diagram.

To display more than two pages on one screen, up to a maximum of ten pages, click on the Zoom Out button in the preview screen toolbar. The screen now includes the horizontal and vertical scroll bars, which you can also use to scroll through the pages of the diagram.
3.6.1.2 Save Model Copy or Shortcut

Enterprise Architect enables you to create a desktop shortcut (or Proxy file) to your model or (for a .EAP file) a direct copy of your model (you cannot create a copy of a DBMS model).

If you are using a database repository other than MS Access, you can configure the shortcut to encrypt the password used to set up the connection between Enterprise Architect and the repository. The Enterprise Architect user does not have the real password, thereby preventing them from accessing the repository using other tools such as Query Analyzer or SQLPlus.

Each shortcut is a file containing the connection string for the model. However, the shortcut also defines views that Enterprise Architect should open as it opens the model, such as:

- The Model Search with a specific text string and search type

**Notes:**

- For searches operating on the current tree selection, a diagram in the target package must be opened first.
- If you use a custom SQL search, the SQL must include `ea_guid AS CLASSGUID` and the object type.

- A specific diagram
- The Relationship Matrix with a saved profile
- The default Discussion Forum

You can define more than one diagram to open (but not more than one search, Discussion Forum or Relationship Matrix profile). Enterprise Architect opens the appropriate windows in the sequence in which you list the options, displaying the last view in the list. For example, you might create your shortcut to open, in sequence:

- A Development module
- The Model Search for a simple search on the term Issue
- The module Issues diagram
- The module Changes diagram.

The project would then open with the Enterprise Architect work area showing the two diagram tabs and the
Model Search tab, and with the Changes diagram displayed in the Diagram View.

Notes:
- These options are not valid for a copy of the model.
- If specified, the shortcut views override any default diagram defined for the model or current user.
- A shortcut does not affect the original Enterprise Architect .exe file or icon, or any other shortcut you might have defined. You can use all of these independently.
- When you use a shortcut to access a project that you have recently opened in Enterprise Architect, the Recent list on the Enterprise Architect Start Page has two entries for the project - one created when you opened the project in Enterprise Architect and one created when you used the desktop shortcut.

To create a copy of your model or a shortcut to your model, you have two options:
- Define each view to open (for example, if you are specifying a working environment in advance, perhaps for other users)
- Capture the current Enterprise Architect work environment which is useful if you want to access the model at exactly the same point in exactly the same environment when you resume work.

3.6.1.2.1 Create Copy Or Shortcut

You can specifically define each view that your model shortcut should open; for example, if you are specifying a working environment in advance, perhaps for other users.

You can also capture the current Enterprise Architect work environment, which is useful if you want to access the model at exactly the same point in exactly the same environment when you resume work.

To specifically set up your start-up shortcut or take a copy of the model, follow the steps below:

1. On the Start Page, open the required project.
2. Select the File | Save Project As menu option. The Save As dialog displays.

3. Click on the [...] (Browse) button at the end of the Target File field. The Save Project As dialog displays.
4. Browse for the appropriate file location (such as C:\Documents and Settings\<username>\Desktop) and, in the File name field, type an appropriate filename. All shortcuts are .EAP files, regardless of whether the model itself is a .EAP file or a DBMS model.
5. Click on the Save button to return to the Save As dialog.
6. Click on one of the following:
   - The Copy radio button to create a direct copy of the model, and click on the OK button to save the copy and end the procedure
   - The Shortcut radio button to create a desktop shortcut for the model
Note:
These radio buttons display only if the model is a .EAP file. If the model is a DBMS file, the target file can only be a shortcut. See the Encrypt Repository Password topic.

7. Click on the Add Other button and select the required option to define:
   - A diagram to open
   - A Relationship Matrix profile to open
   - The Project Discussion Forum
   - A Model Search to perform.
   The appropriate browser or dialog displays to define the view to display. Enter the details and click on the OK button.

8. Repeat step 7 for as many views as you require. Each entry is automatically selected, with a tick in the checkbox.

   To delete an entry, click on the checkbox to remove the tick. The entry is deleted when you save the shortcut.

9. To change the sequence and/or make a different view display first in the Diagram View, click on the appropriate entry and click on the 'Up Hand' or 'Down Hand' buttons.

10. Click on the OK button to save the shortcut. When you subsequently open the Save As dialog, it lists the currently-opened views in the order in which they were opened. You can add further views or remove them from the shortcut.

### 3.6.1.2.2 Capture Current Work Environment

To capture the current Enterprise Architect work environment so that you can access the model at exactly the same point in exactly the same environment when you resume work, follow the steps below:

1. Open Enterprise Architect (perhaps by using an existing shortcut).
2. On the Start Page, open the required project.
3. Perform the work you need to do.
4. When you decide to capture your work environment in a shortcut, ensure that you have opened all diagrams you require and, if necessary, the Discussion Forum, Model Search (with appropriate search term and type) and/or Relationship Matrix (at the appropriate profile). Ensure that the view you want to resume work on is the last one opened.
5. Select the File | Save Project As menu option. The Save As dialog displays, showing a list of actions derived from the views you currently have opened.
6. If you accessed Enterprise Architect via a shortcut, the Target File field displays the file location of that shortcut. If you intend to update the shortcut, go to step 10.

7. Otherwise, click on the [...] (Browse) button at the end of the Target File field. The Save Project As dialog displays.

8. Browse for the appropriate file location (such as C:\Documents and Settings\<username>\Desktop) and, in the File name field, type an appropriate filename. All shortcuts are .EAP files, regardless of whether the model itself is a .EAP file or a DBMS model.

9. Click on the Save button to return to the Save As dialog.

10. Click on one of the following:
    - The Copy radio button to create a direct copy of the model, and click on the OK button to save the copy and end the procedure.
    - The Shortcut radio button to create a desktop shortcut for the model.

    **Note:**
    These radio buttons display only if the model is a .EAP file. If the model is a DBMS file, the target file can only be a shortcut. See the Encrypt Repository Password topic.

11. If you decide not to have a view in the shortcut, click on the checkbox to remove the tick. The entry is deleted when you save the shortcut.

12. If you decide to change the sequence and/or make a different view display first in the Diagram View, click on the appropriate entry and click on the 'Up Hand' or 'Down Hand' buttons.

11. Click on the OK button to save the shortcut.

**Note:**
If you open the Save As dialog when no views are open, the Actions when model is opened field is empty. You can save the shortcut like this to totally clear it. Alternatively, if views are listed that you do not want to use again, click on the Include None button and save the shortcut.
3.6.1.2.3  Encrypt Repository Password

If your model is developed on a DBMS repository, the Save As dialog has an Encrypt Connection String check box instead of the radio buttons, as shown below:

You can create the shortcut actions as described in the Create Copy or Shortcut topic and, if necessary, select the checkbox to encrypt the database connection string. You distribute the shortcut file to the database users who are to access the model. The users then have an encrypted string such as:

EAConnectString: ora10_db --- DBType=3;ConnectEx+=wkI:E:B?e 52+H"e?r:pb_ZyAl3a[Vsfhh8p];Coi:d/bnX$5<("US"*GxvbbRtsK("%AwL4y1(P<je.%R1 ?AY:y?pw$%X%)_EwLWpKg7izLF=T
3.6.2 The Edit Menu

The Edit menu provides a range of functions to apply to diagram elements for the currently open diagram.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo [Ctrl]+[Z]</td>
<td>Undo the last action performed. (Note that some actions cannot be undone.)</td>
</tr>
<tr>
<td>Redo [Ctrl]+[Y]</td>
<td>Re-apply the last undone action.</td>
</tr>
</tbody>
</table>
| Copy [Ctrl]+[C]             | 1. Copy the selected elements to the MS Windows clipboard. To paste the selected elements, see the Paste Elements submenu.  
2. Copy an image of the selected elements to the clipboard. If no elements are selected, the entire diagram is copied.  
The image can be saved as a bitmap or a metafile; you set the format on the Options dialog. |
| Add to Project Clipboard [Ctrl]+[Space] | Add the current element to the Enterprise Architect clipboard. |
| Clear Project Clipboard     | Clear any elements from the Enterprise Architect clipboard. |
| Paste Element(s)            | Paste clipboard elements into current diagram. See the Paste Elements submenu. |
| Select All                  | Select all elements concurrently on the current diagram. |
| Select By Type              | Specify a particular type of element to select. |
| Clear Selection             | Deselect all elements. |
| Find in Model [Ctrl]+[F]    | Search the entire project for particular phrases or words. |
| Bookmark Selected [Shift]+[Space] | Bookmark the selected element(s). If the selected element is already bookmarked, this option removes the bookmark. |
| Clear All Bookmarks         | Clear bookmarks from any bookmarked elements in the current diagram. |
| Delete Selected Element(s) [Ctrl]+[D] | Delete the selected element from the diagram. |

3.6.2.1 Paste Elements Submenu

To paste what is in the buffer either as a new element or as a hyperlink to the element, select the Edit | Paste Element(s) menu option.

Note:
You can paste images from the Enterprise Architect Clipboard or the MS Windows clipboard. The Enterprise Architect clipboard takes precedence, so you must clear that clipboard before you can paste from the MS Windows clipboard.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
</table>
| as Link [Shift]+[Insert]    | Paste the element in the buffer as a link (i.e. a reference) to the element.  
If there are images in the MS Windows clipboard and none in the Enterprise Architect clipboard, you can:
  • Paste an image from the MS Windows clipboard into a new element as the appearance of the new element or  
  • Paste an image from the MS Windows clipboard into the diagram as a new boundary’s appearance. |
<p>| as New [Ctrl]+[Shift]+[V]   | Paste the element in the buffer as a completely new element. |</p>
<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste Image from Clipboard [Ctrl]+[Shift]+[Insert]</td>
<td>Paste the element in the Enterprise Architect Clipboard into the same diagram or a different diagram, as a metafile (that is, a definition of the element) as many times as is necessary. If you paste the element into a different diagram, the classifier identifies the source diagram for the element.</td>
</tr>
</tbody>
</table>
3.6.3 The View Menu

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

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Browser [Alt]+[0]</td>
<td>Show or hide the Project Browser window.</td>
</tr>
<tr>
<td>Properties [Alt]+[1]</td>
<td>Show or hide the Properties window.</td>
</tr>
<tr>
<td>System [Alt]+[2]</td>
<td>Show or hide the System window.</td>
</tr>
<tr>
<td>Testing [Alt]+[3]</td>
<td>Show or hide the Testing window.</td>
</tr>
<tr>
<td>Maintenance [Alt]+[4]</td>
<td>Show or hide the Maintenance window.</td>
</tr>
<tr>
<td>Toolbox [Alt]+[5]</td>
<td>Show or hide the Enterprise Architect UML Toolbox window.</td>
</tr>
<tr>
<td>Resources [Alt]+[6]</td>
<td>Show or hide the Resources window.</td>
</tr>
<tr>
<td>Debug Workbench [Alt]+[8]</td>
<td>Show or hide the Debug Workbench window.</td>
</tr>
<tr>
<td>Tasks Pane [Ctrl]+[Shift]+[9]</td>
<td>Show or hide the Tasks Pane window.</td>
</tr>
<tr>
<td>Notes [Ctrl]+[Shift]+[1]</td>
<td>Show or hide the Notes window.</td>
</tr>
<tr>
<td>Hierarchy [Ctrl]+[Shift]+[4]</td>
<td>Show or hide the Hierarchy window.</td>
</tr>
<tr>
<td>Tagged Values [Ctrl]+[Shift]+[6]</td>
<td>Show or hide the Tagged Values window.</td>
</tr>
<tr>
<td>Project Management [Ctrl]+[Shift]+[7]</td>
<td>Show or hide the Project Management window.</td>
</tr>
<tr>
<td>Output [Ctrl]+[Shift]+[8]</td>
<td>Show or hide the Output window.</td>
</tr>
<tr>
<td>Model Views [Ctrl]+[Shift]+[8]</td>
<td>Show or hide the Model Views window.</td>
</tr>
<tr>
<td>More Windows</td>
<td>Show or hide the Pan and Zoom window, Element Browser window, Relationships window, Rules and Scenarios window and web browser. See View Submenus.</td>
</tr>
<tr>
<td>Toolbars</td>
<td>Show or hide individual toolbars. See View Submenus.</td>
</tr>
<tr>
<td>Element List [Ctrl]+[Alt]+[R]</td>
<td>Display the current diagram or package in a context-sensitive, editable table, the Element List window.</td>
</tr>
<tr>
<td>Model Search [Ctrl]+[Alt]+[A]</td>
<td>Open the Enterprise Architect Model Search window and its facilities.</td>
</tr>
<tr>
<td>Relationship Matrix</td>
<td>Open the Relationship Matrix window to cross reference elements to each other by connector type.</td>
</tr>
<tr>
<td>Discussion Forum [Ctrl]+[Alt]+[U]</td>
<td>Open the Project Discussion Forum window.</td>
</tr>
<tr>
<td>Audit View</td>
<td>Display the Audit View window which shows the information that has been recorded by auditing.</td>
</tr>
<tr>
<td>Show Grid</td>
<td>Show or hide the diagram grid.</td>
</tr>
<tr>
<td>Snap to Grid</td>
<td>See View Submenus.</td>
</tr>
<tr>
<td>Visual Style</td>
<td>See View Submenus.</td>
</tr>
<tr>
<td>Visual Layouts</td>
<td>See View Submenus.</td>
</tr>
</tbody>
</table>
3.6.3.1 View Submenus

The More Windows Sub-Menu
Select the windows to be visible and deselect those to be hidden. You can select from:

- **Element Browser** [Alt]+[9]
- **Relationships** [Ctrl]+[Shift]+[2]
- **Rules and Scenarios** [Ctrl]+[Shift]+[3]
- **Web Browser** [Ctrl]+[Alt]+[W] - Open the web browser page at the site you have specified on the Options dialog, in the Web Home field.
- **Pan and Zoom** [Ctrl]+[Shift]+[N]

**Note:**
This sub-menu is a tear off menu.

The Toolbars Sub-Menu
Select the toolbars to be visible and deselect those to be hidden. You can select from:

- **Default Tools**
- **Project**
- **Code Generation**
- **UML Elements**
- **Diagram**
- **Current Element**
- **Current Connector**
- **Format Tool**
- **Status Bar**
- **Workspace Views**
- **Other Views**

**Note:**
This sub-menu is a tear off menu.

The Snap to Grid Sub-Menu
The **Snap to Grid** menu offers two options:

- **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 **Smart Placement** options are not enabled, the elements can be placed freely on the diagram.

The Visual Style Sub-Menu
Displays windows with the following **visual styles**:

- A flat XP look and feel
- The **2003** style look and feel
- The **2005** style look and feel
- The modern **2007** look and feel
- **Classic** Windows look and feel.

You can also use the:

- **Animate Autohide Windows** option to animate windows that have been automatically hidden.
- **Show Toolbar Customize Button** option to toggle the button on the end of the toolbar that enables you to customize the toolbar buttons, as shown below:
Hide Diagram Caption Bar option to hide or redisplay the diagram caption bar at the top or bottom of a diagram. The caption bar is illustrated below:

The Visual Layouts Sub-Menu

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

- Default Layout
- Custom Layouts...
- Iconix
3.6.4 The Project Menu

Use the Project menu for tasks related to the management of your project, such as recording issues, setting estimation parameters and compiling a glossary.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Package [Ctrl]+[W]</td>
<td>Create a new package</td>
</tr>
<tr>
<td>Add Diagram [Ctrl]+[Insert]</td>
<td>Create a new diagram in the current package.</td>
</tr>
<tr>
<td>Add Element [Ctrl]+[M]</td>
<td>Create a new element on the current diagram.</td>
</tr>
<tr>
<td>Documentation</td>
<td>See Documentation Submenu</td>
</tr>
<tr>
<td>Source Code Engineering</td>
<td>See Source Code Engineering Submenu</td>
</tr>
<tr>
<td>Build and Run</td>
<td>See Build and Run Submenu</td>
</tr>
<tr>
<td>Database Engineering</td>
<td>See Database Engineering Submenu</td>
</tr>
<tr>
<td>Model Transformations</td>
<td>See Model Transformations Submenu</td>
</tr>
<tr>
<td>Model Validation</td>
<td>See Model Validation Submenu</td>
</tr>
<tr>
<td>Web Services</td>
<td>See Web Services Submenu</td>
</tr>
<tr>
<td>XML Schema</td>
<td>See XML Schema Submenu</td>
</tr>
<tr>
<td>Security</td>
<td>See Security Submenu</td>
</tr>
<tr>
<td>Version Control</td>
<td>See Version Control Submenu</td>
</tr>
<tr>
<td>Import/Export</td>
<td>See Import/Export Submenu</td>
</tr>
<tr>
<td>Manage Baselines [Ctrl]+[Alt]+[B]</td>
<td>Store a model branch as a snapshot or baseline. Available in the Corporate edition only.</td>
</tr>
<tr>
<td>Use Case Metrics</td>
<td>Set Use Case Metrics to assist in estimating project size.</td>
</tr>
<tr>
<td>Project Statistics</td>
<td>View some basic project statistics.</td>
</tr>
</tbody>
</table>

3.6.4.1 Documentation Submenu

To generate various types of documentation, select the Project | Documentation menu option.

**Note:**

This sub-menu is a tear off menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML Report [Shift]+[F8]</td>
<td>Generate a report for the currently selected package in HTML format.</td>
</tr>
<tr>
<td>Diagrams Only Report</td>
<td>Generate an RTF report containing only diagrams.</td>
</tr>
<tr>
<td>Testing Report</td>
<td>Generate an RTF report of the model's existing tests.</td>
</tr>
<tr>
<td>Issues</td>
<td>Generate an RTF report of the model's issues.</td>
</tr>
<tr>
<td>Glossary</td>
<td>Generate an RTF report of the model's Glossary.</td>
</tr>
<tr>
<td>Implementation Details</td>
<td>Generate an implementation report for the currently-selected package.</td>
</tr>
<tr>
<td>Dependency Details</td>
<td>Generate a dependency report for the currently-selected package.</td>
</tr>
<tr>
<td>Testing Details</td>
<td>Generate test details for the currently-selected package.</td>
</tr>
</tbody>
</table>
## 3.6.4.2 Source Code Engineering Submenu

To forward and reverse engineer code using the language of your choice select the *Project | Source Code Engineering* menu option (Corporate and Professional editions only).

**Note:**

This sub-menu is a tear off menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Package Source Code [Ctrl]+[Alt]+[K]</td>
<td>Generate source code for the currently selected package.</td>
</tr>
<tr>
<td>Synchronize Package Contents [Ctrl]+[Alt]+[M]</td>
<td>Synchronize selected package with the source code.</td>
</tr>
<tr>
<td>Import Source Directory [Ctrl]+[Shift]+[U]</td>
<td>Import and reverse engineer an entire directory structure.</td>
</tr>
<tr>
<td>Import Binary Module</td>
<td>Import a binary module</td>
</tr>
<tr>
<td>Import ActionScript Files</td>
<td>Import code written in <em>ActionScript</em> with the file extension .AS.</td>
</tr>
<tr>
<td>Import C Files</td>
<td>Import code written in <em>ANSI C</em> with the file extension .C or .H.</td>
</tr>
<tr>
<td>Import C# Files</td>
<td>Import code written in the <em>C# programming language</em> with the file extension .CS.</td>
</tr>
<tr>
<td>Import C++ Files</td>
<td>Import code written in the <em>C++ programming language</em> with the file extension .H, .HPP or .HH.</td>
</tr>
<tr>
<td>Import Delphi Files</td>
<td>Import code written in the <em>Delphi programming language</em> with the file extension .PAS.</td>
</tr>
<tr>
<td>Import Java Files</td>
<td>Import code written in the <em>Java programming language</em> with the file extension .JAVA.</td>
</tr>
<tr>
<td>Import PHP Files</td>
<td>Import code written in <em>PHP</em> with the file extension .PHP, .PHP4, .INC.</td>
</tr>
<tr>
<td>Import Python Files</td>
<td>Import code written in <em>Python</em> with the file extension .PY.</td>
</tr>
<tr>
<td>Import Visual Basic Files</td>
<td>Import code written in the <em>Visual Basic programming language</em> with the file extension .FRM, .CLS, .BAS or .CTL.</td>
</tr>
<tr>
<td>Import VB.Net Files</td>
<td>Import code written in the <em>VB.Net programming language</em> with the file extension .VB.</td>
</tr>
</tbody>
</table>

## 3.6.4.3 Build and Run Submenu

To link your project with a compiler for building, running and debugging, select the *Project | Build and Run* menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Build Scripts [Shift]+[F12]</td>
<td>Create and configure <em>Package Build compiler scripts</em>.</td>
</tr>
</tbody>
</table>
### UML Modeling Tool Features | The Main Menu

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build [Ctrl]+[Shift]+[F12]</td>
<td>Build the application for your current script. Execute the Build command in the Package Build Scripts dialog.</td>
</tr>
<tr>
<td>Test [Ctrl]+[Alt]+[T]</td>
<td>Execute the Test command you configured in the Package Build Scripts dialog.</td>
</tr>
<tr>
<td>Run [Ctrl]+[Alt]+[N]</td>
<td>Execute the Run command you configured in the Package Build Scripts dialog.</td>
</tr>
<tr>
<td>Deploy [Ctrl]+[Shift]+[F12]</td>
<td>Execute the Deploy command you configured in the Package Build Scripts dialog.</td>
</tr>
<tr>
<td>Debug Run [F6]</td>
<td>Run the application and Debug the Run command in the Package Build Scripts dialog.</td>
</tr>
<tr>
<td>Debug Pause</td>
<td>Pause and restart execution of a debug run.</td>
</tr>
<tr>
<td>Step Into [Shift]+[F6]</td>
<td>Step into the current function.</td>
</tr>
<tr>
<td>Step Over [Alt]+[F6]</td>
<td>Step over the current function.</td>
</tr>
<tr>
<td>Step Out [Ctrl]+[F6]</td>
<td>Step out of the current function.</td>
</tr>
<tr>
<td>Debug Stop [Ctrl]+[Alt]+[F6]</td>
<td>Stop the current debug session.</td>
</tr>
<tr>
<td>Start Debug Recording</td>
<td>Start recording your trace for a debug session.</td>
</tr>
<tr>
<td>Stop Debug Recording</td>
<td>Stop a debug recording.</td>
</tr>
<tr>
<td>Auto Record Thread</td>
<td>Autorecord your debug session. The Stack Trace History, Stack tab and Source Code Editor dynamically update to reflect the current execution sequence for the thread; Stack Trace Recording ends when the thread ends or when you click on the Stop button.</td>
</tr>
<tr>
<td>Show/Hide Execution</td>
<td>Display the executing code when a thread has encountered a breakpoint. The option presents the source code file in an editor window with the current line of code highlighted for the thread that has the current focus.</td>
</tr>
<tr>
<td>Create Sequence Diagram</td>
<td>Create a Sequence diagram from the Stack Trace History.</td>
</tr>
</tbody>
</table>

### 3.6.4.4 Database Engineering Submenu

Select the Project | Database Engineering menu option.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import DB Schema from ODBC</td>
<td>Import a database schema from an ODBC data source.</td>
</tr>
<tr>
<td>Generate Package DDL</td>
<td>Generate a Package DDL script to create the tables in the currently selected package.</td>
</tr>
</tbody>
</table>

### 3.6.4.5 Model Transformations Submenu

Select the Project | Model Transformations menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform Selected Elements [Ctrl]+[H]</td>
<td>Perform an MDA-style transformation to the currently selected elements.</td>
</tr>
<tr>
<td>Transform Current Package</td>
<td>Perform an MDA-style transformation to the currently selected package.</td>
</tr>
</tbody>
</table>
### 3.6.4.6 Model Validation Submenu

Select the **Project | Model Validation** menu option.

**Note:**

This sub-menu is a tear off menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate Selected <strong>[Ctrl]+[Alt]+[V]</strong></td>
<td>Validate a selected element, diagram or package from the <strong>Project Browser</strong>.</td>
</tr>
<tr>
<td>Cancel Validation</td>
<td>Cancel the validation process.</td>
</tr>
<tr>
<td>Configure</td>
<td>Configure the Validation rules from the list of available rules.</td>
</tr>
</tbody>
</table>

### 3.6.4.7 Web Services Submenu

Select the **Project | Web Services** menu option.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import WSDL</td>
<td><strong>Reverse engineer</strong> a Web Service Definition Language (WSDL) file as a UML Class model.</td>
</tr>
<tr>
<td>Generate WSDL</td>
<td><strong>Forward</strong> engineer a UML Class model to a WSDL file.</td>
</tr>
</tbody>
</table>

### 3.6.4.8 XML Schema Submenu

Select the **Project | XML Schema** menu option.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import XML Schema</td>
<td><strong>Reverse</strong> engineer a W3C XML Schema (XSD) file as a UML Class model.</td>
</tr>
<tr>
<td>Generate XML Schema</td>
<td><strong>Forward</strong> engineer a UML Class model to a W3C XML Schema (XSD) file.</td>
</tr>
</tbody>
</table>

### 3.6.4.9 Security Submenu

To configure security settings for your project, select the **Project | Security** menu option.

**Notes:**

- This feature is available in the Corporate edition only.
- This sub-menu is a tear off menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage Users</td>
<td>Add, modify and remove <strong>users</strong>, including maintaining permissions.</td>
</tr>
<tr>
<td>Manage Groups</td>
<td>Add, modify and remove security <strong>groups</strong>, including maintaining</td>
</tr>
</tbody>
</table>
### 3.6.4.10 Version Control Submenu

Select the **Project | Version Control** menu option.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Current Package [Ctrl]+[Alt]+[P]</td>
<td>Specify whether this package (and its children) is controlled and, if so, which file it is controlled through.</td>
</tr>
<tr>
<td>Version Control Settings</td>
<td>Specify the options required to connect to a Source Code Control (SCC) provider.</td>
</tr>
<tr>
<td>Work Offline</td>
<td>Toggle between ‘offline’ version control and online version control.</td>
</tr>
</tbody>
</table>

### 3.6.4.11 Import/Export Submenu

To perform import and export to XMI and CSV, select the **Project | Import/Export** menu options.

**Note:**
This sub-menu is a tear off menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import Package from XMI [Ctrl]+[Alt]+[I]</td>
<td>Import a package from an XMI (XML based) file.</td>
</tr>
<tr>
<td>Export Package to XMI [Ctrl]+[Alt]+[E]</td>
<td>Export the currently selected package to an XMI (XML based) file.</td>
</tr>
<tr>
<td>CSV Import/Export [Ctrl]+[Alt]+[C]</td>
<td>Import or Export information on Enterprise Architect elements in CSV format.</td>
</tr>
<tr>
<td>CSV Import/Export Specifications</td>
<td>Set up CSV import Export Specifications.</td>
</tr>
<tr>
<td>Batch XMI Export</td>
<td>Export a group of controlled packages in one action.</td>
</tr>
<tr>
<td>Batch XMI Import</td>
<td>Run a batch import of multiple packages.</td>
</tr>
</tbody>
</table>
3.6.5 The Diagram Menu

The Diagram menu enables you to save diagram images to file as well as configure diagram properties and options.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties [F5]</td>
<td>View and edit the <code>&lt;type&gt; Diagram: &lt;name&gt;</code> dialog for the current diagram.</td>
</tr>
<tr>
<td>Layout Diagram</td>
<td>Configure automatic diagram layout settings (not available for Behavioral diagrams).</td>
</tr>
<tr>
<td>Lock Diagram</td>
<td>Prevent the diagram from being edited.</td>
</tr>
<tr>
<td><strong>Note:</strong> This does not apply in the Corporate edition if security is enabled. In that case, see Lock Model Elements.</td>
<td></td>
</tr>
<tr>
<td>Save [Ctrl]+[S]</td>
<td>Save the current position of all diagram elements.</td>
</tr>
<tr>
<td>Save Image [Ctrl]+[T]</td>
<td>Save the diagram as a bitmap (.BMP), GIF (.GIF) or Windows Metafile (.WMF).</td>
</tr>
<tr>
<td>Copy Image [Ctrl]+[B]</td>
<td>Copy an image of the current diagram to the clipboard. The image can be in metafile or bitmap format; you set the format on the Options dialog.</td>
</tr>
<tr>
<td>Save UML Pattern</td>
<td>Save the current diagram as a UML pattern.</td>
</tr>
<tr>
<td>Swimlanes and Matrix</td>
<td>Add, modify and delete swimlanes or the swimlanes matrix for the current diagram.</td>
</tr>
<tr>
<td>Visible Relations [Ctrl]+[I]</td>
<td>Hide or show individual connectors for the current diagram.</td>
</tr>
<tr>
<td>Property Note</td>
<td>Display the properties note for the current diagram.</td>
</tr>
<tr>
<td>Sequence Messages</td>
<td>Change the order of the communication messages in the current diagram.</td>
</tr>
<tr>
<td>Find in Project Browser</td>
<td>Locate the current diagram in the Project Browser window.</td>
</tr>
<tr>
<td>Make User Default</td>
<td>In the Corporate edition of Enterprise Architect, if security is enabled you make the current diagram the default diagram opened when the current user re-opens this model. The User Default diagram overrides the Model Default diagram (see Make Model Default, below).</td>
</tr>
<tr>
<td>To cancel a User Default diagram, either:</td>
<td></td>
</tr>
<tr>
<td>• Create a dummy diagram, set it as the User Default and delete it, or</td>
<td></td>
</tr>
<tr>
<td>• Delete the original diagram (if it is no longer relevant).</td>
<td></td>
</tr>
<tr>
<td>This still blocks the Model Default diagram, whilst Security is enabled. To re-establish the Model Default diagram, set it as the User Default.</td>
<td></td>
</tr>
<tr>
<td>Make Model Default</td>
<td>Make the current diagram the default diagram opened when the current model is re-opened.</td>
</tr>
<tr>
<td>To cancel a Model Default diagram, either:</td>
<td></td>
</tr>
<tr>
<td>• Create a dummy diagram, set it as the Model Default and delete it, or</td>
<td></td>
</tr>
<tr>
<td>• Delete the original diagram (if it is no longer relevant).</td>
<td></td>
</tr>
<tr>
<td>Change Type</td>
<td>Change the type of the current diagram.</td>
</tr>
<tr>
<td>Repeat Last Element [Shift]+[F3]</td>
<td>Create an instance of the same type as the last element created.</td>
</tr>
<tr>
<td>Repeat Last Connector [F3]</td>
<td>Create an instance of the same type as the last connector created.</td>
</tr>
<tr>
<td>Zoom</td>
<td>Change the zoom factor on the current diagram.</td>
</tr>
</tbody>
</table>
3.6.6 The Element Menu

You can configure and access element details using the Element menu. This enables you to control element layout, generate documentation and manage project resources.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties [Alt]+[Enter]</td>
<td>View the Properties dialog of the selected element.</td>
</tr>
<tr>
<td>Add Tagged Value [Ctrl]+[Shift]+[T]</td>
<td>Add a Tagged Value to the currently selected element.</td>
</tr>
<tr>
<td>Linked Document [Ctrl]+[Alt]+[D]</td>
<td>Link the element to a rich text document.</td>
</tr>
<tr>
<td>Attributes [F9]</td>
<td>View and edit the attributes for the selected element.</td>
</tr>
<tr>
<td>Operations [F10]</td>
<td>View and edit the operations for the selected element.</td>
</tr>
<tr>
<td>Inline Features</td>
<td>See the element Inline Features Submenu.</td>
</tr>
<tr>
<td>Feature Visibility [Ctrl]+[Shift]+[Y]</td>
<td>Select which features and characteristics of the selected element are visible on a diagram.</td>
</tr>
<tr>
<td>Advanced</td>
<td>See the element Advanced Submenu.</td>
</tr>
<tr>
<td>Rich Text Format (RTF) Report</td>
<td>Generate a report for the currently selected package in rich text format.</td>
</tr>
<tr>
<td>Source Code Engineering</td>
<td>See the element Source Code Engineering Submenu.</td>
</tr>
<tr>
<td>Open Source in External Editor [F12]</td>
<td>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.)</td>
</tr>
<tr>
<td>Find in Project Browser [Alt]+[G]</td>
<td>Find the currently selected element in the Project Browser window. (If no element is selected, finds the current diagram in the Project Browser window.)</td>
</tr>
<tr>
<td>Find in Diagrams [Ctrl]+[U]</td>
<td>Display all occurrences of the currently selected element.</td>
</tr>
<tr>
<td>Custom References [Ctrl]+[J]</td>
<td>Show model element cross references.</td>
</tr>
<tr>
<td>Appearance</td>
<td>See the element Appearance Submenu.</td>
</tr>
<tr>
<td>Alignment</td>
<td>See the element Position Submenus.</td>
</tr>
<tr>
<td>Make Same</td>
<td>See the element Position Submenus.</td>
</tr>
<tr>
<td>Z Order</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Space Evenly</td>
<td></td>
</tr>
</tbody>
</table>

3.6.6.1 Inline Features Submenu

The Inline Features sub-menu provides various options to directly edit Class diagram model elements from the Class diagram. Select the Element | Inline Features menu option to access this submenu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Selected [F2]</td>
<td>Attach a note or attach a constraint to the element.</td>
</tr>
<tr>
<td>View Properties</td>
<td>Open the dialog containing details of the selected element feature, or the element if no feature is selected.</td>
</tr>
<tr>
<td>Insert New After Selected</td>
<td>Insert a new item in the element.</td>
</tr>
</tbody>
</table>
### 3.6.6.2 Advanced Submenu

The Advanced sub-menu provides various options to choose from to customize the appearance of model elements. Select the Element | Advanced menu option to display this submenu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overrides &amp; Implementations [Ctrl]+[Shift]+[O]</td>
<td>Automatically override methods from parent Classes and from realized interfaces.</td>
</tr>
<tr>
<td>Set Parents and Interfaces [Ctrl]+[I]</td>
<td>Manually set the element's parents or an interface that it realizes.</td>
</tr>
<tr>
<td>Embedded Elements [Ctrl]+[Shift]+[B]</td>
<td>Attach elements to the currently selected element.</td>
</tr>
<tr>
<td>Change Type</td>
<td>Change the element type of the selected element.</td>
</tr>
</tbody>
</table>

### 3.6.6.3 Source Code Engineering Submenu

To forward and reverse engineer code using the language of your choice, select the Element | Source Code Engineering menu option.

**Note:**

This sub-menu is a tear off menu.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronize Current Element [F7]</td>
<td>Synchronize selected Class with source code.</td>
</tr>
<tr>
<td>Batch Generate Selected Element(s)</td>
<td>Batch generate source code for the currently selected element(s).</td>
</tr>
</tbody>
</table>
3.6.6.4 Appearance Submenu

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

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autosize [Alt]+[Z]</td>
<td>Auto-size a group of elements in a diagram to a best fit.</td>
</tr>
<tr>
<td>Default Appearance [Ctrl]+[Shift]+[E]</td>
<td>Set border, font and background color and border thickness for the selected element, as its default appearance.</td>
</tr>
<tr>
<td>Alternate Image [Ctrl]+[Shift]+[W]</td>
<td>Select an alternative image for the selected element.</td>
</tr>
<tr>
<td>Apply Image From Clipboard</td>
<td>Insert the image currently held on the clipboard.</td>
</tr>
<tr>
<td>Set Font</td>
<td>Change the font of the text displayed on the element in a diagram.</td>
</tr>
</tbody>
</table>

3.6.6.5 Position Submenus

These Element menu submenus enable you to size and position elements on the diagram, relative to each other.

The Alignment Sub-Menu

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

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left [Ctrl]+[Alt]+[Left]</td>
<td>Align left edges of elements.</td>
</tr>
<tr>
<td>Right [Ctrl]+[Alt]+[Right]</td>
<td>Align right edges of elements.</td>
</tr>
<tr>
<td>Top [Ctrl]+[Alt]+[Up]</td>
<td>Align top edges of elements.</td>
</tr>
<tr>
<td>Bottom [Ctrl]+[Alt]+[Down]</td>
<td>Align bottom edges of elements.</td>
</tr>
<tr>
<td>Centers</td>
<td>Align centers of elements, horizontally or vertically.</td>
</tr>
</tbody>
</table>

The Make Same Sub-Menu

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

The Z Order Sub-Menu

Use the Z Order sub-menu to move the selected element(s) back, forward, to the front of all other elements or behind all other elements.

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.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Across [Alt]+[-]</td>
<td>Space elements evenly, horizontally.</td>
</tr>
<tr>
<td>Down [Alt]+[=]</td>
<td>Space elements evenly, vertically.</td>
</tr>
</tbody>
</table>
3.6.7 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.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spell Check Project [Ctrl]+[F7]</td>
<td>Spell check the current project.</td>
</tr>
<tr>
<td>Spell Check Current Package [Ctrl]+[Shift]+[F7]</td>
<td>Spell check the current package.</td>
</tr>
<tr>
<td>Spelling Language</td>
<td>Specify the language to use for spell checking.</td>
</tr>
<tr>
<td>Data Management</td>
<td>Manage your project's data.</td>
</tr>
<tr>
<td>Manage .EAP File</td>
<td>Repair, compact or replicate your .EAP file.</td>
</tr>
<tr>
<td>Run Patch</td>
<td>Execute an SQL patch.</td>
</tr>
<tr>
<td>Export Reference Data</td>
<td>Export reference data to XML files for convenient model updating.</td>
</tr>
<tr>
<td>Import Reference Data</td>
<td>Import reference data from XML files for convenient model updating.</td>
</tr>
<tr>
<td>Import Technology</td>
<td>Import Technology file.</td>
</tr>
<tr>
<td>Generate MDG Technology File</td>
<td>Display the MDG Technology Wizard. See SDK for Enterprise Architect.</td>
</tr>
<tr>
<td>Wordpad</td>
<td>Open Wordpad.</td>
</tr>
<tr>
<td>Windows Explorer</td>
<td>Open Windows Explorer.</td>
</tr>
<tr>
<td>Customize</td>
<td>Customize the operation of Enterprise Architect.</td>
</tr>
<tr>
<td>Options [Ctrl]+[F9]</td>
<td>Customize your general settings through the Options dialog.</td>
</tr>
</tbody>
</table>

3.6.7.1 Data Management Submenu

Use the options on this submenu to manage your project's data.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Transfer</td>
<td>Move a complete project from one repository to another.</td>
</tr>
<tr>
<td>Note: You cannot move a project from a source .EAP file of a version earlier than 3.5.0.</td>
<td></td>
</tr>
<tr>
<td>Project Compare</td>
<td>Compare the total project sizes of two projects.</td>
</tr>
<tr>
<td>Project Integrity Check [Shift]+[F9]</td>
<td>Check the data integrity of a project.</td>
</tr>
</tbody>
</table>

3.6.7.2 Manage .EAP File Submenu

Use the options on this submenu to repair, compact or replicate your .EAP file.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repair .EAP File</td>
<td>Repair an Enterprise Architect project. If a project has not been closed properly, in rare cases it might not open correctly. This option attempts to repair such projects.</td>
</tr>
</tbody>
</table>
### The Customize Dialog

The Customize dialog enables you to customize Commands, Toolbars, Tools, Keyboard Keystrokes, Menus, and Options within Enterprise Architect.

#### Customize Dialog Example

<table>
<thead>
<tr>
<th>Commands:</th>
<th>Commands:</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Undo</td>
</tr>
<tr>
<td>Edit</td>
<td>Redo</td>
</tr>
<tr>
<td>View</td>
<td>Copy</td>
</tr>
<tr>
<td>Project</td>
<td>Add to Project Clipboard</td>
</tr>
<tr>
<td>Diagram</td>
<td>Clear Project Clipboard</td>
</tr>
<tr>
<td>Element</td>
<td>as Link</td>
</tr>
<tr>
<td>Tools</td>
<td>as New</td>
</tr>
<tr>
<td>AddIns</td>
<td>Paste to Fam Clipboard</td>
</tr>
<tr>
<td>Settings</td>
<td></td>
</tr>
<tr>
<td>Window</td>
<td></td>
</tr>
<tr>
<td>Help</td>
<td></td>
</tr>
<tr>
<td>New Menu</td>
<td></td>
</tr>
<tr>
<td>All Commands</td>
<td></td>
</tr>
</tbody>
</table>

**Description:** Copy all selected diagram objects to clipboard

---

© 1998-2008 Sparx Systems Pty Ltd
3.6.7.3.1 Customize Commands

The Customize window Commands tab provides access to many of Enterprise Architect’s functions, enabling you to place them into a toolbar.

To add a command to a toolbar, click on the category in the Categories: panel and select the command from the list for that category in the Commands: panel. Drag the command on top of the toolbar to add it to.

If you right-click on the command icon in the toolbar while the customize window is open, a context-sensitive menu displays. This menu offers options for deleting commands from a toolbar, and for changing the appearance of commands.

To remove a command from the toolbar, right-click on the command graphic or text and select the Delete menu option.

To change the appearance of a command graphic, right-click on the command graphic or text and select the Button Appearance... menu option. The Button Appearance dialog displays, which you can use to add graphical icons to commands that do not have them by default.

Note:

Some commands do not come with a convenient icon, which results 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 procedure of the Customize Toolbars topic.

3.6.7.3.2 Customize Toolbars

The Toolbars tab on the Customize window enables you to:

- Hide or show toolbars by selecting the appropriate checkbox
- Rename toolbars
- Create new toolbars
Delete toolbars
Modify toolbar contents by dragging commands from the Commands tab onto a visible toolbar
Reset a toolbar to its default contents and position, and
Display text labels under the Toolbar icons (perhaps temporarily, just to check what the icons do).

To access the Toolbars tab, either:

- On the Tools menu select Customize, or
- At the far right of any toolbar, click on the drop-down arrow and select the Customize option.

Create a New Toolbar and Populate it with Commands

To create a new toolbar and populate it with commands:

1. Select the Tools | Customize menu option. The Customize dialog displays.
2. Click on the Toolbars tab.
3. Click on the New button. The Toolbar Name dialog displays.
4. In the Toolbar Name field, type a name for your new toolbar and click on the OK button. Your new toolbar is created and shown 'floating' on the main screen (see the red circle below).
5. Now add commands to your toolbar. Click on the **Commands** tab. This forces the new toolbar behind the **Customize** dialog, so you might have to drag the **Customize** dialog to the side to find your new toolbar.

7. Find the command to add to your toolbar in the **Commands** list. The **Categories** list on the left represents the Enterprise Architect menu structure and the **Commands** list updates each time you click on a different category.

8. Drag the selected command from the list into the new toolbar. If you selected the **Show text labels** checkbox, your toolbar should now look like this:
If you did not select the **Show text labels** checkbox, your toolbar should look like this:

You can add as many commands to your toolbar as required. Your new toolbar behaves the same way as other toolbars; you can position it next to the other toolbars at the top of the application workspace, dock it to the side of the workspace or close it.

**3.6.7.3.3 Custom Tools**

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

**Add and Configure Custom Tools**

1. Select the **Tools | Customize** menu option. The **Customize** dialog displays.
2. Click on the **Tools** tab.
3. Click on the **New** icon (left of the red X). A blank field displays in the **Menu contents** list.
4. Type in the name of the tool as it should appear in the menu.

5. In the Command field, type the name of the tool.exe file to use; the tool must be a valid filename.

   **Note:**

   Programs installed with your operating system (e.g. Notepad, Wordpad) do not require a full file path. Programs installed separately (e.g. Microsoft Visual Studio) require the full file path in the Command field. If necessary, use the [...] (Browse) button to locate the tool in the file system (see Using Parameters).

6. Add any arguments required by the tool (see Opening External Tools and Passing Parameters to External Applications), and specify an initial directory if required.

7. Close the Customize dialog. Your tool should have now been added to the Tools menu as shown below.
3.6.7.3.3.1 Open External Tools

When configuring custom tools in Enterprise Architect, you can specify a file to be opened by the external application.

Select the Tools | Customize menu option. The Customize dialog displays; click on the Tools tab. Now you can:

- Specify a custom tool (application) using the Command field
- Define a 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 is c:\Temp.

Tip:

If there are any spaces in the paths in the Command, Arguments or Initial Directory fields, you must enclose the whole path in double quotes. For example: "c:\Temp\Customer Account.cls" must have quotes but c:\Temp\CustomerAccount.cls does not have to have quotes.

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; you can select this using the [...] (Browse) button to locate the VB executable. If you save from within VB the initial directory is c:\Temp.
3.6.7.3.3.2 Pass Parameters to Applications

When configuring custom tools in Enterprise Architect, you can pass parameters to the application. Select the Tools | Customize menu option. The Customize dialog displays; click on the Tools tab. Now you can:

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

The available parameters for passing information to external applications are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f</td>
<td>Project Name</td>
<td>For example, C:\projects\EAexample.eap.</td>
</tr>
<tr>
<td>$F</td>
<td>Calling Application (Enterprise Architect)</td>
<td>Enterprise Architect.</td>
</tr>
<tr>
<td>$p</td>
<td>Current Package ID</td>
<td>For example, 144.</td>
</tr>
<tr>
<td>$P</td>
<td>Package GUID</td>
<td>GUID for accessing this package.</td>
</tr>
<tr>
<td>$d</td>
<td>Diagram ID</td>
<td>Id for accessing associated diagram.</td>
</tr>
<tr>
<td>$D</td>
<td>Diagram GUID</td>
<td>GUID for accessing associated diagram.</td>
</tr>
<tr>
<td>$e</td>
<td>Comma separated list of element IDs</td>
<td>All elements selected in the current diagram.</td>
</tr>
<tr>
<td>$E</td>
<td>Comma separated list of element GUIDs</td>
<td>All elements selected in the current diagram.</td>
</tr>
</tbody>
</table>

Tip:
For more information on using the Automation Interface, visit [www.sparxsystems.com/AutIntVB.htm](http://www.sparxsystems.com/AutIntVB.htm).
3.6.7.3.4 Customize Keyboard

The Keyboard tab on the Customize window enables 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.

Modify Keyboard Shortcuts

To modify a keyboard shortcut, follow the steps below:

1. Select the Tools | Customize menu option. The Customize dialog displays.
2. Click on the Keyboard tab, and in the Category field click on the drop-down arrow and select the menu containing the command to modify.
3. In the Command field, click on the drop-down arrow and select the command. The current shortcut key (if any) for the command is displayed in the Current Keys field.
4. Move the cursor to the Press New Shortcut Key field and press the required shortcut key(s) for this command.

Note:

Press the actual keys to use. For example, to use [F5] press the [F5] key, don't type F then 5.
In the example above, the Assign button is disabled. This is because [F5] is already a shortcut to view diagram properties. If this occurs you must select a different shortcut key.

5. Once you have selected an available shortcut, click on the Assign button to apply the change. In the example below, the new shortcut is [N].

6. This has added a new shortcut so that both [N] and [Ctrl]+[N] create a new Enterprise Architect project.

   If you intend [N] to be the only shortcut for this action, select [Ctrl]+[N] in the Current Keys list and click on the Remove button.

   **Tip:**
   Remember that you can always revert to the default shortcut keys by clicking on the Reset All button.

   **Note:**
   Modified shortcut keys are stored in the registry, so they only affect the current user.
Warning:

About Custom Layouts and Keyboard Shortcuts: If you have set keyboard shortcuts, these are not 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 are 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.

3.6.7.3.5 Customize Menu

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

Application Frame Menus

Currently the Show Menus For feature is disabled as Enterprise Architect 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 list:

- None
- Unfold
- Slide
- Fade

Menu Shadows

Menu shadows can be toggled on or off by selecting or clearing the Menu shadows checkbox.
3.6.7.3.6 Customize Options

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

You can toggle the following options by selecting or clearing the checkboxes:

- Show Screen Tips on toolbars
- Show shortcut keys in Screen Tips
- Use Large Icons.
### 3.6.8 The Add-Ins Menu

The **Add-Ins** menu enables you to connect to, display information on, work with and manage your Add-Ins. The option displays only after you have installed an Add-In on your system.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect External Project</td>
<td>List external Add-Ins and, when you select one of them, open a project list for the Add-In. If there is only one active project available, the Add-In might automatically go on to open that project.</td>
</tr>
<tr>
<td>&lt;Add-In Name&gt;</td>
<td>Access the Add-In submenu for the selected Add-In.</td>
</tr>
<tr>
<td>Manage Add-Ins</td>
<td>Display the Manage Add-Ins dialog, which you use to enable or disable Add-Ins for use. Any technology loaded by an Add-In is automatically enabled; if you do not want to use it, you can disable it on the dialog.</td>
</tr>
</tbody>
</table>

### Add-In Submenu

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Add-In specific options&gt;</td>
<td>List options to perform various functions specific to the Add-In. For example, the MDG Technology For Zachman Framework, as an Add-In, has the options <strong>Open Example Model</strong> and <strong>Insert New Framework Model</strong>.</td>
</tr>
<tr>
<td>Help</td>
<td>Display the Help subsystem for the Add-In.</td>
</tr>
<tr>
<td>About</td>
<td>Display information on the Add-In installation, such as version, registration details and copyright statement.</td>
</tr>
</tbody>
</table>
### 3.6.9 The Settings Menu

The Settings menu enables 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.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>Display the <strong>People</strong> dialog, which enables you to configure the authors, clients, resources and roles for your project.</td>
</tr>
<tr>
<td>General Types</td>
<td>Display the <strong>General Types</strong> dialog, which enables you to configure requirements, status types, constraints and scenarios for your project.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Display the <strong>Maintenance</strong> dialog, which enables you to track problem types and test types.</td>
</tr>
<tr>
<td>Project Indicators</td>
<td>Define the project indicators (risks, efforts and metrics) used in Resource Management.</td>
</tr>
<tr>
<td>Estimation Factors</td>
<td>Display the <strong>Estimation factors</strong> dialog, which enables you to configure estimation factor types (Technical Complexity Factors, Environmental Complexity Factors and Default Hour Rate) for your project.</td>
</tr>
<tr>
<td>UML</td>
<td>Configure stereotypes, Tagged Value Types and the cardinality list for your project.</td>
</tr>
<tr>
<td>MDG Technologies</td>
<td>Display the <strong>MDG Technologies</strong> dialog, which enables you to load in and use MDG Technology files.</td>
</tr>
<tr>
<td>Namespaces</td>
<td>Locate and delete model namespaces.</td>
</tr>
<tr>
<td>Template Package</td>
<td>Configure or change the default element template directory.</td>
</tr>
<tr>
<td>Local Paths</td>
<td>Configure local directories and paths.</td>
</tr>
<tr>
<td>Auto Name Counters</td>
<td>Configure automatic naming for elements.</td>
</tr>
<tr>
<td>Code Datatypes</td>
<td>Add, modify and delete programming languages datatypes.</td>
</tr>
<tr>
<td>Database Datatypes</td>
<td>Add, modify and delete database datatypes.</td>
</tr>
<tr>
<td>Preprocessor Macros</td>
<td>Add and delete preprocessor macros.</td>
</tr>
<tr>
<td>Code Generation Templates</td>
<td>Modify code generation templates using the Code Templates Editor.</td>
</tr>
<tr>
<td>Transformation Templates</td>
<td>Modify transformation templates using the Transformation Templates Editor.</td>
</tr>
<tr>
<td>Images</td>
<td>Open the Image Manager.</td>
</tr>
<tr>
<td>Colors</td>
<td>Configure the custom colors for the project. There are two options:</td>
</tr>
<tr>
<td></td>
<td><strong>Get Project Custom Colors</strong> - get the custom colors</td>
</tr>
<tr>
<td></td>
<td><strong>Set Project Custom Colors</strong> - set the custom colors</td>
</tr>
</tbody>
</table>
|                            | Custom colors are as used in the Appearance dialog.
### 3.6.10 The Window Menu

The Window menu provides access to various actions related to configuring open windows.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Screen</td>
<td>Reset the display to full screen so that only the work area and main menu are shown - no toolbars or windows. To return to your normal working display, either click on the Full Screen option again or click on the Close Full Screen pop-up menu option. You can also restore individual menus and toolbars using the View menu options.</td>
</tr>
<tr>
<td>Close Active Window [Ctrl]+[F4]</td>
<td>Close the window that currently has focus.</td>
</tr>
<tr>
<td>Autohide Active Window [Ctrl]+[Shift]+[F4]</td>
<td>Autohide the window that currently has focus.</td>
</tr>
<tr>
<td>Autohide All Docked Windows</td>
<td>Autohide all windows that are docked.</td>
</tr>
<tr>
<td>Close Current View</td>
<td>Close the current view.</td>
</tr>
<tr>
<td>Close All Except Current</td>
<td>Close all except the currently selected view.</td>
</tr>
<tr>
<td>Reload Current View</td>
<td>Refresh the current view.</td>
</tr>
<tr>
<td>Save All Modified</td>
<td>Save all modified data.</td>
</tr>
<tr>
<td>Close All</td>
<td>Close all opened windows in the main tab view.</td>
</tr>
<tr>
<td>Always on Top</td>
<td>Force the main Enterprise Architect window to be on top of all other windows.</td>
</tr>
<tr>
<td>Set Focus to Current View [Ctrl]+[Shift]+[0]</td>
<td>Make the current view the active one, so that key strokes immediately act on that view.</td>
</tr>
</tbody>
</table>
3.6.11 The Help Menu

The Help menu provides access to the Enterprise Architect help files, the Read Me file, the Enterprise Architect End User License Agreement and various features on the Sparx Systems website.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>About EA</td>
<td>View information about Enterprise Architect, including your registration details.</td>
</tr>
<tr>
<td>Register and Manage License Key(s)</td>
<td>Configure and manage the license keys used to register the name and keys for Enterprise Architect and its Add-Ins. For more information see the License Management topic.</td>
</tr>
<tr>
<td>Configure Zicom Mentor</td>
<td>Register a third party Add-In for Enterprise Architect. For more information see the Zicom Mentor page on the Sparx Systems website.</td>
</tr>
<tr>
<td>Read Me</td>
<td>View the Readme.txt file, which details the changes and enhancements in Enterprise Architect, build by build.</td>
</tr>
<tr>
<td>View License Agreement</td>
<td>View the Enterprise Architect End User License Agreement.</td>
</tr>
<tr>
<td>Ordering Information</td>
<td>View information on how to purchase Enterprise Architect. (See the Introduction to Enterprise Architect.)</td>
</tr>
<tr>
<td>Help Contents</td>
<td>Display the introductory page of the Enterprise Architect Help.</td>
</tr>
<tr>
<td>Keyboard Accelerator Map</td>
<td>View the keyboard accelerator map. You can customize your keyboard shortcuts if required.</td>
</tr>
<tr>
<td>On-Line Resources</td>
<td>See below.</td>
</tr>
<tr>
<td>EA on the Web</td>
<td>Visit the Sparx Systems website.</td>
</tr>
</tbody>
</table>

The On-Line Resources Sub-Menu

Access help and resources on-line at the Sparx Systems website.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Forum and News</td>
<td>Visit the Enterprise Architect user discussion forum.</td>
</tr>
<tr>
<td>Request-a-Feature</td>
<td>Request a feature you would like to see in Enterprise Architect.</td>
</tr>
<tr>
<td>Bug Report Page</td>
<td>Report the details of a bug you have found in Enterprise Architect.</td>
</tr>
<tr>
<td>Latest Version Details</td>
<td>Display the Announcements folder of the User Forum, providing details of the latest Enterprise Architect build.</td>
</tr>
<tr>
<td>Introducing UML</td>
<td>Access the Sparx Systems online UML tutorials.</td>
</tr>
<tr>
<td>Pricing and Purchase Options</td>
<td>Purchase or upgrade Enterprise Architect over the internet.</td>
</tr>
</tbody>
</table>
3.7 The Enterprise Architect Toolbox

What is the Toolbox?

The Enterprise Architect UML Toolbox is a panel of icons that you use to create elements and connectors on a diagram. Within the Toolbox, related UML elements and connectors are organized into pages, each page containing the elements or connectors used for a particular type of diagram. The diagrams include standard UML diagrams, Enterprise Architect Extended diagrams, and any MDG Technologies and UML Profiles that you have added to Enterprise Architect. When you open a diagram, the Toolbox automatically provides the element and relationship pages corresponding to the diagram type. This does not prevent you using elements and connectors from other pages in a given diagram, though some combinations might not represent valid UML.

Display the Toolbox

To display the Toolbox, select the View | Toolbox menu option, or press [Alt]+[5].

The Toolbox can be docked on either side of the diagram, or free floated on top of the diagram to expose more surface for editing.

Tip:

You can also hide and show the whole Toolbox using the Hide/Show Toolbox button on the shortcut toolbar.

To display specific pages in the Toolbox, select the More Tools option at the top of the Toolbox, select the appropriate UML, Extended or customized Toolbox option, and select the diagram type from the menu.

In most cases, three pages display: <type> elements, <type> Relationships and Common. If you select the <default> option, you display only the Common page.
Create Elements and Connectors:

1. In the Project Browser, double-click on the icon against the required diagram. The diagram opens with the appropriate Toolbox pages for that diagram type. (If you want a different set of elements and connectors, click on More tools and select the appropriate diagram type as explained above).

2. Click on the required item; for example, the Class element or Associate relationship.

3. For element items, click anywhere on the diagram to place the new element.

4. For connector items, drag the cursor between the source and target elements on the diagram. The solid border of the elements changes to a hatched border as you pass the cursor over them, indicating the source and potential target elements. To add bends to the connector, press [Shift] as you change the drag direction of the cursor.

   Alternatively, drag from the source element to an empty area of the diagram; the Quick-linker enables you to create the target element.

5. Edit the element properties or connector properties, as required.

Tip:

If you are creating several elements of one type, after creating the first just press [Shift]+[F3] or [Ctrl]+click to create the next element of that type. For connectors, click on the source element and press [F3] to create another connector of the same type.

Notes:

- The Toolbox pages relate to specific UML diagrams.
- Dropping a Package element from the Toolbox into a diagram creates a new package in the Project Browser, and a default diagram of the same type as the current diagram.

Customize The Toolbox

You can customize the Toolbox pages by pinning them within the Toolbox, or by adding MDG Technologies and UML Profiles to the Toolbox.

Note:

Enterprise Architect provides Toolbox pages for the ICONIX, BPMN 1.4, Data Flow Diagrams and Mind Mapping technologies as part of the initial install.

You can also change other features of the Toolbox appearance - see UML Toolbox Appearance Options.
3.7.1 **UML Toolbox Appearance Options**

You can modify the appearance of the Enterprise Architect UML Toolbox pages in several ways, through the context menu. Right-click on the Toolbox page to display the menu.

- To hide the element or relationship labels (and subsequently redisplay them), select the **Hide Labels** or **Show Labels** context menu option. The icons in the page then 'wrap' within the page, without text labels.

- To 'pin' the page so that it displays for any group in the Toolbox, select the **Pin in Toolbox** menu option. (This is not available on the **Common** page, which displays in all groups anyway.) For example, if you 'pinned' the **Class** elements page, and switched to the **Communication** pages, the Toolbox would include a collapsed **Class** elements page underneath the **Communication** pages.

- To unpin the page so that it displays only in its own Toolbox group, right-click on it and select the **Unpin from Toolbox** context menu option.

- To collapse a page to just show the heading (type elements, type Relationships or Common), click on the 'minus' box at the left of the page heading. To expand the page again, click on the heading. Alternatively, collapse the page by right-clicking on the page and selecting the **Collapse** context menu option.

**Set Toolbox Visibility**

To tailor the Toolbox to list all pages you require at the same time, follow the steps below:

1. Click on the **Set Toolbox Visibility** context menu option. The **Visible Toolbox Pages** dialog displays.
2. By default, the dialog lists the *element* pages only, in the order: UML pages, Extended pages, MDG Technology pages. To include the corresponding *relationship* pages, select the **Show Relationship Pages** checkbox at the bottom of the dialog.

3. For each page to display on the *Toolbox*, select the **Visible** checkbox. Deselect the checkbox if you no longer require a page to be displayed.

4. When you have defined the list of pages to display, click on the **OK** button. The pages you have selected are pinned to the *Toolbox* in a collapsed state, underneath the current diagram-type pages.

5. To expand a page, click on the heading. You can remove a page individually by expanding it, right-clicking on it and selecting the **Unpin from Toolbox** context menu option.
**Note:**

MDG Technologies can impose their own Toolbox page visibility. For example, if ICONIX is the active technology, all seven ICONIX pages are automatically exposed in the Toolbox. If the active Technology pages duplicate UML or Extended pages (as the ICONIX pages do) then the pinned Technology pages override and replace the pinned UML and Extended pages.

For example, if ICONIX is active and you have pinned the Extended Analysis page, the Analysis page in the list is the ICONIX-defined page, not the Extended Analysis page.
3.7.2 UML Toolbox Shortcut Menu

To add elements and connectors into a diagram, you can access the UML Toolbox shortcut menu instead of employing the full Enterprise Architect UML Toolbox. The menu provides options to select:

- **Elements** specific to the current diagram type (*Timing* diagram in the example shown below)
- **Relationships** specific to the current diagram type
- Elements and relationships from any pages pinned in the Toolbox (*Object*, *Composite* and *Communication* in the example below)
- **Common** elements and relationships
- Elements and connectors for other diagram types.

<table>
<thead>
<tr>
<th>Other</th>
<th>Use Case</th>
<th>Activate Use Case Toolbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Lifeline</td>
<td>Class</td>
<td>Actor</td>
</tr>
<tr>
<td>Value Lifeline</td>
<td>Object</td>
<td>Use Case</td>
</tr>
<tr>
<td>Message Label</td>
<td>Composite</td>
<td>Collaboration</td>
</tr>
<tr>
<td>Message Endpoint</td>
<td>Communication</td>
<td>Boundary</td>
</tr>
<tr>
<td>Diagram Gate</td>
<td>Interaction</td>
<td>Package</td>
</tr>
<tr>
<td>Message</td>
<td>Timing</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>State</td>
<td></td>
</tr>
<tr>
<td>Composite</td>
<td>Activity</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Component</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deployment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Metamodel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extended</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPMN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Flow Diagrams</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ICONIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MindMapping</td>
<td></td>
</tr>
</tbody>
</table>

The advantage of using the UML Toolbox shortcut menu is that it provides an increased amount of the workspace to be used for diagramming rather than to display fixed (instead of pop-up) menus.

To use the UML Toolbox shortcut menu, follow the steps below:

1. Open a diagram.
2. Either:
   - Click on the diagram background and press [Insert] or [Spacebar]
   - Press and hold [Ctrl] and right-click on the diagram background.
   - The shortcut menu displays, listing the current diagram-type elements and connectors.
3. If necessary, select the Other option or a pinned Toolbox page option to list elements and connectors for a different diagram type.
4. Select the element or connector to include in the diagram. The object is added to the diagram.

If you select the Other context menu option, the final menu in the sequence offers the Activate <Type> Toolbox option. This opens and activates the corresponding page in the UML Toolbox, if the Toolbox is visible.
**Note:**

As for the UML Toolbox itself, if an MDG Technology:

- is active
- automatically pins Toolbox pages and
- duplicates UML or Extended pages

the pinned Technology pages override and replace the pinned UML or Extended pages in the initial UML Toolbox shortcut menu.
3.7.3 **Common Group**

The **Common** page of elements and relationships is displayed at the bottom of every other group. It contains the elements and relationships that can be used on any diagram.

**Toolbox Elements and Connectors**

**Note:**

Click on the elements and connectors below for information.

<table>
<thead>
<tr>
<th>Common Elements</th>
<th>Common Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>Dependency</td>
</tr>
<tr>
<td>Constraint</td>
<td>Realize</td>
</tr>
<tr>
<td>Document</td>
<td>Trace</td>
</tr>
<tr>
<td>Artifact</td>
<td>Information Flow</td>
</tr>
<tr>
<td>Requirement</td>
<td>Note Link</td>
</tr>
<tr>
<td>Issue</td>
<td></td>
</tr>
<tr>
<td>Change</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td></td>
</tr>
</tbody>
</table>
3.7.4 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 interacts with the system. Often Sequence diagrams are associated 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 is drawn as a stick figure, although the user could be another computer system or similar. A Use Case is a discrete piece of functionality the system provides that enables 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 is to provide to a user.

The connectors available include: associate (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 might realize some business requirement).

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.

**Note:**

Invokes and Precedes relationships are defined by the Open Modeling Language (OML).
3.7.5 Class Group

The Class group can be used for Package diagrams, Class diagrams, and Object diagrams, those that usually display elements concerned with the logical structure of the system. These include Objects, Classes and Interfaces. Logical models can include domain models (high level business driven object model) to strict development Class models (define inheritance, attributes, operations).

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 or Aggregation relationships. See the Class Model for an example of this.

Use the Table element to insert a stereotyped Class for use in database modeling. See the Data Modeling topic for more details.

To add an element to the current diagram, click on 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 on the required icon, click on the start element in the diagram and drag to the end element.
3.7.6 Object Group

The Object group is used to create Object diagrams. Object diagrams reflect multiplicity and the roles instantiated Classes could serve. They are useful in creating different cases in which relationships and Classes are applied.

The user is called an Actor and is drawn as a stick figure, although the user could be another computer system or similar.

An Object is an instance of a Class.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.7 Composite Group

The Composite group is used for Composite Structure diagrams, which 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 might not show.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.8 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, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.

**Note:**

Communication diagrams were know as Collaboration diagrams in UML 1.4.
3.7.9 Interaction Group

The Interaction group is used for Interaction diagrams (Sequence, Timing, Communication or Interaction Overview), 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 Sequence diagrams topic. The Message relationship is used to model the flow of information and processing between elements.

The following model objects are supported: Actor, Lifeline, Boundary, Control and Entity (all sequence elements) and Message (sequence relationship).

Note:

Messages can be simple or recursive calls.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.10 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, and time-driven business processes.

Timing diagrams can be used for defining hardware-driven or embedded software components, and 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.

A 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, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
### 3.7.11 State Group

The State group is used by State Machine diagrams to show the enableable states a Class or element might 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 Object Flow relation. State Machine diagrams are used to model the states or conditions that elements might be in at runtime, such as active, inactive, idle, accelerating or braking. States can have substates; for example, accelerate or brake might be substates of active.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.12 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 might 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 might 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 enable you to describe the dynamics of the system from the point of view of activities and flows between them. Activities can be stereotyped as a process to display a business process icon.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
### 3.7.13 Component Group

The **Component** group enables 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 might be deployed on a Windows 2000 Server (Node). See the [Deployment Diagram](#) topic for an example of this.

The **Component** group contains elements related to the actual building of the system: the components that 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, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.14 Deployment Group

The Deployment group enables 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 is to exist. For example, DLLs or Web Servers are Components that could be deployed on a Windows 2000 Server (Node). See the Deployment Diagram topic for an example of this.

The Deployment group contains elements related to the actual building of the system; the components that make up the system (e.g. ActiveX DLLs or Java beans) and the nodes those components run on, including the physical connections between nodes.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.15 Profile Group

The Profile group contains some extended UML elements that can be used to create and modify Profiles for rapidly creating stereotyped and Tagged Values that can be applied to structures such as elements, attributes, methods and connectors.

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 structures such as elements, attributes, methods, connectors and connector ends.

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.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
### 3.7.16 Metamodel Group

The **Metamodel** group enables you to create metamodel diagrams with support for MOF diagrams.

<table>
<thead>
<tr>
<th>Metamodel Elements</th>
<th>Metamodel Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Generalize</td>
</tr>
<tr>
<td>Class</td>
<td>Associate</td>
</tr>
<tr>
<td>Enumeration</td>
<td>Compose</td>
</tr>
<tr>
<td>Primitive</td>
<td>Aggregate</td>
</tr>
</tbody>
</table>

A **Package** is a namespace as well as an element that can be contained in other package's namespaces.

A **Class** is a representation of objects, that reflects their structure and behavior within the system.

An **Enumeration** is a Class with an enumeration stereotype.

A **Primitive** supports the MOF specification.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.17 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, and controller.

To add an element to the current diagram, click on 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 on the required icon, click on the start element in the diagram and drag to the end element.
3.7.18 **Custom Group**

The Custom group contains a few extended UML elements that might be of use in modeling or designing your system in a Custom diagram.

A **Package** is a namespace as well as an element that can be contained in other package's namespaces.

A **Requirement** is a custom element used to capture requirements outside of standard UML elements. A Requirement expresses required system behavior that can cross several Use Cases. You can connect Requirements to other elements using the Realize connector to express the implementation of a requirement and hence the traceability from user requirements to what is being built.

An **Issue** element is a structured comment that contains information about defects and issues relating to the system/model (see the Defects (Issues) topic). Affected elements are connected by Trace connectors.

An **Change** element is a structured comment that contains information about changes requested to the system/model (see the Changes topic). Affected elements are connected by Trace connectors.

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 **Test Case** element defines what must be set up in order to test a particular feature (see the Test Cases Window topic). It enables you to define a set of tests once for a number of elements, and provides greater visibility for tests.

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.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.19 Requirement Group

As an analysis step, often it is desirable to capture simple system requirements. These are eventually realized by Use Cases.

A Package is a namespace as well as an element that can be contained in other package's namespaces.

Specify the Requirement of a system. Note that there are a few different requirement types, as listed below.

- Display
- Functional
- Performance
- Printing
- Report
- Testing
- Validate

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.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.20 Maintenance Group

The Maintenance elements are defects, changes, issues and tasks.

A Package is a namespace as well as an element that can be contained in other package's namespaces.

An Issue element is a structured comment that contains information about defects and issues relating to the system/model. Affected elements are connected by Trace connectors.

A Change element is a structured comment that contains information about changes requested to the system/model. Affected elements are connected by Trace connectors.

A Test Case describes what must be set up in order to test a particular feature.

An Object is an instance of a Class.

To add an element to the current diagram, click on 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 on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.21 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 packages’ namespaces.

A Screen element represents a graphical user interface. You can place GUI elements onto the screen element.

UI Control 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 boxes.

An Object is an instance of a Class.

GUI Element Stereotype available

The following GUI elements are available as stereotypes.

The following diagram illustrates GUI elements with each of the stereotypes as they appear on a screen element.
Example GUI Diagram
3.7.22 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 a UML component. Its interfaces represent the **WSDL services**.

A WSDL **Port Type** is modeled as a UML interface. Its **Port Type Operations** are realized by **Binding** elements. Each of the operation parameters is derived from the Message elements defined in the **Messages** package.

To add an element to the current diagram, click on 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 on the required icon, click on the start element in the diagram and drag to the end element.
3.7.23 **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 a UML package, which contains the type and element definitions for a particular targetNamespace. Drag this item onto an open diagram to create the package to contain your schema model elements. The package is stereotyped as **XSD schema**.

Open the logical diagram created under the XSD schema package and add additional schema elements as required.

To add an element to the current diagram, click on the required icon and drag it into position on the diagram. Set the name and other properties as prompted.

To add a relationship, click on the required icon, then click on the start element in the diagram and drag to the end element.
3.7.24 Data Modeling Group

This group is used for database modeling and database design, in conjunction with the UML Data Modeling Profile.

The **Table** element defines a table on the data model.

The **View** element represents database views in the data model.

The **Procedure** element represents stored procedures in the data model.

To add an element to the current diagram, click on 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 on the required icon, click on the start element in the diagram and drag to the end element.
3.8 Workspace Toolbars

Enterprise Architect provides you with a selection of toolbars that you can drag and dock within the application frame. These toolbars provide convenient shortcuts to common tasks. You can also float toolbars over the application by tearing them off the application toolbar section; this is useful when you are using a certain set of functions a lot in a particular area.

You can customize toolbars by deleting and reordering the default button set. See Customize Commands for more information. You can customize which toolbars are active by right-clicking on the toolbar background and selecting the required items in the context menu.

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

The toolbars available include:
- Default Tools Toolbar
- Project Toolbar
- Code Generation Toolbar
- UML Elements Toolbar
- Diagram Toolbar
- Current Element Toolbar
- Current Connector Toolbar
- Format Toolbar
- Workspace Views
- Other Views Toolbar
- Status Bar
- Rich Text Notes Toolbar
3.8.1 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 [Ctrl]+[S]
- Copy to Enterprise Architect clipboard [Ctrl]+[Space]
- Paste from Enterprise Architect clipboard as instance [Shift]+[Insert]
- Undo last action [Ctrl]+[Z]
- Redo last undone action [Ctrl]+[Y]
- Print Preview (for generated documents and diagrams)
- Page setup
- Print [Ctrl]+[O]
- Show Element List for currently-selected package or diagram [Ctrl]+[Alt]+[R]
- Open Model Search [Ctrl]+[Alt]+[A]
- Select the layout of docked windows, toolbars and the Enterprise Architect UML Toolbox (<default> is Enterprise Architect, other options display for any MDG Technologies you have enabled)
- Help [F1].

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the View | Toolbars menu option.
3.8.2 Project Toolbar

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

- **Reload current project** \([\text{Ctrl}]+[\text{Shift}]+[\text{F11}]\)
- New diagram
- New package \([\text{Ctrl}]+[\text{W}]\)
- New element \([\text{Ctrl}]+[\text{M}]\)
- Search Project Browser window \([\text{Ctrl}]+[\text{Shift}]+[\text{F}]\)
- Search entire project \([\text{Ctrl}]+[\text{F}]\)
- New RTF document \([\text{F8}]\)
- Project issues
- Project glossary
- Options (preferences) \([\text{Ctrl}]+[\text{F9}]\)

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the View | Toolbars menu option.
3.8.3 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 [F11]
- Batch generate code for one or more selected Classes [Shift]+[F11]
- Synchronize selected Classes with source code [F7]
- View code in default editor [F12].

Set Default Code Language

To set the default language for the model click on the Default Language drop-down arrow and select the appropriate language.

Set Default Database

To set the default database type for modeling click on the Default Database drop-down arrow and select the appropriate database type.

Import Code

To select a language for code generation, click on the drop-down arrow for the Import button.
You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the **View | Toolbars** menu option.
3.8.4 UML Elements Toolbar

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

- Insert new System Boundary - swim lanes element
- Insert new Note
- Insert new Text element
- Insert new diagram note
- Insert diagram Legend
- Insert new hyperlink
- Insert new note link

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the View | Toolbars menu option.
3.8.5 Diagram Toolbar

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

- Align selected elements to the left: `[Ctrl]+[Alt]+[←]
- Align selected elements to the right: `[Ctrl]+[Alt]+[→]
- Align selected elements to the top: `[Ctrl]+[Alt]+[↑]
- Align selected elements to the bottom: `[Ctrl]+[Alt]+[↓]
- Bring selected element to top of Z order
- Move selected element to bottom of Z order
- Go to previous diagram: `[Alt]+[←]
- Go to next diagram: `[Alt]+[→]
- Go to default diagram
- Zoom In
- Zoom Out
- Zoom to fit diagram
- Zoom to fit page
- Zoom to 100%
- Auto-layout diagram (not for Behavioral diagrams)
- Show diagram properties: `[F5]
- Paste appearance as copied into the Painter from an element's Appearance context menu
- Delete selected element(s): `[Ctrl]+[D]

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show this toolbar from the View | Toolbars menu option.
3.8.6 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 [Ctrl]+[Shift]+[Y]
- Specify the run state of an element (or, for Parts, property value) [Ctrl]+[Shift]+[R]
- View use of element in other structures such as diagrams [Ctrl]+[U]
- Locate the element in the Project Browser window [Alt]+[G]
- View the cross reference list for this element [Ctrl]+[J]
- Lock or unlock the current element

Note:

This does not apply in the Corporate edition if security is enabled. In that case, see Lock Model Elements.

- Add a Tagged Value to the current element [Ctrl]+[Shift]+[T]

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar using the View | Toolbars menu option.
3.8.7 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
- 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).

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the View | Toolbars menu option.
3.8.8 Format Toolbar

Use the Format toolbar to change the appearance of a selected element (or several selected elements) in the current diagram; this does not affect any other occurrence of the selected elements anywhere else in the model.

Notes:

- To set the global appearance of all elements throughout a model, use the Options dialog. Select the Tools | Options menu option, then select Standard Colors from the options tree.
- To override the default appearance of a selected element (or several selected elements) on all diagrams on which it occurs, right-click on the element and select the Appearance | Default Appearance context menu option. The Default Appearance dialog displays.

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

- Fill Color (drop-down color palette)
- Text Color (drop-down color palette)
- Border or Connector Line Color (drop-down color palette)
- Border or Connector Line Width (arrows increase/decrease between 1 and 5)
- Apply Style to Element(s)
- Copy Style from Element
- Style list for selecting saved styles
- Save style (see below).

Note:

If the Format toolbar is not displayed, select the View | Toolbars | Format Tool menu option.

If you click on the drop-down arrow for the Save Style (pencil) button, you can select an option from the following list:

Save as New Style
Save Style
Refresh List
Delete Current Style

The Fill Color button can be used in conjunction with the Project Custom Colors menu options to enable users to have access to custom-defined project colors. To activate this feature select the Tools | Options | Standard Colors menu option and ensure that the Show Project Custom Colors in Element Format checkbox is selected. To define a set of custom colors see the Get and Set Project Colors topic.

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show this toolbar using the View | Toolbars menu.
3.8.9 Workspace Views

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

- Project Browser window [Alt]+[0]
- Properties window [Alt]+[1]
- Enterprise Architect UML Toolbox [Alt]+[5]
- Glossary and System window [Alt]+[2]
- Maintenance window [Alt]+[4]
- Testing window [Alt]+[3]
- Debug Workbench [Alt]+[8]
- Source Code window [Alt]+[7]
- Element Hierarchy window [Ctrl]+[Shift]+[4]
- Project Management window [Ctrl]+[Shift]+[7]
- Output window [Ctrl]+[Shift]+[8]
- Element Relationships window [Ctrl]+[Shift]+[2]
- Requirements and Constraints (Rules and Scenarios) window [Ctrl]+[Shift]+[3]
- Pan and Zoom window [Ctrl]+[Shift]+[N]

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

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the View | Toolbars menu option.
3.8.10 Other Views Toolbar

The Other Views toolbar provides access to the following project information windows in Enterprise Architect, in order:

- **Element List** - Displays the current diagram or package in a context-sensitive, editable table  
  
- **Model Search** - Opens the Enterprise Architect Model Search and its facilities

- **Relationship Matrix** - Opens the relationship matrix to cross reference elements to each other by connector type

- **Discussion Forum** - Opens the Project Forum

- **Audit View** - Displays the Audit View, which shows the information that has been recorded by auditing

- **Web Browser** - Opens the web browser page at the site you have specified on the Options dialog, in the Web Home field.

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

**Note:**

The buttons on this toolbar - and the facilities they access - are not all available in all three editions of Enterprise Architect. For example, the Discussion Forum is available only in the Enterprise Architect Corporate and Professional editions, and the Audit View is available only in the Enterprise Architect Corporate edition.

You can move this toolbar to any dockable position and it retains that position in subsequent sessions. You can hide or show the toolbar from the View | Toolbars menu option.
3.8.11 Status Bar

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

In particular the Status bar lists the currently selected element, the status of [Caps Lock], [Num Lock] and
[Scrlk] (scroll lock) and the current coordinates of the selected element.

You can hide or show this toolbar from the View | Toolbars menu option. The Status bar cannot be docked in
any other position.
3.8.12 Rich Text Notes Toolbar

Although it is not an independent toolbar that you can pin to the screen top or sides, or float in your work area, the Rich Text Notes toolbar appears in many places across Enterprise Architect in the Notes and Description fields of:

- The element Properties dialog:
  - General tab
  - Requirements tab
  - Scenario tab
  - Hyperlink Notes
- The Diagram Properties dialog
- The Connector Properties dialog
- The Message Properties dialog
- The Operations and Attributes Properties dialogs
- The Testing Window descriptions
- The Notes window
- The Rules and Scenarios Window for:
  - Requirements
  - Linked Requirements
  - Scenarios.

Notes:

- If the toolbar is displayed but grayed out, the text field is read-only and cannot be edited. Other Description or Notes fields in Enterprise Architect might not have the toolbar, in which case the Rich Text Notes facility is not available for those fields.

- For any Notes text that is displayed on a diagram, you must select the Render Formatted Notes checkbox on the Feature Visibility dialog in order to reproduce the formatting.

The options of this toolbar operate on selected text and any new text continuing from the formatting. The options (with some keyboard shortcuts) are, from left to right:

- Make text bold [Ctrl]+[B]
- Make text italic [Ctrl]+[I]
- Underline text [Ctrl]+[U]
- Change the font color of the text
- Insert list bullet points [Ctrl]+[.] (full stop)
- Insert list numbering [Ctrl]+[1]
- Make text superscript
- Make text subscript

Additional keyboard shortcuts:

- Undo changes [Ctrl]+[Z]
- Redo changes [Ctrl]+[Y] or [Ctrl]+[Shift]+[Z]
- Copy [Ctrl]+[C]
- Paste [Ctrl]+[V]
- Cut [Ctrl]+[X]

Any Note text appearing in the element Note compartments in diagrams is not formatted.
3.9 Diagram Tabs

Diagram Tabs are located at either the bottom or the top of the diagram area, above the status bar. The default location is at the bottom of the diagram area; for details of how to place the diagram tabs at the top, see the Configure Local Options | General topic. Each time you open a diagram, the diagram name is shown in the tab for easy identification and access.

Notice that the Use Case Model tab is bold; this means that the current diagram is the Use Case Model diagram.

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

The Diagram Tabs Menu

To access the Diagram Tabs menu, right-click on an appropriate tab. In the example below, the Deliver Books tab was right-clicked.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Changes to ‘&lt;tab name&gt;’</td>
<td>Save the unsaved changes made to the diagram.</td>
</tr>
<tr>
<td>Save All</td>
<td>Save the model.</td>
</tr>
</tbody>
</table>
| Reload ‘<tab name>’        | 1. Reopen the diagram without the unsaved changes; that is, revert to the state before any changes were made.  
                              | 2. Refresh the diagram from the repository, to show any changes made by other users in a shared model.       |
| Close ‘<tab name>’         | Close the diagram; Enterprise Architect prompts you to save changes to the diagram.                           |
| Close All                 | Close all open diagrams; Enterprise Architect prompts you to save any diagrams with unsaved changes.         |
| Close All Except ‘<tab name>’ | Close all diagrams except for `<tab name>`; Enterprise Architect prompts you to save any diagrams with unsaved changes. |
3.10 View Options

Models in Enterprise Architect are viewed in different ways in the application workspace, either in the Diagram View or the Element List. You can also develop tailored views of your model or project in the Model Views window.

See Also
- Relationship Matrix
- Model Search
3.10.1 Diagram View

The Diagram View is the main workspace window that enables you to create and display diagrams. You can open many diagrams, but you can view only one at a time. You open the first diagram by double-clicking on the diagram name in the Project Browser. You can then open further diagrams in the same way, or from within an open diagram by clicking on hyperlinks or elements that contain other diagrams.

Tip:
You can also use the Element List to manipulate elements.

Typical diagram activities include:

- **Add new elements** to the diagram using the Enterprise Architect UML Toolbox
- **Add existing elements to the diagram** by dragging them from the Project Browser
- **Add connectors** between elements using connectors from the Enterprise Architect UML Toolbox
- **Copy elements** in a diagram to link or copy elsewhere
- **Zoom a diagram** to different magnifications
- **Use the Diagram toolbar**, forward and back arrows to display the previous or next diagram
- **Align** and **resize** multiple elements
- **Delete** elements from the diagram (but not the project)
- **Double-click on the diagram background** to open the diagram Properties dialog and set **diagram properties**
- **Print and print preview** diagrams
- **Save the diagram image to file**
- **Save the diagram image to the clipboard**
3.10.2 Element List

The Element List is a tabular, editable view of elements that can be displayed in the main workspace. You can use the Element List to streamline the process of creating and updating elements in a package or diagram selected from the Project Browser. This can be particularly useful for analysts to create and maintain formal requirement definitions within the model. You can also print the list or generate an RTF document directly from the entries on the Element List.

To access the Element List, either:

- Select a diagram or package in the Project Browser and select the View | Element List menu option
- Select a diagram or package in the Project Browser and press [Ctrl]+[Alt]+[R]
- Right-click on a diagram or package in the Project Browser and select the Show Element List menu option
- Right-click on the background of an open diagram and select the Show as Element List context menu option
- Click on the Element List button on the Other Views toolbar.

The Element List tab displays, showing the element information for the selected package or diagram.

Note:
As you navigate the Element List, the item you have currently selected is highlighted in the Project Browser.

In the Element List you can:

- Sort the items by any column value in ascending or descending order, by clicking on the column header; initially the elements are listed in numerical order (if level numbering is turned on in the Project Browser)

  Note:
  This turns off collapsible regions; collapsible regions are shown for the State 7 and Submachine A elements in the illustration above. They are displayed only if you generate the Element List for a package, not for a diagram.

- Change the sequence of columns, by dragging column headers left or right; however, the order you place them in is not retained in subsequent sessions
- Display the Properties dialog for an item by double-clicking on the item entry
- Report on specific element types by clicking on the drop-down arrow in the Element List Toolbar and selecting the appropriate element type from the filter list, or All to list all objects; the report then lists only elements of that type.
- Change the elements available in the filter list by clicking on the Select another toolbox or technology icon to the right of the list field, and selecting the appropriate category.
- Select:
  - An element by clicking on it.
  - A specific value by clicking twice on it (not double-clicking).
  - Several individual elements by holding [Ctrl] as you click on them.
  - A range of elements by holding [Shift] as you click on the first and last in the range.
- Edit a specific value in the list by clicking three times on it (or twice and press [F2]); either the value becomes directly editable or the Properties dialog displays in which you can edit the value.
- Add new items to the Element List by pressing [Ctrl]+[N] or [Insert].
- Automatically add elements to a diagram by generating the Element List on the diagram and adding elements to the list.
- Delete elements from the list by selecting the item and pressing [Ctrl]+[D].

Tip:
The report also includes each element's documentation (notes), where this exists. See the Submachine A element in the illustration above. This is a useful way to determine which elements in a diagram require further work and which are complete. You can add or edit notes by clicking on the item and pressing [Enter].

You can do further work on the Element List using the toolbar and context menu options.

### 3.10.2.1 Element List Options

#### Toolbar Options

You can also influence what information is displayed on the Element List by clicking on the Options icon in the toolbar. This displays the Element List Options dialog, with two checkboxes:

- **Show nested packages content** - defaults to deselected, so child packages are not expanded in the list; select to show the contents of child packages in the selected package or element.
- **Show notes compartment** - defaults to selected to show the element notes; deselect the checkbox to hide the notes text.

#### Audit History

In the Corporate edition of Enterprise Architect, if Auditing is turned on and the Element List is open, you can view a history of changes to any selected element or connector, in the Audit History tab of the Output window. (If security is enabled, you must have at least Audit View permissions to display the audit history).

#### Work on Elements in the Element List

You can also use the context menu to perform operations on elements in the Element List. Right-click on the required element to display the context menu. The menu options are described below:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Display the Properties dialog for the selected element.</td>
</tr>
<tr>
<td>Add New</td>
<td>If the Filter List field in the toolbar is set to All, display the New Element dialog, through which you create an element of the required type.</td>
</tr>
<tr>
<td></td>
<td>If the Filter List field is set to a specific element type, this option adds an element of that type to the package or diagram in the Element List, the Project Browser and the Diagram View.</td>
</tr>
<tr>
<td></td>
<td>Alternatively, select the Add New icon in the Element List toolbar.</td>
</tr>
<tr>
<td>Edit Notes</td>
<td>Make the Notes area of the selected item available to create or edit the note text.</td>
</tr>
<tr>
<td></td>
<td>Alternatively, select the Edit Notes icon in the Element List toolbar.</td>
</tr>
<tr>
<td>Menu Option</td>
<td>Use to</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Show Regions (Hide Regions)</td>
<td>Show or hide the parent-child relationships between elements in a package (not in a diagram). If regions are hidden, all elements are listed with equal status. If regions are shown, the parent element has an expand/collapse box () that enables you to hide its child elements.</td>
</tr>
<tr>
<td>Usage</td>
<td>Display the diagram that uses the element or, if the element is used in multiple diagrams, display the Element Usage dialog which lists the diagrams that contain the element.</td>
</tr>
<tr>
<td>Bookmark Selected</td>
<td>Bookmark the element.</td>
</tr>
<tr>
<td>RTF Report</td>
<td>Generate an RTF report. You have two options:</td>
</tr>
<tr>
<td></td>
<td>• Generate a separate report on each selected object in the report</td>
</tr>
<tr>
<td></td>
<td>• Generate one report on all selected objects.</td>
</tr>
<tr>
<td></td>
<td>In either case, the Generate RTF Documentation dialog displays.</td>
</tr>
<tr>
<td></td>
<td>Alternatively, select the RTF Report icon in the Element List toolbar. This generates one report for all selected items.</td>
</tr>
<tr>
<td>Delete Selected</td>
<td>Delete the selected element from the Element List.</td>
</tr>
<tr>
<td></td>
<td>Alternatively, select the Delete Selected icon in the Element List toolbar.</td>
</tr>
<tr>
<td>Show as Diagram</td>
<td>Show the elements as the diagram instead of as the Element List.</td>
</tr>
<tr>
<td>Show Notes (Hide Notes)</td>
<td>Display or hide the text of any internal notes that the elements have. (Not the text of Notes elements.)</td>
</tr>
<tr>
<td>Print</td>
<td>Print out the filtered results.</td>
</tr>
<tr>
<td></td>
<td>Alternatively, select the Print icon in the Element List toolbar.</td>
</tr>
</tbody>
</table>
3.10.3 Model Views

The Model Views window enables you to encapsulate your model into the areas you are interested in. You display the window by selecting the View | Model Views menu option.

There are three types of View available:

- Model Views - stored in the model and visible to all users; you can have many of these
- My Views - stored locally on your machine and visible only to you; you can have only one of these
- Technology-defined Views - read only; each View is stored with and populated by the corresponding active MDG Technology.

When you open the Model Views window for the first time on a project, a Model Views root section and My Views root section are added for you. These can not be deleted or renamed. However you can create further Model View root nodes which you can modify and delete.

Under each root section you can then add a single level of View folders, which enable you to group Views and Favorites folders as best suit your requirements. You can also export all of the View folders containing Views from any root section as an XML file, and import a Views XML file as an additional, editable Model View root node.

A View is a folder of elements or structures that you assemble by assigning a model search to the folder. When you double-click or expand the folder, the search runs and refreshes the folder contents.

In the Model View only, you can also create Favorites folders, which give you easy access to commonly-used items in the Project Browser. To create hyperlinks in your Favorites folder to the required items in the Project Browser, drag items from the Project Browser into the Favorites folder.

**Note:**

These are single-level items; if you drag a package into the Favorites folder, you cannot expand that package there. To select specific items inside a package, expand it in the Project Browser and then drag the items into the Favorites folder.
3.10.3.1 Model Views Toolbar

The availability of the Model Views Toolbar options depends on the type of object selected. The options are, from left to right:

- Displays the appropriate Properties dialog for the item selected. Alternatively, double-click on the item.
- Locates the selected object in any diagrams in which it has been used in the model, and either displays the single diagram with the object highlighted or lists the several diagrams in which the object has been located.
- Locates and highlights the selected object in the Project Browser.
- Creates a new Model View root node, and displays the New Model View dialog in which you enter the root node name.
- Creates a new Views folder in the currently-selected root node.
- Creates a new Favorites folder in the currently-selected Views folder.
- Creates a new View in the currently-selected Views folder, and displays the Create New View dialog to define the search that populates the View.
- Refreshes the selected Model Views root node, folder, View or Favorites. For a View, this runs the Model Search defined in the View properties.
- Move the currently-selected object up or down within its type; you cannot move - for example - a package below a diagram, or a View above a Favorites folder.
- Displays a prompt to confirm deletion of the selected object and - if appropriate - its contents. You cannot delete the original Model Views and My Views root nodes.

3.10.3.2 Model Views Context Menus

The Model Views window context menus display different options, depending on which level of the Model Views hierarchy you right-click on. The options are described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Display the appropriate Properties dialog for the selected object. (Not the My Views, initial Model Views or Technology-defined root nodes.) You can edit any of the properties, if required. Changes to objects populated from the model are reflected in all other views (Properties window, diagrams, reports) of that object.</td>
</tr>
<tr>
<td>New Views Folder</td>
<td>Display a prompt for the Views folder name and creates the folder in the selected root node. (Root node only.)</td>
</tr>
<tr>
<td>Import Views From XML</td>
<td>Prompt for the XML file location and creates a new Model Views node to hold the imported Views. (Root node only.)</td>
</tr>
<tr>
<td>Export to XML (Views Only)</td>
<td>Prompt for a file path and name, and copies all Views under the selected root node to an XML file at that location. (Root node only.)</td>
</tr>
<tr>
<td>Remove Model View</td>
<td>Display a prompt to delete the user-defined Model View and, if confirmed, deletes the root node and all contents. (Not for the My Views, initial Model Views or Technology-defined root nodes.)</td>
</tr>
<tr>
<td>New View</td>
<td>Display the Create New View dialog (similar to the View Properties dialog) for you to define the search that populates the View. (View folder only.)</td>
</tr>
<tr>
<td>New Favorites Package</td>
<td>Display the New Favorites Package dialog, which prompts for the package name. (View folder only.)</td>
</tr>
<tr>
<td>Remove Folder</td>
<td>Display a prompt to delete the Views folder and, if confirmed, deletes the folder and all contents. (View folder only.)</td>
</tr>
</tbody>
</table>
### 3.10.3.3 Model Views Operations

#### Define View Search

When you either:

- First create a View or
- Select to display the View properties

the View Properties dialog displays.

![View Properties Dialog](image)

In the **Name** field, type a name for the View.

In the **Search** field, either:

- Type the name of an existing search
- Click on the drop-down arrow and select an existing search from the list, or
- Click on the [ ... ] (Browse) button to display the Manage Searches dialog, edit an existing search or define a new one, then close the dialog and type or select that search name in the **Search** field.

If required, in the **Search Term** field type a specific value to search for.

Click on the **OK** button. The View is created (or updated) in a collapsed state. When you expand the View, the search executes and populates the View.

#### Move Objects Into Favorites

Drag any required package, diagram or element from the Project Browser into the required Favorites folder.
Move Objects Between Views

Views and Favorites folders are fixed in the Views folder in which you create them, and you cannot move them. However, you can copy (by dragging) objects from any View into any Favorites folder, and move (by dragging) objects between any two Favorites folders.

Use Objects From Model Views

To make use of the elements, diagrams and packages held in any View or Favorites folder, click on the item and drag it into a diagram or a Discussion Forum posting. The item behaves in the same way as if you dragged it from the Project Browser.

Export/Import Views

You export Views to create an XML file that you can:

- Import into another model as a user-created Model View or
- Call from an MDG Technology Selection (MTS) file to provide the Technology-defined View provided by the active MDG Technology.

The export and import functions are available from the Model Views root-node context menus.

When you use the export function, it acts on the complete set of View folders in the selected My Views root node, Model Views root node, or user-generated root node. You cannot export individual Views, nor can you export Favorites folders. The function displays the Save As dialog, on which you browse for the directory location for the exported XML file, and specify the file name.

When you use the import function, it displays the Select Import Filename dialog on which you browse for the directory and XML file you want to import. The import creates a new Model View folder with the same name as the copied root node.

Set Up a Technology-Defined View

To set up the Technology-defined View for an MDG Technology, you:

1. Create a user-generated Model View in Enterprise Architect while using the technology
2. Populate it with the required View folders and Views.
3. Export the Views from that Model View as an XML file to an appropriate location
4. Create a call to the file from the technology's MTS file

Thereafter, any model for which the MDG Technology is active automatically displays those Views in a Technology-defined View.
3.10.4 Model Search

The Model Search generates a report list that you can view in the main workspace. It lists each object in the Project Browser that meets the search criteria you specify within the search terms and search type.

For more information on conducting searches see the Use the Model Search topic.

When you have generated your search results, you can print them or generate an RTF report on them.

To access the Model Search, either:
- Select the View | Model Search menu option
- Click on a package in the Project Browser and press [Ctrl]+[Alt]+[A], or
- Click on the Model Search icon in the Other Views toolbar

The Model Search tab displays.

The Model Search is also displayed when you search the whole project using the Edit | Find in Model menu option.

In the Model Search you can:
- Sort the items by any column value in ascending or descending order, by clicking on the column header
- Change the sequence of columns, by dragging column headers left or right; however, the order you place them in is not retained in subsequent sessions
- Display element or diagram properties, by double-clicking on the item
- Select:
  - An element or diagram by clicking on it
  - Several individual elements or diagrams by holding [Ctrl] as you click on them
  - A range of elements or diagrams by holding [Shift] as you click on the first and last in the range
  - All elements or diagrams in the list by pressing [Ctrl]+[A].

Enterprise Architect User Guide
Tip:
The search results include the documentation (notes) for each element or diagram, where this exists. This can be a useful way to determine which elements in a diagram, or which diagrams, require further work and which are complete.

The Options Button
The **Options** button displays the **Search Options** submenu, which enables you to display the search results as a tab of the **Output** window rather than as an Enterprise Architect View. An advantage of moving the search results to the **Output** window is that you can select items from the search results and drag them onto a diagram, which you cannot do when the results are in the Enterprise Architect View. If you select the **Dock in Output Bar** menu option, when you next display the menu this option becomes **Dock in Main View**.

The **Search Options** submenu also provides the means of performing advanced searches on your project, and displaying project metrics.

The View Header
By right-clicking on a column heading you can **Hide** or **Show** that column.
If you select **Hide**, the individual selected column is no longer shown.
If you select **Show**, the following dialog displays, and you can select to show or hide several columns at once. Select the checkbox for each column to display in the header, and deselect the checkbox of each column to hide (or click on the **Select All** or **Clear All** buttons to select all headings or deselect all headings).

Work on Objects in the Model Search
You can select elements or diagrams in the **Model Search** and perform various operations on them, as well as simply dragging the item into a Discussion Forum post.
Right-click on the required object to display the following context menu:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Selected to Clipboard</td>
<td>Copy the selected item to the MS Windows clipboard so that it can be pasted to a document, spreadsheet or email.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Generate an RTF report. You have two options:</td>
</tr>
<tr>
<td></td>
<td>- Generate a separate report on each selected object in the Model Search.</td>
</tr>
<tr>
<td></td>
<td>- Generate one report on all selected objects.</td>
</tr>
<tr>
<td></td>
<td>In either case, the Generate RTF Documentation dialog displays.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you generate the report using a custom SQL search, the SQL must include <code>ea_guid AS CLASSGUID</code> and the object type.</td>
</tr>
<tr>
<td>Print Results</td>
<td>Print out the filtered results.</td>
</tr>
<tr>
<td>Clear Results</td>
<td>Clear the search results from the Model Search.</td>
</tr>
<tr>
<td>Usage</td>
<td>Display the diagram that uses the element or, if the element is used in multiple diagrams, display a list of diagrams to choose from.</td>
</tr>
<tr>
<td>Bookmark Selected</td>
<td>Bookmark the element.</td>
</tr>
<tr>
<td>Delete Selected</td>
<td>Delete the selected element from the Model Search.</td>
</tr>
<tr>
<td>Close</td>
<td>Close the Model Search.</td>
</tr>
<tr>
<td>Help</td>
<td>Display this Help topic on the Model Search.</td>
</tr>
</tbody>
</table>
3.10.4.1 Use the Model Search

You perform searches within your project in the Model Search. You search the whole model, unless you have selected the Current Tree Selection option in the Manage Searches facility. In that case, you can search within a specific package selected from the Project Browser.

In the Search Term field type the text to search for, and in the Search field select the type of search to perform (the default being Simple). Click on the Run button to display your results.

If you click on an object in the Model Search, Enterprise Architect locates and highlights that object in the Project Browser. This enables you to search for items of interest and access them quickly and directly.

You can perform more complex searches and create your own search definitions. To begin these tasks:

1. Click on the Options button. The Search Options submenu displays.
2. Click on the Manage Searches option. The Manage Searches dialog displays.

External Access to Model Search

You can access the Model Search facilities and perform specific searches indirectly, from Add-Ins, MDG Technologies, from a hyperlink and from a shortcut to access your model. This entails setting up a search profile either in the appropriate tool, or as an XML file accessed by the tool.

For information on performing a search from: See

An MDG Technology Selection (MTS) File (using an exported search definition) Working With MTS Files
A Login Shortcut Shortcuts To .EAP Files
An Add-In Add-In Search
A Hyperlink Hyperlinks

See Also

- Search a Model
3.10.4.2 Search a Model

In Enterprise Architect, you can search elements or diagrams over an entire project for a particular phrase or words. Select the Edit | Find in Model menu option. The Search Model dialog displays, which enables you to enter a search term and select the search parameters from a defined search filter, the default being a Simple search. The search filter can be one of the default filters or one that you define. For more details on defining a search see Search Definitions. Search results are displayed in the Model Search view.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Term</td>
<td>Specify the search term.</td>
</tr>
<tr>
<td>Search</td>
<td>Select a filter. If required, define your own custom search filters in Enterprise Architect.</td>
</tr>
<tr>
<td>Run</td>
<td>Run the search.</td>
</tr>
<tr>
<td>Manage Searches</td>
<td>Perform complex searches</td>
</tr>
<tr>
<td>Close</td>
<td>Close the Search Model dialog.</td>
</tr>
</tbody>
</table>

See Also
- Use the Model Search
- Create Search Definitions

3.10.4.3 Search Definitions

You provide search filters and create new search definitions using the advanced Manage Searches dialog, which you display in one of two ways:

- On the Model Search tab, click on the Options button. The Search Options submenu displays. Click on the Manage Searches option.
- Either press [Ctrl]+[F] or select the Edit | Find in Model menu option. The Search Model dialog displays. Click on the Manage Searches button.
Search filters enable you to perform customized searches on a **Search Term** in order to locate model elements. The **Search List** drop-down list provides several pre-defined search definitions.

The default is a **Simple** search, which searches all elements, looking at the **Name** and **Notes** fields only. If the search term is found in the **Name** field or the **Notes** field, those elements are displayed.

**Important:**

The fields listed in a search have an OR relationship when no **Required** checkboxes are ticked; i.e. If the search term is found in any one of those fields, then the element is displayed.

In the Simple search below, the **Name** and **Notes** fields both have the **Required** checkbox ticked, so the two fields have an **AND** relationship. The search displays only those elements that contain the search term in both the **Name** and **Notes** fields.
Note:

Any field having the Required checkbox ticked overrides fields where the Required checkbox is not ticked.

The following search finds elements that must have the search term in the Name field and that might or might not have the search term in the Notes field.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Term</td>
<td>Specify the term to search for.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Search List</td>
<td>Select a previously-defined search definition.</td>
</tr>
<tr>
<td>Run Search</td>
<td>Run the selected search. The results are displayed in the Model Search.</td>
</tr>
<tr>
<td>New Search</td>
<td>Create a new search definition, with new search criteria. See Create Search Definitions.</td>
</tr>
<tr>
<td>Save Search</td>
<td>Save a modified or new search.</td>
</tr>
<tr>
<td>Copy Search</td>
<td>Copy an existing search, to modify.</td>
</tr>
<tr>
<td>Delete Search</td>
<td>Delete the search definition from the Search List.</td>
</tr>
<tr>
<td>Get Default</td>
<td>Restore the default Search definition of the current type, after you have edited the search parameters.</td>
</tr>
<tr>
<td>Export</td>
<td>Display a selection box that enables you to select searches to export to an external directory as an XML Search file.</td>
</tr>
<tr>
<td>Import</td>
<td>Display the Windows Directory Explorer Open dialog to enable you to import searches as XML Search files from an external directory.</td>
</tr>
<tr>
<td>Search On panel</td>
<td>Display the element search filters that are contained in the 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. You edit the filters by double-clicking on the panel contents, or by clicking on the Edit Filter button, to display the Edit Filters dialog.</td>
</tr>
<tr>
<td>Search In</td>
<td>Select the type and name of each element to search on.</td>
</tr>
<tr>
<td>Condition</td>
<td>Select the condition of the search parameter. The available options are Contains, Equal To, Not Equals and One Of.</td>
</tr>
<tr>
<td>Look for</td>
<td>Specify the search term to perform the conditional search on. This value can pertain to the selected element type. For example, the value could be a date for DateCreated or a text value for other element types. The search term can contain multiple values, separated by commas.</td>
</tr>
<tr>
<td>Required</td>
<td>Indicate that the search results must include elements with your search term in that field.</td>
</tr>
<tr>
<td>Element Features</td>
<td>Specify whether element features are optional or required. These appear as a new branch underneath the root element term in the Search On panel. The Extended search is a good example; select this definition in the Search List field. If you scroll down the Search In column, you see sub branches such as Attribute, Change, and Custom Property. These are the element features. You can add these features by clicking on the Add Filter button. The Add Filters dialog displays, with a list of all the filters you can choose for an element or element feature. Click on the Search On Element drop-down arrow to see a list of the element features you can search on. Each feature has its own set of filters such as Name, Notes and Alias, which you can add to your search. To search on an element Attribute name, you would add the Attribute feature with a Name filter to your search. The Optional radio button enables you to generate a list of elements that meet one of the element filters (Element Type = Object), or one of the feature filters (Attribute Name = Class1). For example, if your search is Element Name = Class11, Attribute Name = m_Att1 or Scope = Public and you selected Optional, the search results would list all the elements that have the name of Class11 and all the elements that have an Attribute Name of m_Att1 or a Scope of Public. The Required radio button enables you to generate a list of elements that must have the element features you have added. For example, if your search is Element Name = Class, Attribute Name = m_Att1 or Scope = Public you would get elements that must have the name of Class AND an Attribute with a name of m_att1 or a Scope of Public.</td>
</tr>
<tr>
<td>Add Filter</td>
<td>Add a new set of parameters to filter the search on.</td>
</tr>
<tr>
<td>Edit Filter</td>
<td>Open the Edit Filters dialog, which enables you to change the search parameters.</td>
</tr>
</tbody>
</table>
### Search In

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove Filter</td>
<td>Remove the selected filter from the search.</td>
</tr>
<tr>
<td>Search In</td>
<td>Select either:</td>
</tr>
<tr>
<td></td>
<td>- Entire Model - the default, which searches the entire model, or</td>
</tr>
<tr>
<td></td>
<td>- Current Tree Selection - which runs a search in a specific package, which you select from the Project Browser.</td>
</tr>
</tbody>
</table>

**Note:**

If you select the Current Tree Selection option, navigating the Project Browser does not change your search results until you click on the Run button. That is, to search different areas of the project, click on the first required package in the Project Browser and click on the Run button, check the results, and then click on another package in the Project Browser and click on the Run button again.

#### 3.10.4.3.1 Create Search Definitions

Search definitions are created using the Manage Searches dialog. To access this dialog:

- Click on the Options button in the Model Search and then on the Manage Searches menu option, or
- Select the Edit | Find in Model menu option, then click on the Manage Searches button.

To create a new search definition, follow the steps below:

1. Click on the New Search button. The Create New Search Query dialog displays.

   ![Create New Search Query dialog](image)

   - **Search Name:** Type a name for your new search.
   - **Add-in Name and method:**
   - **Editor Type:** Select the type of search you require:
     - The Query Builder option provides an interface that enables you to design your own search.
     - The SQL Editor option enables advanced users to directly write SQL Select statements.
     - The Add-In Search option enables you to supply the name of your Add-In and a method (e.g. MyAddin.RunThisMethod). This method is called whenever the search is run. This search can be exported and distributed as a part of your Add-In. For more information, see Add-Ins.
   - **OK**

2. In the Search Name field, type a name for your new search.
3. Select the radio button for the type of search you require:
   - The Query Builder option provides an interface that enables you to design your own search.
   - The SQL Editor option enables advanced users to directly write SQL Select statements.
   - The Add-In Search option enables you to supply the name of your Add-In and a method (e.g. MyAddin.RunThisMethod). This method is called whenever the search is run. This search can be exported and distributed as a part of your Add-In. For more information, see Add-Ins.
4. Click on the OK button.

#### Query Builder

Your search definition now appears as being selected in the Search List drop-down. The main window displays the message 'There are no items to show in this view'. You can now click on the Add Filter button to Add Filters.
SQL Editor

The Custom SQL dialog displays, enabling you to input your SELECT statement.

For all Enterprise Architect functions in which you use a custom SQL statement (such as RTF reporting or Model Views) the statement should return the guid and type of the object found so that Enterprise Architect can search for the selected item in the Project Browser. For example:

```
SELECT ea_guid AS CLASSGUID, Object_Type AS CLASSTYPE, Name FROM t_object
```
When you have defined the SELECT statement, click on the Save button to save this search. The search is then available from the Search List.

You can display an item's properties by simply double-clicking on a table name in the Table Name list, and executing the SELECT statement that displays for that table in the Query panel. For example:

SELECT * FROM t_object WHERE NAME=<Search Term>'

You can extend the usability of your SQL searches using the aliases CLASSGUID and CLASSSTYPE. These enable Enterprise Architect to display the Properties dialog and icon for elements, connectors, attributes or operations, as well as selecting them in the Project Browser. Some simple examples for using these aliased fields are provided below:

SELECT ea_guid AS CLASSGUID, Object_Type AS CLASSSTYPE, Name FROM t_object
SELECT ea_guid AS CLASSGUID, Connector_Type AS CLASSSTYPE, Name FROM t_connector
SELECT ea_guid AS CLASSGUID, 'Operation' AS CLASSSTYPE, Name FROM t_operation
SELECT ea_guid AS CLASSGUID, 'Attribute' AS CLASSSTYPE, Name FROM t_attribute.

Add-In Search

Type in the field the name of your Add-In, a period (full stop) and then the name of the method to be called (e.g. MyAddin.RunThisMethod). Your search is automatically saved and available from the Search List.

3.10.4.3.2 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 and Test Cases.
- **Attribute Details** - Searches for elements with attributes relating to the search term, including Tagged Values, constraints, and common attribute data fields.
- **Find Orphans** - Searches for orphaned elements throughout the model, with the 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** - Searches for elements containing internal test cases where the search term is in any common Test Case field and the Status value is 'Fail'.
- **Method Details** - Searches for elements with operations and methods relating to the search term, including Tagged Values, constraints and common operation and method data fields.
- **Responsibility** - Searches for elements with internal responsibilities/requirements where the search term relates to any common Responsibility/Requirement field.
- **Resources** - Searches for elements with assigned resources where the search term relates to any common Resource field.
- **Requirements** - Searches for Requirement element types where the search term relates to any common element field.
3.10.4.3 Add Filters

Click on the Add Filter button in the Manage Searches dialog. The Add Filters dialog displays.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Search On</strong></td>
<td>Select items to build up search filters on any information about an object. The following is a list of what is available, before you have defined a search.</td>
</tr>
<tr>
<td>Include</td>
<td>Select each field item to include in your search. (Select the checkbox.)</td>
</tr>
<tr>
<td>Field</td>
<td>Identify the name of the field to search. See Fields and Conditions. The list presents items specific to the filter Search On item.</td>
</tr>
<tr>
<td>Condition</td>
<td>Specify the condition of the search parameter. See Fields and Conditions.</td>
</tr>
<tr>
<td>Value</td>
<td>Type a value pertaining to the selected element type. For example, the value could</td>
</tr>
</tbody>
</table>
### UML Modeling Tool Features | View Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>Select a particular field to generate a result set that <strong>must</strong> contain your search term in that field.</td>
</tr>
<tr>
<td>Check All</td>
<td>Select all the items to include them in the search definition.</td>
</tr>
<tr>
<td>Uncheck All</td>
<td>Deselect all the items to omit them from the search definition.</td>
</tr>
<tr>
<td>OK</td>
<td>Apply the filter. The fields selected are added to the search definition.</td>
</tr>
</tbody>
</table>

You can add multiple search definitions as necessary. Note that if you select the **Required** field in multiple definitions the search rapidly becomes impractical. Multiple search definitions are better for ‘and/or’ searches.

### 3.10.4.3.3.1 Fields and Conditions

When you click on a condition for a particular field, a selection of conditions becomes available, as shown in the following example:

<table>
<thead>
<tr>
<th>Contains</th>
<th>Equal To</th>
<th>Not Equals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal To</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Equals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One Of</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For some conditions, the value field contains an ellipsis (⋯). Click on this to display a selection dialog. Examples of selection dialogs are shown below.

#### Example Selection dialog for One Of section

For the **Types** field, the dialog shows:

- **Protected**
- **Friend**
- **Internal**
- **Published**
- **Private**
- **Protected**
- **Public**

#### Date Selection dialog for Before or After section

- **1/06/2007**

#### Date selection from the drop-down

- **June 2007**

**See Also**
- [Create Search Definitions](#)
- [Add Filters](#)
3.11 The Web Browser

The Web Browser displays as a tab of the central work area, like the Start Page, Model Search, Element List and Diagram View. It provides access within Enterprise Architect to internet facilities such as email, websites and search engines.

To access the Web Browser:

- Press [Ctrl]+[Alt]+[W]
- Click on the Web Browser icon ( ) in the Other Views toolbar, or
- Select the View | More Windows | Web Browser menu option.

The Web Browser opens at the default home web site; you define the default home website, search engine and email exchange address on the General page of the Options dialog.

To access the:

- Email exchange server, click on the 'envelope' icon in the toolbar; the email login window displays
- Web search engine (such as Google), click on the 'spyglass' icon in the toolbar; the search engine screen displays
- Home web site, after displaying other web pages, click on the 'house' icon in the toolbar.

To go directly to another website or email server (your internet security permitting), in the Address field type or select the website http address and click on the Go button.
3.12 Arrange Windows and Menus

Enterprise Architect enables you to rearrange the windows and some menus to suit your work habits. For example, you can:

- Dock Windows
- Autohide Windows
- Tear Off Menus
3.12.1 Dock Windows

A number of Enterprise Architect windows can be freely positioned on the screen, or docked against any edge of the application workspace. These windows are collectively called dockable windows. Drag the window around the application workspace until you find a comfortable way of working. The examples below describe a few ways you can rearrange the windows to suit your work habits.

Floating Windows
To float a window anywhere on the screen, just drag the window by its title bar to the required position.

Dock a Window Against an Edge

The navigation compass enables you to dock windows against an edge of the application workspace. You drag the window over one of the points of the compass to dock it into a tabbed location. The window does not overlap any other window, so if you are docking several windows you could cover the workspace; however, you can avoid this by combining them in a single tabbed frame.

To dock a window against an edge, follow the steps below:

1. Click on the item to move and start dragging it towards the required position. This activates the navigation compass.

2. Drag the window onto a compass point. The screen display shades the area where the window is to be placed.

3. Release the mouse button over the compass point to confirm the position; this docks the window.

In the example below, when the mouse button is released the Project Browser is docked into the shaded area.
Dock Required Windows into One Frame

You can also dock all of the windows you are using into a single frame, by either:

- Dragging the title bar of each window up to the title bar of the first docked window, or
- Dragging each window over the 'tabbed frame' icon in the middle of the compass, when another window is already docked.
You can do this with all dockable windows. The following example shows the Testing, Project Browser, Resources, Properties, Tagged Values and Notes windows all in one frame.

To separate a window from a combined frame, click on the window tab at the bottom of the frame and drag it away.
3.12.2 Autohide 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 menus within a frame, click on the button.

Use Automatically Hidden Windows

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

Hover the cursor over a tab to access the associated window.

Tip:

You can also use the View | Visual Style | Animate Autohide Windows menu option to animate windows that have been automatically hidden.
3.12.3 Tear Off Menus

Some sub-menus in the Enterprise Architect main menu are tear off menus. This is indicated by the bar at the top. For example, the Element | Make Same sub-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 it away. The menu detaches 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.
There are several dockable tab windows available to use in Enterprise Architect. These can be accessed either:

- Through the View menu or the View | More Windows sub menu, or
- Through the context menu accessed by right-clicking on the main menu.

The dockable windows available include:

- Project Browser
- Properties
- Enterprise Architect UML Toolbox
- Resources
- Notes
- System
- Testing
- Maintenance
- Source Code Viewer
- Element Browser
- Relationships
- Rules
- Hierarchy
- Tagged Values
- Project Management
- Output
- Tasks Pane
- Pan & Zoom

**Tip:**
If the text in a window panel is too small to read comfortably, click on it, press and hold [Ctrl] and use the mouse wheel to expand and reduce the text size.
3.13.1 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 window shows the element's name, stereotype, version, author, dates and other pertinent information.

Tip:

The Properties window 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 window is divided into three expandable sections:

- `<Element type>` Settings - for the basic element details
- Project - for general housekeeping settings
- Advanced - only active for generalizable elements.

Note:

When you click on a field name, a brief explanation of that field displays at the bottom of the Properties window, unless you have selected the Hide Properties Info Section checkbox on the General page of the Options dialog. If you click on the field value for an editable field, a drop-down arrow displays that enables you to select a different value. Both of these features are shown in the example below:
3.13.2 The Resources Window

The Resources window displays a tree of Model Elements, Scripts, Documents, UML Profiles and Patterns, and Matrix profiles. This view provides useful shortcuts and re-use functions that you can use to add stock elements to the current model, patterns and elements for additional information.

Tip:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Configure Resources permission to maintain Resources window items.

- **MDG Technologies** provide a convenient way to insert complex new elements and features without having to retype or reconfigure each element.
- **Templates** provides templates you can design yourself to apply to your documentation in either RTF style or Web style to customize the RTF and HTML that make up Enterprise Architect reports.
- **Documents** provides a shortcut to the RTF documentation functions.

Tip:
To add a document to the shortcut list, select the Project | Documentation | Rich Text Format (RTF) Report menu option. Once you have defined your document click on the Resource Document button and type in a name. The document name then displays in the Resources window. By right-clicking on the document name you can regenerate documents individually or as a batch, or open them directly from Enterprise Architect.

- **Matrix Profiles** provide quick access to saved Relationship Matrix profiles; double-click on a profile to load the matrix with the saved settings and source-target packages.
- **Favorites** provides a shortcut to elements that you configure as a shortcut.
• **Stylesheets** enables you to import XSL Style sheets, which are then available in the drop-down list on the XML Export dialog.

  **Note:**

  If you select a style sheet on export, Enterprise Architect applies that style sheet to the XMI generated before saving to file. This makes it convenient to generate other forms of output from the base XMI content. Combined with UML Profiles, this is a powerful means of extending Enterprise Architect to generate almost any content required.

### 3.13.2.1 Favorites

The **Resources** window contains a **Favorites** folder. Here you can hyperlink 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 - such as the list of Actors in a system - 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.

#### Modify 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.
- From the context menu select the **Find | Add to Favorites** option.
- Switch back to the **Resources** window 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 **Resources** window.
- Select **Delete** from the context menu.
- Confirm the action by clicking on the **Yes** button.

**View Properties of a Favorite**

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

- Select and right-click on the favorite in the **Resources** window.
- Select **Element Properties** from the context menu.
3.13.3 The Notes Window

You use the Notes window to view and edit the element documentation (notes) associated with elements, diagrams, attributes, operations and connectors, either from a diagram (for both elements and connectors) or from the Project Browser (elements only). When you select an element, the note displayed changes to reflect the current selection. If you make changes to notes in this tab, they are saved.

Notes are the main documentation feature you use to describe an element or connector. In the documentation that Enterprise Architect generates, notes feature prominently.

You can format the notes text using the Rich Text Notes toolbar at the top of the Notes window.

Tip:

You can also edit notes by double-clicking on an element or connector in a diagram or in the Project Browser, to open the Properties dialog. Any formatting changes made in one display are reflected in the other.
3.13.4 The System Window

The System window documents tasks and issues that relate directly to the current project. It has three tabs:

- **Project Tasks** - a list of major project tasks that require attention; you can filter tasks based on their current status - right-click for a popup menu, or double-click on a line item to modify details.
- **Project Issues** - a list of events, occurrences and situations that impact on project development and delivery; you can review issues using the right-click menu or by double-clicking on selected issues.
- **Project Glossary** - a list of all the 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.

**Note:**

Right-clicking in the System window displays 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 clicking in the title bar of the column that the items are to be indexed on.
3.13.5 The Source Code Viewer

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

A number of options change the way the Source Code viewer works. They can be altered via the Options dialog (select the Tools | Options | Source Code Engineering | Code Editors menu option).

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.

The Source Code viewer also displays any DDL generated for a selected table in your diagram.

File Parsing

The Source Code viewer parses files for a number of reasons. The first is to enable it to jump to the location in the file at which the currently selected item is found.

Additionally, parsing displays a structure tree showing an overview of the file in a similar fashion to the main Project Browser. You can also select anything in that and jump to the appropriate line in the editor.

The viewer cannot parse DDL, and therefore does not show the structure tree for a DDL file.

The Source Code Viewer Toolbar Buttons

The toolbar buttons in the Source Code viewer enable you to edit, view and interact with the code contained in the Source Code viewer. The function of each button is described below:

- Open button - opens the source code from an existing file, this option is also active in the 'Lite', read-only version of Enterprise Architect
- Save button - saves the changes to the currently loaded source code; this option is also active in the 'Lite', read-only version of Enterprise Architect
- Copy button - copies the highlighted text
- Paste button - pastes the text that is currently contained in the buffer to the source code viewer
- Undo button - cancels the previous action
- Generate and Reload button - generates and reloads the current object source
- Save and Resynch button - saves the source code and resynchronizes the Class
- **Code Templates** button - accesses the Code Templates Editor
- **Build and Run** button - provides quick access to the following commands:
  - **Build** - run package build scripts
  - **Test** - run package test scripts
  - **Run** - run debug
  - **Package Build Scripts** - configure package build scripts
    (these options are also active in the 'Lite', read-only version of Enterprise Architect)
- **Set Font** button - sets the font for the text contained in the Source Code viewer.
3.13.6 The Element Browser

The Element Browser window displays all aspects of the selected element as shown below.

Select the View | More Windows | Element Browser menu option, or right-click on the Main Menu bar to display the context menu and select the Element Browser menu option.

The following are displayed, where available:

- Operations
- Attributes
- Tags
- Constraints
- Requirements
- Files
- Connectors (Links)
- Scenarios
- Documents.
3.13.7 The Relationships Window

The Relationships window displays all connectors between the currently selected element and other elements. This provides a quick overview of an element’s relationships in the model.

For each connector, the connector type and target element are displayed. If a ‘Yes’ appears in the Target in Diagram column, the target element is visible in the currently loaded diagram. This is useful when you are dragging related elements from the relationships list onto the current diagram.

Double-click on a connector in the list to open the <connector type> Properties dialog, where you can edit the connector attributes. Right-click on a connector to open the context menu.

You can locate the related element, view the related element properties or delete the connector. You can also hide certain connectors from appearing in diagrams, and show hidden connectors (first example of the menu, below).

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

Tip:

In the Corporate edition of Enterprise Architect, with security on, the diagram and the source and target elements must be free for editing before some of these options are available for use.
3.13.8 The Rules & Scenarios Window

The Rules & Scenarios window displays all requirements, constraints, linked requirements and scenarios against an element. The Rules & Scenarios 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 either press [Ctrl]+[Shift]+[3] or select the View | More Windows | Rules & Scenarios menu option; the Rules & Scenarios window displays.
2. Click on the required folder; for example, Requirements. You can:
   - Add a new rule by pressing [Ctrl]+[N]
   - Save by pressing [Ctrl]+[S]
   - Delete by pressing [Ctrl]+[D]
   - Add formatted notes in the Notes field, using the Rich Text Notes toolbar at the top of the field.
### 3.13.9 The Hierarchy Window

The Hierarchy window shows a mini picture of the composition of the current element 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; embedded elements are also shown. This helps extend the picture of where an element exists in the model space.

Display of each type of relationship is optional, and can be toggled using the toolbar for the hierarchy window. Roll the cursor over each toolbar icon to display the types of relationship that the icon filters. For example, T filters the hierarchy to display Transformation Dependencies, and R filters for Custom References.

**Tip:**
You can alter the maximum number and the initial number of levels that the hierarchy opens to by selecting the Tools | Options menu option and, on the General tab, updating the Max Hierarchy View Depth and Open Hierarchy View to fields.
### 3.13.10 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. Tagged Values are by default set to hide duplicate and fully qualified values, as in the first window below. If you prefer, you can change the settings to show duplicate values and fully-qualified values (which show exactly where the Tagged value came from) as in the second window below:

<table>
<thead>
<tr>
<th>Tagged Values</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Activity</strong></td>
<td>Task</td>
</tr>
<tr>
<td>ActivityType</td>
<td>Task</td>
</tr>
<tr>
<td>AdHocCompletionCondition</td>
<td>false</td>
</tr>
<tr>
<td>AdHocOrdering</td>
<td>Parallel</td>
</tr>
<tr>
<td>Assignments</td>
<td></td>
</tr>
<tr>
<td>Categories</td>
<td></td>
</tr>
<tr>
<td>ComplexML_FlowCondition</td>
<td></td>
</tr>
<tr>
<td>DiagramRef</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>Unspecified</td>
</tr>
<tr>
<td>InMessage</td>
<td></td>
</tr>
<tr>
<td>InputPropertyMaps</td>
<td></td>
</tr>
<tr>
<td>Instantiate</td>
<td>false</td>
</tr>
</tbody>
</table>

**AdHocOrdering**
Values: Sequential, Parallel
Default: Parallel

<table>
<thead>
<tr>
<th>Tagged Values</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Activity</strong></td>
<td>Task</td>
</tr>
<tr>
<td>BPMN::Activity::ActivityType</td>
<td>Task</td>
</tr>
<tr>
<td>BPMN::Activity::AdHoc</td>
<td>false</td>
</tr>
<tr>
<td>BPMN::Activity::AdHocCompletionCondition</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::AdHocOrdering</td>
<td>Parallel</td>
</tr>
<tr>
<td>BPMN::Activity::Assignments</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::Categories</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::ComplexML_FlowCondition</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::DiagramRef</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::Id</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::Implementation</td>
<td>Unspecified</td>
</tr>
<tr>
<td>BPMN::Activity::InMessage</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::InputPropertyMaps</td>
<td></td>
</tr>
<tr>
<td>BPMN::Activity::Instantiate</td>
<td>false</td>
</tr>
</tbody>
</table>

**BPMN::Activity::AdHocOrdering**
Values: Sequential, Parallel
Default: Parallel
Note:

Fully qualified Tagged Values can be displayed only if the Tagged Value was created in Enterprise Architect release 7.1 or later. You cannot display the fully qualified path for Tagged Values from earlier releases.

A Technology Developer can also create new structured Tagged Values, predefined reference data Tagged Value types and custom Tagged Value types, as described in SDK for Enterprise Architect.

The Tagged Values window is a dockable window. You can use it to perform the following actions:

- Assign a Tagged Value to an Item
- Modify Tagged Values
- Assign Notes to a Tagged Value

Model Elements and Features with Tagged Values

The following model components can use the Tagged Values window as a convenient way to quickly view and modify Tagged Values:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>Elements display their own Tagged Values along with any inherited values.</td>
</tr>
<tr>
<td>Object Instances</td>
<td>Object Instances display owned tags and those obtained from their classifier.</td>
</tr>
<tr>
<td>Ports and Parts</td>
<td>Ports and parts display information similar to objects and display Port/Part 'Type' instead of a classifier. Tags are included for all parents and other structures of the Ports type.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Include owned Tagged Values and those received from attribute type classifiers, with the inclusion of any inherited ones.</td>
</tr>
<tr>
<td>Operations</td>
<td>Owned properties only.</td>
</tr>
<tr>
<td>Connectors</td>
<td>Owned properties only.</td>
</tr>
</tbody>
</table>

When over-riding an inherited property, Enterprise Architect 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 use the Tagged Values toolbar, as described below.

Tagged Values Toolbar Buttons

The buttons in the Tagged Values toolbar enable you to add, edit, sort, delete and arrange the Tagged Values of model features.

From left to right, the button functions are as follows:

- The Compartments button displays the Tagged Values in element compartments on diagrams
- The Sort Alphabetically button sorts the current Tagged Values for the element alphabetically
- The New Tag button adds a new Tagged Value
- The Edit Notes button enables you to create notes that explain the purpose of the Tagged Value
- The Delete selected button removes the currently selected Tagged Value
- The Default Tagged Value types button enables quick access to tag definitions created in the Configuration menu
- The Options button enables you to show or hide the fully qualified paths for the Tagged Values in the window
- The Help button displays help relating to use of the Tagged Values window.

3.13.10.1 Assign a Tagged Value to an Item

You can assign Tagged Values to several model features, as listed in the Model Elements and Features with Tagged Values topic.

To add a Tagged Value follow the steps below:
1. Create user-defined Tagged Values either using a predefined Tagged Value type or by creating a Custom Tagged Value type (as described in SDK for Enterprise Architect).
2. Select the model feature to associate with the defined Tagged Value.
3. Ensure that the Tagged Values window is visible (select the View | Tagged Values menu option, or press [Ctrl]+[Shift]+[6]).
4. Either click on the New Tags button or press [Ctrl]+[N]. The Tagged Values dialog displays.
5. On the Tag field, click on the drop-down arrow and select the appropriate defined Tagged Value to assign to the item.

**Note:**

Direct entry of predefined Tagged Values is only available for predefined tags of type string.

6. To confirm selection of the Tagged Value, click on the OK button.

**Modify Tagged Values with the Tagged Values Window**

Once a Tagged Value has been assigned to the model feature it is possible to edit the values from the Tagged Values window. Model features that you can apply Tagged Values to are detailed in the Model Elements and features with Tagged Values topic. To edit the Tagged Values follow the steps below:

1. Click on the View | Tagged Values menu option, or press [Ctrl]+[Shift]+[6]. The Tagged Values window displays.
2. Click on the model feature for which to edit the Tagged Values. The window shows all of the Tagged Values for the selected feature.
3. Edit the fields as appropriate. The information entered can 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.

There are three types of value field for a Tagged Value:

- *'Open'* fields, in which you can type any appropriate value
- *'Drop-down list'* fields, where you click on the drop-down arrow to select from a discrete list of possible values such as M or F, or Win, Lose or Draw
- *'Further detail'* fields, where you click on an ellipsis ([...]) in the field to display a dialog in which you enter information (such as notes) or indicate a source of further information (such as a classifier).

The example below shows a value being modified, using a drop-down list.
3.13.10.2 Assign Notes 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 that you can apply Tagged Values to are detailed in the Model Elements and Features with Tagged Values topic). To facilitate this from the Tagged Values window, follow the steps below:

1. Click on the View | Tagged Values menu option, or press [Ctrl]+[Shift]+[6]. The Tagged Values window displays.
2. Click on the model feature for which to edit the Tagged Values. The Tagged Values for the selected model feature display.
3. Click on the Tagged Value to add information to.
4. Click on the Edit Notes button or press [Ctrl]+[E]. The Tagged Value Note dialog displays.

Note:
To override a Tagged Value defined in a parent element, edit the value in the from <parentname> compartment of the Tagged Values window. Once this has been done the tag is moved into the selected element's Tagged Values; this does not affect the Tagged Values defined in the parent element.
5. In the **Note** field, type the information relating to the Tagged Value. This information is displayed in the lower portion of the **Tagged Values** dockable window whenever the Tagged Value is selected.

### 3.13.10.3 Show Duplicate Tags

Tagged Values are by default set to hide duplicate values. This setting is used to facilitate inherited and overridden tag names.

To show duplicate Tagged Values follow the steps below:

1. Select the **Tools** | **Options** menu option. The **Options** dialog displays.
2. From the hierarchical tree, select the **Objects** item.
3. Select the **Show Duplicate Tags** checkbox.
3.13.11 The Project Management Window

The Project Management window enables you to input the resources, effort, risks and metrics that can be added to elements contained in the model.

Open the Project Management window by selecting the Element | Resourcing Metrics & Risk menu option, or the View | Project Management menu option (or press [Ctrl]+[Shift]+[7]).

Right-click on the list to view the context menu, which enables you to add and delete list items.

Toolbar

These buttons have the following functions (in order as shown on the toolbar):

- **New**: Create new item
- **Save**: Save changes to an item
- **Copy**: Enables you to duplicate an existing entry. You must change an item's Role for this to become enabled
- **Delete**: Delete an item from the list
- **Sort**: Sort Items in the list
- **Print**: Print item data from the list
- **Show/Hide Details**: Swap between detailed and summary new window styles
- **Help**: Show help contents for this window.
3.13.12 The Output Window

The Output window is used to display data that is either system generated or Add-In generated. Examples of situations where Enterprise Architect generates items include:

- Validation Items
- Launch of external processes
- Command line output from Build and Test
- Parse errors generated during import
- (Corporate edition of Enterprise Architect) In the Audit History tab, a history of changes to any element or connector selected from the Audit View, the Element List, the Project Browser or the current diagram (Auditing must be turned on and the Element List open)
- Re-docking the Model Search results into the Output window.

You can drag suitable items out of the Output window and add them to diagrams.

Double-click on model validation errors or parsing errors to display the source of the error.

You can also right-click on an item and select context menu options to:

- Copy the selected item to the clipboard
- Copy all items to the clipboard
- Save the output to an external file
- Clear the output from the window.

The Output window can also be used by Add-Ins, if they are configured to do so via the Automation Interface. See SDK for Enterprise Architect.
3.13.13 The Tasks Pane Window

The Tasks Pane window provides access to a range of context-specific help topics, online resources and Enterprise Architect facilities to give you quick access to information and facilities in areas of interest in Enterprise Architect. When you first open Enterprise Architect, the Tasks Pane automatically displays on the right of the screen.

The Tasks Pane has several topic areas such as:

- Getting Started
- Managing Requirements
- Debug and Profile
- Code Engineering.

The list of topic areas varies, and can include topics specific to any MDG Technologies being used with Enterprise Architect.

To switch between the topic areas, either:

- Click on the More Tasks option in the toolbar and select the required area from the list, or
- Click on the left or right arrow buttons in the toolbar.

The ‘Home’ icon returns you to the Getting Started topic area.

Tasks Pane Contents

The Tasks Pane provides several types of information and resources. Click on a:

- icon to open appropriate topics from the Enterprise Architect Help file
- icon to open web pages or documents on the Sparx Systems web site
- icon to begin Enterprise Architect tasks appropriate to the Tasks Pane topic area; you must be in an appropriate functional area of Enterprise Architect in order for these tasks to function, such as in an open diagram
- icon to begin Add-In tasks appropriate to the Tasks Pane topic area; you must be in an appropriate functional area in order for these tasks to function
- icon to open report facilities to provide information or data collation tools
- icon to start demonstrations of Enterprise Architect functions in action.

The selected information, web page or demonstration displays on a Browser tab in the main view, or the appropriate task or report window opens.

If you close the Tasks Pane window, to access it again:
From any Enterprise Architect screen, press [Ctrl]+[Shift]+[9]
Select the View | Tasks Pane menu option, or
From the main menu/toolbar banner at the top of the Enterprise Architect screen, right-click to display the context menu and select the Tasks Pane option.
3.13.14 The Pan & Zoom Window

The Pan & Zoom window provides a 'birds-eye' view of diagrams. It enables you to navigate quickly around large diagrams. To display this window, click on the View | More Windows | Pan & Zoom menu option.

The shaded box represents the viewed area on the open diagram. 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.

Move the cursor inside the window and hold down the mouse button to pan over the open diagram by moving the shaded box. To zoom, use either the Zoom Slider or the buttons located on the tool bar.
3.14 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 arrow is displayed at the upper right corner of the element, as shown below:

```
Simply clicking and dragging the icon enables you to create new connectors and elements on a diagram, as explained in the following topics:

- Create New Elements
- Create Connectors
```

The connectors and elements suggested by the Quick Linker are the commonest objects appropriate to the context. You can select others from the Enterprise Architect UML Toolbox pages. Also, a Technology Developer can edit the lists of elements and connectors, and create new combinations. For further information, see SDK for Enterprise Architect.
3.14.1 Create New Elements

To create new elements using the Quick Linker, follow the steps below:

1. Select a start element on the current diagram.

2. Drag the Quick Linker arrow onto an empty area of the diagram.

3. Use the Quick Linker context menu to select the type of element and connector to create.

Tips:
- Press and hold [Shift] while selecting the type of connector to select an existing classifier as the target.
- 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 Linker options depend on the type of element selected. For example, the Quick Linker options for a Class (above) differ from those of an Actor (below). These are the most appropriate, commonly used elements and connectors for the source element; you can create other target elements and connectors by selecting them from the appropriate Toolbox page.
3.14.2 Create Connectors

To create new connectors between existing elements using the Quick Linker, follow the steps below:

1. Select the source element on the current diagram.

2. Drag the **Quick Linker** arrow onto another element in the diagram.
3. Release the mouse button and use the **Quick Linker** context menu to select the type of connector.

**Notes:**
- The list of connectors provides the most appropriate, commonly-used connectors for the source and target element types. If you want to use a different connector, select the appropriate Toolbox page, click on the required connector and then on the source element, and drag across to the target element.
- The connector does not actually establish until you release the mouse button over the target element. However, a dotted line shows where the connector would be at any point, and the solid outline of the nearest element or extension changes to a hatched outline as you move the cursor onto it; this helps you identify where the connector will connector to, if there are many closely-arranged elements, Parts, Ports and other extensions.
- You can also bend the connector, pressing [Shift] as you drag the cursor in a new direction.
3.15 Defaults and User Settings

You can configure various settings using the Options dialog, which you display by selecting the Tools | Options menu option. In addition, there are several options to change the overall look and feel of Enterprise Architect in the View | Visual Style submenu. Those settings and options are explored in this topic.

See Also
- Custom Layouts
3.15.1 Configure Local Options

There are several options to customize how Enterprise Architect displays and works with models and model elements. This topic describes those settings that are local to a particular user and machine.

Select the Tools | Options menu option to display the Options dialog.

Most of these settings are stored in your registry so they are set for your use only. For a networked workplace, registry settings can be copied down to any network workstation you log in to. Otherwise, the settings are valid for the current machine only.

You select the required page of options by clicking on the appropriate category name in the left hand list on the dialog. For information on the options on a specific page, select the appropriate page title below.

- General
- Standard Colors
- Diagrams
- Diagram Behavior
- Diagram Sequence
- Objects
- Links
- Communication Message Colors
- XML Specifications
- Source Code Engineering
- Code Editors
- Object Lifetimes
- Attribute/Operations
- ActionScript
- C
- C#
- Delphi
- Java
- PHP
- Python
- Visual Basic
- VB.Net

Note:

The options in the second and third columns above, and additional defaults and settings, are discussed under the various code generation and import/export topics in this User Guide.
3.15.1.1 General

The General page of the Options dialog is shown below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Set the default author when new elements are created and modifications made.</td>
</tr>
<tr>
<td>Clipboard Format</td>
<td>Set the graphic format in which to save image to the clipboard; Metafile has the best detail.</td>
</tr>
<tr>
<td>Double-click on Browser</td>
<td>Configure the Project Browser behavior.</td>
</tr>
<tr>
<td>Use Jet 4.0 - requires restart</td>
<td>Set JET 4.0 as the database engine; this ensures compatibility with .EAP files that are in turn compatible with versions of MS Access later than Access 97.</td>
</tr>
<tr>
<td>Use extended « and » characters</td>
<td>Apply the guillemet characters to stereotypes. For some double byte character sets, it is best to select this checkbox.</td>
</tr>
<tr>
<td>Allow custom RTF tags in reports</td>
<td>Enable you to use customized rich text format code in report templates when generating reports with the Legacy RTF Report Generator. From release 7.0 of Enterprise Architect, with the Rich Text Notes facility, this option is not really necessary.</td>
</tr>
<tr>
<td>Project Directory</td>
<td>Specify the default location of Enterprise Architect projects.</td>
</tr>
<tr>
<td>File Directory</td>
<td>Specify the default location for files.</td>
</tr>
<tr>
<td>Web Home</td>
<td>Specify the default home page to open when you click on the Home button in the internal web browser.</td>
</tr>
<tr>
<td>Web Search</td>
<td>Specify the default web page to open when clicking on the Web Search button in the internal web browser.</td>
</tr>
<tr>
<td>Web Mail</td>
<td>Specify the email server address (<a href="http://xxxxx/exchange/">http://xxxxx/exchange/</a>) for accessing email through the web browser within Enterprise Architect.</td>
</tr>
<tr>
<td>Confirm Deletes</td>
<td>Use or bypass the Confirm Delete dialog; only clear this dialog if you are an experienced user!</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Allow Free Sorting</td>
<td>Enable ‘free sorting’ of elements in the Project Browser regardless of type.</td>
</tr>
<tr>
<td>Show Stereotypes</td>
<td>Show element and feature stereotypes in the Project Browser.</td>
</tr>
<tr>
<td>Ensure Dropped is Visible</td>
<td>Open the destination folder in the Project Browser when you move and drop an element into it from another folder. If you deselect this checkbox, the destination folder stays closed when you drop the element into it.</td>
</tr>
<tr>
<td>Hide Properties Info Section</td>
<td>Hide or show the properties information status bar on the Properties window.</td>
</tr>
<tr>
<td>Always open maximized</td>
<td>Ensure that Enterprise Architect always starts up in a maximized window.</td>
</tr>
<tr>
<td>Main Diagram Tabs at Bottom (requires restart)</td>
<td>Display the diagram tabs at the bottom of the main view (default). Clear the checkbox to show the tabs at the top of the main view.</td>
</tr>
<tr>
<td>Max Hierarchy View Depth</td>
<td>Set the maximum number of levels the hierarchy opens to on the Hierarchy tab.</td>
</tr>
<tr>
<td>Open Hierarchy View to</td>
<td>Set the initial number of levels the hierarchy opens to on the Hierarchy tab.</td>
</tr>
</tbody>
</table>

### 3.15.1.2 Standard Colors

The **Standard Colors** page of the Options dialog enables you to set the display color of a range of objects and their backgrounds. On first use, the page displays the system default colors, as shown below:

![Standard Colors](image1.png)

To display the range of colors available for an item, or define a new color, click on the down arrow at the end of the appropriate field. The selection pallet displays.

![Selection Pallet](image2.png)

Click on the required color. This sets the field on the Standard Colors page to the selected color.
If you require a wider selection of colors, click on the **Other** button and select from the color chart, or customize a color using RGB/HSL codes.

If you decide to reset the color to the system default, click on the **Default** button.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper</td>
<td>Define the paper (background) color in diagrams.</td>
</tr>
<tr>
<td>Element Fill</td>
<td>Define the fill color of elements.</td>
</tr>
<tr>
<td>Element Line</td>
<td>Define the line color of elements.</td>
</tr>
<tr>
<td>Shadow</td>
<td>Define the color of element outline shadows.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Define the color of attribute text.</td>
</tr>
<tr>
<td>Method</td>
<td>Define the color of method (operation) text.</td>
</tr>
<tr>
<td>Note Color</td>
<td>Define the note background color.</td>
</tr>
<tr>
<td>Note Compartment Text</td>
<td>Define the color of text in the element Note compartment.</td>
</tr>
<tr>
<td>Screen</td>
<td>Define the screen (element) color.</td>
</tr>
<tr>
<td>Connector Line</td>
<td>Define the connector line color.</td>
</tr>
<tr>
<td>Behavior</td>
<td>Define the color for behaviors in Activity diagrams.</td>
</tr>
<tr>
<td>Show Project Custom</td>
<td>Enable use of project custom colors; for more information on setting</td>
</tr>
<tr>
<td>Colors in Element Format</td>
<td>getting the custom colors see the Get and Set Project Custom Colors</td>
</tr>
<tr>
<td>toolbar</td>
<td>topic.</td>
</tr>
<tr>
<td>Show Gradient Fill For</td>
<td>Switch between having a color gradient in the diagram background, or</td>
</tr>
<tr>
<td>Paper Color</td>
<td>having a solid, uniform background color.</td>
</tr>
<tr>
<td>Gradient Fill Direction</td>
<td>Select the direction for the color gradient within element boxes, or</td>
</tr>
<tr>
<td>for an Element</td>
<td>&lt;none&gt; for no color gradient.</td>
</tr>
</tbody>
</table>

**Notes:**

- Using this page of the **Options** dialog, you can set the background of a diagram to be a specific color and to be either a uniform color or to have a fade gradient from top to bottom. Alternatively, you can create a background image for the diagram; see the Create Custom Diagram Background topic.

- To override the default appearance of a specific element on all diagrams on which it is found, right-click on the element and select the **Appearance | Default Appearance** menu option. The Default Appearance dialog displays.

- To change the appearance of a specific element on the current diagram only, use the Format toolbar. If the Format toolbar is not displayed, select the **View | Toolbars | Format Tool** menu option.
3.15.1.3 Diagram

The New Diagram Defaults page of the Options dialog enables you to configure overall options for new diagrams and general diagram behavior.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Page Size</td>
<td>Show the default page size for new diagrams, which you set by clicking on the Page Setup button to display the Page Setup dialog.</td>
</tr>
<tr>
<td>Print with Border</td>
<td>Print pages with a border.</td>
</tr>
<tr>
<td>Landscape</td>
<td>Print pages in landscape orientation. This checkbox is controlled from the Page Setup dialog.</td>
</tr>
<tr>
<td>Show Public Features</td>
<td>Set the default visibility of Class features.</td>
</tr>
<tr>
<td>Show Protected Features</td>
<td></td>
</tr>
<tr>
<td>Show Private Features</td>
<td></td>
</tr>
<tr>
<td>Show Diagram Notes</td>
<td>Display the diagram details in the top left corner of all diagrams in the model. Details include diagram name, package, version and author.</td>
</tr>
<tr>
<td>Use Opaque Diagram Labels</td>
<td>Specify where opaque diagram labels should display. Screen and Printing are best, Clipboard and Files might not be desirable.</td>
</tr>
<tr>
<td>Strict UML Syntax</td>
<td>Enforce compliance with UML syntax when adding new connectors and other structures.</td>
</tr>
<tr>
<td>Disable fully scoped object names</td>
<td>Disable fully scoped object names, when an element is in a diagram; don't use when the element is in its home package. A scoped name is of the format MyClasses::foo, the :: character indicating that the Class is within another namespace.</td>
</tr>
<tr>
<td>Allow change of Created Date</td>
<td>Enable the creation date on the Diagram Properties dialog to be altered.</td>
</tr>
<tr>
<td>Zoom to best scale</td>
<td>Resize diagrams to neatly fit the screen.</td>
</tr>
<tr>
<td>Auto-pan with middle</td>
<td>Turn on auto-panning using the middle mouse button. With this option off, the middle mouse button causes a different type of panning.</td>
</tr>
</tbody>
</table>
### Default and User Settings

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mouse button</strong></td>
<td>Determine the behavior of Enterprise Architect when text at zoom levels other than 100% would not fit inside the boundary of an element. Enterprise Architect either compresses the text to fit within the boundary, or expands the element.</td>
</tr>
<tr>
<td><strong>Compress text to fit within Element</strong></td>
<td>Force text anti-aliasing in diagrams. If you deselect the checkbox, Enterprise Architect applies the MS Windows default setting. Therefore, if you do not want to use anti-aliasing, ensure that the Windows anti-aliasing default is also set to OFF.</td>
</tr>
<tr>
<td><strong>Anti-aliased text</strong></td>
<td>Enable Enterprise Architect to save bitmaps at a higher resolution, suitable for using in published works.</td>
</tr>
<tr>
<td><strong>Scale Saved Bitmaps to</strong></td>
<td>Set an image memory limit 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 Enterprise Architect starts to scale down the image; a low memory setting means it scales the image sooner.</td>
</tr>
<tr>
<td><strong>Image Memory Limit</strong></td>
<td>Select where diagram frames are automatically added to images of diagrams in files saved to disk, print-outs, and the default Enterprise Architect clipboard. A diagram frame is a labeled outline around the diagram image, providing both a border and a reference.</td>
</tr>
</tbody>
</table>

#### 3.15.1.3.1 Behavior

The **Diagram Behavior** page of the Options dialog is shown below:

<table>
<thead>
<tr>
<th>Diagram Behavior</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Instance</td>
<td>Show Linked Items in Package</td>
</tr>
<tr>
<td>Instance has Classifier style</td>
<td>Show Package Item Scope</td>
</tr>
<tr>
<td>Show Life Lines</td>
<td>Shift-Mouse Autoresize</td>
</tr>
<tr>
<td>Print in Color</td>
<td>Show Hidden Parents</td>
</tr>
<tr>
<td>Layout uses all relations</td>
<td>Use Automatic SubActivities</td>
</tr>
<tr>
<td>Auto Save Changes</td>
<td>Always Highlight Contained Elements</td>
</tr>
<tr>
<td>Auto Group Elements (hold Alt to toggle)</td>
<td>Use Watermark</td>
</tr>
<tr>
<td>Watermark</td>
<td>Text</td>
</tr>
</tbody>
</table>

**Grid**

- Objects Snap to Grid
- Show Grid

**Grid Size**: 10

**Snap Lines**

- Auto Tidy
- Tidy line gap: 12

**Alias usage**

- Alias only
- Alias and Name
- Use alias field for partition/role

### Enterprise Architect User Guide
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>instantiated from when the instance is created.</td>
</tr>
<tr>
<td>Show Life Lines</td>
<td>Show life lines for Sequence elements in non-Sequence diagrams.</td>
</tr>
<tr>
<td>Print in Color</td>
<td>Print your diagrams in color. Deselect the checkbox to print the diagrams in black and white.</td>
</tr>
<tr>
<td>Layout uses all relations</td>
<td>Show all relationships in a diagram layout; deselect the checkbox to show only Generalizations and Associations.</td>
</tr>
<tr>
<td>Auto Save Changes</td>
<td>Automatically save your changes as you work, without having to confirm prompts to do so.</td>
</tr>
<tr>
<td>Auto Group Elements</td>
<td>Also move visually composed elements when moving diagram nodes. A node is considered composed if it is contained by the moved element and has a higher z-order. Press and hold [Alt] whilst moving an element to toggle this option.</td>
</tr>
<tr>
<td>Show Linked Items in Package</td>
<td>Display connected items on packages.</td>
</tr>
<tr>
<td>Show Package Item Scope</td>
<td>Display the + and - indicators representing the scope of the items.</td>
</tr>
<tr>
<td>Shift-Mouse Autoscroll</td>
<td>Enable you to press and hold [Shift] and use the mouse to autoscroll around diagrams.</td>
</tr>
<tr>
<td>Show 'Hidden' Parents</td>
<td>Display any parents of elements in the diagram that are not part of the diagram.</td>
</tr>
<tr>
<td>Use Automatic SubActivities</td>
<td>Generate a new Structured Activity linked to the diagram from a Structured Activity diagram dragged from the Project Browser.</td>
</tr>
<tr>
<td>Always Highlight Context Element</td>
<td>Show a hatch border around a selected element.</td>
</tr>
<tr>
<td>Use Watermark</td>
<td>Add a watermark to any diagrams you print.</td>
</tr>
<tr>
<td>Text</td>
<td>Define the watermark text, if a watermark is to be used.</td>
</tr>
<tr>
<td>Objects Snap to Grid</td>
<td>Snap all elements to the grid lines.</td>
</tr>
<tr>
<td>Show Grid</td>
<td>Display the grid.</td>
</tr>
<tr>
<td>Grid Size</td>
<td>Specify the grid size, if you have selected Objects Snap to Grid.</td>
</tr>
<tr>
<td>Auto Tidy</td>
<td>Automatically tidy line angles for custom connectors. This 'nudges' the custom line into horizontal and vertical increments.</td>
</tr>
<tr>
<td>Tidy line gap</td>
<td>Specify the amount Enterprise Architect should enable you to move a line away from horizontal and vertical when you are tidying lines for custom connectors. (See Auto Tidy above).</td>
</tr>
<tr>
<td>Alias only</td>
<td>Display the alias instead of the element name on elements with aliases.</td>
</tr>
<tr>
<td>Alias and Name</td>
<td>Display both the element name and the Alias in the format (Alias) name.</td>
</tr>
<tr>
<td>Use alias field for partition/role</td>
<td>Replace the Alias property of instances with a Role property.</td>
</tr>
</tbody>
</table>
### 3.15.1.3.2 Sequence

The Sequence page of the Options dialog enables you to configure various font settings and the focus of the control indicator for Sequence diagrams.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Spacing</td>
<td>Specify the vertical gap (in points) between Sequence messages (can be overridden manually by dragging a message up or down).</td>
</tr>
<tr>
<td>Default page layout is Landscape</td>
<td>Set the default orientation of Sequence diagrams to landscape.</td>
</tr>
<tr>
<td>Show Sequence Numbering</td>
<td>Show sequence numbers on Sequence messages.</td>
</tr>
<tr>
<td>Assume right to left messages are Return</td>
<td>Automatically generate return messages.</td>
</tr>
<tr>
<td>Default concurrency is Asynchronous</td>
<td>Set the default concurrency for Sequence Messages to Asynchronous; deselect to set the default concurrency to Synchronous.</td>
</tr>
<tr>
<td>Width</td>
<td>Select the line width (in points) of the ‘focus of control’ rectangle (thick part of lifeline).</td>
</tr>
<tr>
<td>Assume message returns</td>
<td>Assume implicit returns when none are explicitly drawn (recommended).</td>
</tr>
<tr>
<td>GarbageCollect</td>
<td>Automatically truncate lifelines for created elements after the last message (that is, assume garbage collect rather than explicit delete).</td>
</tr>
<tr>
<td>Name</td>
<td>Display the MS Windows Font dialog (click on [...] ) and define the font of the caption bar heading (above your diagram); this is particularly useful for non-English character sets.</td>
</tr>
<tr>
<td>Size</td>
<td>Specify the size of the heading font (this overrides the font size in the Font dialog, above).</td>
</tr>
<tr>
<td>Use System</td>
<td>Apply the Enterprise Architect system default heading font.</td>
</tr>
</tbody>
</table>

---

![Sequence Options Dialog](image-url)
### 3.15.1.4 Objects

The Objects page of the Options dialog enables you to configure how elements look and respond in diagrams.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Border Width</td>
<td>Set the default border width (in pixels).</td>
</tr>
<tr>
<td>Configure Default Fonts</td>
<td>Set the default model and user text fonts</td>
</tr>
<tr>
<td>Automatically increase size of all objects</td>
<td>Automatically increase the size of all objects on a diagram by up to 50%, without affecting other users reading that diagram.</td>
</tr>
<tr>
<td>Version</td>
<td>Set the default version for new elements.</td>
</tr>
<tr>
<td>Phase</td>
<td>Set the default phase for new elements.</td>
</tr>
<tr>
<td>Highlight References</td>
<td>Highlight parameters in operations that are passed by reference rather than value.</td>
</tr>
<tr>
<td>Reference Char(s)</td>
<td>Specify a character to use for the reference.</td>
</tr>
<tr>
<td>Suffix/Prefix</td>
<td>Indicate whether to use the Reference Char(s) value as a prefix (before) or a suffix (after).</td>
</tr>
<tr>
<td>No warning for spaces in class names</td>
<td>Bypass or display the warning message if a Class name has embedded spaces.</td>
</tr>
<tr>
<td>Classes honor analysis stereotypes</td>
<td>Show Classes as their stereotype; e.g. if a Class is stereotyped as a Boundary, it appears as a Boundary rather than a Class.</td>
</tr>
<tr>
<td>Show stereotype icon for requirements</td>
<td>Show or hide a code letter in the top right corner of Requirement (E, for external), Change (C) and Issue (I) elements.</td>
</tr>
<tr>
<td>Support for Composite Objects</td>
<td>Enable you to drag child elements onto parent elements in a diagram and automatically embed them (and drag embedded child elements out of parent elements, breaking the child-parent relationship).</td>
</tr>
<tr>
<td>Auto-resize marks diagram ‘dirty’</td>
<td>Ensure that auto-resizing of elements (e.g. Classes) marks the current diagram as changed (asterisk on the diagram name tab), so it should be saved.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Highlight (abstract) elements</td>
<td>Highlight abstract elements with a suitable tag (abstract) in the top right of the Class.</td>
</tr>
<tr>
<td>Allow elongated Use Cases</td>
<td>Stretch Use Cases or Use Case extension points with long names to enable space for the name. If you deselect the checkbox, Use Case re-sizing is proportional.</td>
</tr>
<tr>
<td>Show status colors on diagrams</td>
<td>Enable color coding ( \text{Ctrl}+\text{Shift}+\text{V} ) for Requirements and similar elements.</td>
</tr>
<tr>
<td>Copy inheritance links on duplicate</td>
<td>Duplicate Inheritance and Realization connectors when an edit/copy is performed ( \text{Ctrl}+\text{Shift}+\text{V} ).</td>
</tr>
<tr>
<td>Port and Part type visible by default</td>
<td>Enable Port and Part types to be shown by default.</td>
</tr>
<tr>
<td>Sort Features Alphabetically</td>
<td>Sort element features alphabetically. Features include Attributes, Operations, Tags, Constraints and Test Cases.</td>
</tr>
<tr>
<td>Bold Object Names</td>
<td>Show elements in bold face in diagrams.</td>
</tr>
<tr>
<td>Shadows On</td>
<td>Toggle element shadows.</td>
</tr>
<tr>
<td>Edit Object on New</td>
<td>Automatically show the element Properties dialog when a new element is added.</td>
</tr>
<tr>
<td>Show «column» stereotype</td>
<td>Hide or show the «column» stereotype used when data modeling.</td>
</tr>
<tr>
<td>Extend Complexity</td>
<td>Extend levels of complexity to five levels in the Complexity option in the Properties window. Otherwise only three levels are available.</td>
</tr>
<tr>
<td>UML 1.5 Components</td>
<td>Use UML 1.5 components (Enterprise Architect versions 4 and later support UML 2.x).</td>
</tr>
<tr>
<td>Show State Compartment</td>
<td>Show or hide the State Compartment divider under the state name.</td>
</tr>
<tr>
<td>Show Duplicate Tags</td>
<td>Enable duplicate tags to be shown.</td>
</tr>
<tr>
<td>Group Operations by Stereotype</td>
<td>Group an element's operations by their stereotype on the diagram.</td>
</tr>
<tr>
<td>Group Attributes by Stereotype</td>
<td>Group an element's attributes by their stereotype on the diagram.</td>
</tr>
<tr>
<td>Inverted rotated text for metafiles</td>
<td>Use different text format when external metafile readers are causing issues.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Set the visibility of certain elements in reports and in diagram packages.</td>
</tr>
</tbody>
</table>

### 3.15.1.4.1 Set Default Fonts

Enterprise Architect enables you to define a standard font to apply across the model, or a font to apply to any diagrams you create personally. You can define both, but the model font overrides any user font, to ensure that all members of a project team have a consistent and coherent view of the model. This avoids the problem of one user creating a diagram in a small font, and another user trying to view it in a larger font, which distorts the diagram.

It is recommended that a project authority sets the model default, and all project members abide by it and do not change it without project approval.

To set the default fonts, follow the steps below:

1. On the Objects page of the Options dialog, click on the Configure Default Fonts button. The Configure Default Fonts dialog displays.
2. To set a model font, in the **Font Face** field of the **Model Font** panel, click on the drop-down arrow and select the appropriate typeface.

3. In the **Font Size** field, click on the drop-down arrow and select the required font size.

4. To clear a model font so that the user font takes effect, click on the **Clear** button. (Ensure that this is acceptable to all other team members.)

5. To set a user font, in the **Font Face** field of the **User Font** panel, click on the drop-down arrow and select the appropriate typeface.

6. In the **Font Size** field, click on the drop-down arrow and select the required font size.

7. To return the user font to the Enterprise Architect default (Arial 8), click on the **Restore Defaults** button.

8. To save the changes, click on the **OK** button.

Both model and user fonts are overridden by specifically-defined element fonts, so that the element is viewed as designed regardless of the model or user defaults. To define the font for a specific element, right-click on the element in a diagram and select the **Appearance | Set Font** context menu option.

If you cannot read the diagrams because the default font makes the objects and text too small, you can scale up all objects (that is, all diagram displays) to a more readable size. Everything on the diagram is enlarged to the same extent, so it remains in proportion and readable. To do this, return to the **Objects** page of the **Options** dialog and enter a suitable percentage value in the **Automatically increase size of all objects** field.

### 3.15.1.4.2 Element Visibility

Some elements do not appear in packages and in RTF output by default. Click on the **Advanced** button on the **Objects** page of the **Options** dialog to specify which elements should be visible.

See the topic on **customizing element visibility** for more details.
### 3.15.1.5 Links

The Links page of the Options dialog shown below provides options for the creation, behavior, and notation for connectors.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Edit Connector on New</strong></td>
<td>Automatically show the connector Properties dialog when a new connector is added.</td>
</tr>
<tr>
<td><strong>Association default = source -&gt; target</strong></td>
<td>Set the direction of new Associations to source-&gt;target (i.e. with an arrow head at the target).</td>
</tr>
<tr>
<td><strong>Generalization link style Default = Tree</strong></td>
<td>Show Generalizations as tree style hierarchies.</td>
</tr>
<tr>
<td><strong>Shade Qualifier boxes</strong></td>
<td>Lightly shade all Qualifier boxes.</td>
</tr>
<tr>
<td><strong>Draw Aggregations Reversed</strong></td>
<td>Draw Aggregate and Composite connectors from target element to source element. When deselected (the default), these connectors are drawn from source to target.</td>
</tr>
<tr>
<td><strong>Prompt on connector deletes</strong></td>
<td>Display a prompt before deleting connectors, offering the choice of hiding the connector on the diagram or deleting it completely. If you deselect this option, the delete operation defaults to the last setting on the dialog.</td>
</tr>
<tr>
<td><strong>Suppress Link Constraints</strong></td>
<td>Suppress connector constraints in diagrams.</td>
</tr>
<tr>
<td><strong>Suppress Qualifier boxes</strong></td>
<td>Suppress boxes when displaying qualifiers.</td>
</tr>
</tbody>
</table>

**Note:**

All tools 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.
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Show Uses arrowheads</strong></td>
<td>Show an arrowhead on Actor-&gt;Use Case Associations.</td>
</tr>
<tr>
<td><strong>Show Override Operation dialog on new connector</strong></td>
<td>Show the Override Operation dialog automatically when adding generalizations and realizations between Classes and Interfaces, if the target element has features that can be overridden.</td>
</tr>
<tr>
<td><strong>Suppress ’ + ’ Role Scope</strong></td>
<td>Ensure that the role and scope are not displayed on the diagram.</td>
</tr>
<tr>
<td><strong>Default Style</strong></td>
<td><strong>Pen Width</strong> — Set the default connector width.</td>
</tr>
<tr>
<td><strong>Routing</strong></td>
<td><strong>Routing</strong> — Set the default connector style for new connectors.</td>
</tr>
<tr>
<td><strong>Quick Linker</strong></td>
<td><strong>Enable</strong> — Enable the Quick Linker.</td>
</tr>
<tr>
<td></td>
<td><strong>Show Help</strong> — Add a ‘help’ menu option to the end of the Quick Linker menu.</td>
</tr>
<tr>
<td><strong>New Connector End-Points</strong></td>
<td><strong>Center to center</strong> — Change the position of the dashed guide line for new connectors.</td>
</tr>
<tr>
<td><strong>Exact placement</strong></td>
<td><strong>Force perpendicular line</strong> — Change the position of the dashed guide line for new connectors.</td>
</tr>
</tbody>
</table>

### 3.15.1.6 Communication Message Colors

The Communication Message Coloring page in the Options dialog enables you to configure the colors used in Communication diagrams. When you enable this option, Communication messages appear in different colors depending on the sequence group they belong to on a diagram; e.g. 1.n are black, 2.n are red, 3.n are green.

Select the Use Communication Message Coloring checkbox to turn on colored messages.

Click on the down arrow in each color field, and click on the appropriate color for the message group. Set the color sequence as required; the pattern repeats after 8 sequence groups.
### 3.15.1.7 XML Specifications

The XML Specifications page of the Options dialog enables you to configure various settings for working with XML.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editor</td>
<td>Set the default editor for any XML documents you open within Enterprise Architect.</td>
</tr>
<tr>
<td>Format XML Output</td>
<td>Set whether or not formatting is applied to your XML output.</td>
</tr>
<tr>
<td>Write Log</td>
<td>Set whether or not to write to a log file when you import or export XML.</td>
</tr>
<tr>
<td>Use DTD</td>
<td>Set whether or not to use a Data Type Definition.</td>
</tr>
<tr>
<td>Export Diagrams</td>
<td>Set whether or not to export diagrams when you export XML.</td>
</tr>
<tr>
<td>Export Alternate Images</td>
<td>Set whether or not to export the alternative images used in the model when you export to XML.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If this option is set, and you have packages in your model under version control, then any alternative images used in those packages are also exported to the version control repository when you check in the packages.</td>
</tr>
<tr>
<td></td>
<td>In this case, you would only select the checkbox if the alternative images are subject to frequent change. Otherwise, do not select this option and instead use <code>Export/Import Reference Data</code> to manage alternative images.</td>
</tr>
<tr>
<td>Default XMI Version</td>
<td>Set the XMI version to use: Enterprise Architect or Rose.</td>
</tr>
<tr>
<td>Code Page</td>
<td>Set the Code Page to use; setting a NULL encoding string results in the encoding tag being entirely omitted from the XML output. Click on the Default button to restore the setting to the default Code Page.</td>
</tr>
<tr>
<td>Export Diagram Images</td>
<td>Set whether or not to export diagrams as images when you export XML.</td>
</tr>
<tr>
<td>Image Type</td>
<td>Define the format of the image to export to if <code>Export Diagram Images</code> is selected.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Prefix EA Tagged Values in XMI 1.0 with ea$</td>
<td>Set whether or not to prefix any Enterprise Architect Tagged Values within any XMI 1.0 you create with ea$.</td>
</tr>
<tr>
<td>Update Last Save Time in Controlled Packages</td>
<td>Set whether to update the time you last saved in controlled packages.</td>
</tr>
<tr>
<td>Default XML Directory</td>
<td>Define the default XML directory to use when importing and exporting XML.</td>
</tr>
</tbody>
</table>
3.15.2 Custom Layouts

Enterprise Architect 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.

To access this feature, select the View | Visual Layouts menu option.

When you select an option Enterprise Architect realigns toolbars and docked windows to the defaults for that option.

Setting User Layouts

If you have the layout of the Enterprise Architect windows and tools just as you require them, you can preserve your layout by saving it as a custom user layout. Select the View | Visual Layouts | Custom Layouts menu option. The Custom Layouts dialog displays:

You can save up to five custom layouts. To save the current layout to Layout 1, ensure the Layout 1 option is selected, then click on the Save Layout to Selected button.

If you make changes to an existing layout, to save them open the Custom Layouts dialog, select the radio button for the layout to update, then click on the Save Layout to Selected button.

Load a saved layout by going to the Custom Layouts dialog, selecting the radio button for the layout to load, and click on the Load Selected Layout button.

Warning:

If you have set keyboard shortcuts, these are not 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 Custom Layout, your keyboard shortcuts are overridden, unless you have saved them as part of the custom layout you have switched to. For more information about setting keyboard shortcuts, see the Customize Keyboard topic.
3.15.3 Visual Styles

You can configure the overall look and feel of Enterprise Architect to suit your working environment. Options range from a classic Windows application to an enhanced XP look.

To reset the appearance of Enterprise Architect, follow the steps below:

1. Select the View | Visual Style submenu.
2. Select the required style from the list:
   - XP Similar to applications such as MS Office and MS Visual Studio
   - 2003 Colorful modern style
   - 2005 Rounded modern style
   - 2007 Dark Visual Studio style
   - Classic Classic Windows application look.

You can also enable or disable menu shadows and animate auto-hidden windows.
### 3.16 Keyboard Shortcuts

The table below lists the default keyboard shortcut functions within Enterprise Architect. You can also display the key combinations on the Help Keyboard dialog (or Keyboard Accelerator Map).

If necessary, you can change these keyboard shortcuts using the Keyboard tab of the Customize dialog.

<table>
<thead>
<tr>
<th>Function</th>
<th>Shortcut</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new Enterprise Architect project</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open an Enterprise Architect project</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open Source File</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reload the current project</strong></td>
<td>[Ctrl]+[Shift]+[F11]</td>
<td></td>
</tr>
<tr>
<td>Print the active diagram</td>
<td>[Ctrl]+[P]</td>
<td></td>
</tr>
<tr>
<td><strong>Undo Change</strong></td>
<td>[Ctrl]+[Z]</td>
<td>Edit</td>
</tr>
<tr>
<td><strong>Redo Change</strong></td>
<td>[Ctrl]+[Y]</td>
<td>Edit</td>
</tr>
<tr>
<td>Add a single element to the clipboard list</td>
<td>[Ctrl]+[Space]</td>
<td>Edit</td>
</tr>
<tr>
<td>Paste element as metafile from clipboard</td>
<td>[Ctrl]+[Shift]+[Insert]</td>
<td>Edit</td>
</tr>
<tr>
<td>Paste element as new</td>
<td>[Ctrl]+[Shift]+[V]</td>
<td>Edit</td>
</tr>
<tr>
<td>Paste element(s) from the clipboard as a link</td>
<td>[Shift]+[Insert]</td>
<td>Edit</td>
</tr>
<tr>
<td><strong>Bookmark current element with red marker</strong></td>
<td>[Shift]+[Space]</td>
<td>Edit</td>
</tr>
<tr>
<td>Delete selected element(s) in diagram</td>
<td>[Delete] or [Ctrl]+[D]</td>
<td>Edit</td>
</tr>
<tr>
<td>Delete selected element(s) from model (through diagram OR Project Browser)</td>
<td>[Ctrl]+[Delete]</td>
<td>Edit</td>
</tr>
<tr>
<td>Search for elements in the project</td>
<td>[Ctrl]+[F]</td>
<td>Edit</td>
</tr>
<tr>
<td>Set focus to current window</td>
<td>[Ctrl]+[Shift]+[0]</td>
<td>Window</td>
</tr>
<tr>
<td>Autohide the current window</td>
<td>[Ctrl]+[Shift]+[F4]</td>
<td>Window</td>
</tr>
<tr>
<td>Hide the current window</td>
<td>[Ctrl]+[F4]</td>
<td>Window</td>
</tr>
<tr>
<td><strong>View Project Browser</strong></td>
<td>[Alt]+[0]</td>
<td>View</td>
</tr>
<tr>
<td>View Properties window</td>
<td>[Alt]+[1]</td>
<td>View</td>
</tr>
<tr>
<td>View System window</td>
<td>[Alt]+[2]</td>
<td>View</td>
</tr>
<tr>
<td><strong>View Testing window</strong></td>
<td>[Alt]+[3]</td>
<td>View</td>
</tr>
<tr>
<td>View Maintenance window</td>
<td>[Alt]+[4]</td>
<td>View</td>
</tr>
<tr>
<td>Display Enterprise Architect UML Toolbox</td>
<td>[Alt]+[5]</td>
<td>View</td>
</tr>
<tr>
<td>View Resources window</td>
<td>[Alt]+[6]</td>
<td>View</td>
</tr>
<tr>
<td>View Source Code window</td>
<td>[Alt]+[7]</td>
<td>View</td>
</tr>
<tr>
<td>View Debug Workbench</td>
<td>[Alt]+[8]</td>
<td>View</td>
</tr>
<tr>
<td>View Notes window</td>
<td>[Ctrl]+[Shift]+[1]</td>
<td>View</td>
</tr>
<tr>
<td><strong>View Element Relationships window</strong></td>
<td>[Ctrl]+[Shift]+[2]</td>
<td>View</td>
</tr>
<tr>
<td><strong>View Rules and Scenarios (Requirements and Constraints) window</strong></td>
<td>[Ctrl]+[Shift]+[3]</td>
<td>View</td>
</tr>
<tr>
<td><strong>View Hierarchy window</strong></td>
<td>[Ctrl]+[Shift]+[4]</td>
<td>View</td>
</tr>
<tr>
<td><strong>View Tagged Values window</strong></td>
<td>[Ctrl]+[Shift]+[6]</td>
<td>View</td>
</tr>
<tr>
<td><strong>View Project Management window</strong></td>
<td>[Ctrl]+[Shift]+[7]</td>
<td>View</td>
</tr>
<tr>
<td>Function</td>
<td>Shortcut</td>
<td>Category</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------</td>
<td>------------</td>
</tr>
<tr>
<td>View Output window</td>
<td>[Ctrl]+[Shift]+[8]</td>
<td>View</td>
</tr>
<tr>
<td>View Tasks Pane</td>
<td>[Ctrl]+[Shift]+[9]</td>
<td>View</td>
</tr>
<tr>
<td>View Pan &amp; Zoom Window</td>
<td>[Ctrl]+[Shift]+[N]</td>
<td>View</td>
</tr>
<tr>
<td>View Model Search</td>
<td>[Ctrl]+[Alt]+[A]</td>
<td>View</td>
</tr>
<tr>
<td>View Element List</td>
<td>[Ctrl]+[Alt]+[R]</td>
<td>View</td>
</tr>
<tr>
<td>Open Discussion Forum</td>
<td>[Ctrl]+[Alt]+[U]</td>
<td>View</td>
</tr>
<tr>
<td>Display Web Browser</td>
<td>[Ctrl]+[Alt]+[W]</td>
<td>View</td>
</tr>
<tr>
<td>View Element Browser</td>
<td>[Alt]+[9]</td>
<td>View</td>
</tr>
<tr>
<td>Add new package to project</td>
<td>[Ctrl]+[W]</td>
<td>Project</td>
</tr>
<tr>
<td>Add new diagram to package</td>
<td>[Ctrl]+[Insert]</td>
<td>Project</td>
</tr>
<tr>
<td>Add new element to package</td>
<td>[Ctrl]+[M]</td>
<td>Project</td>
</tr>
<tr>
<td>Create RTF documentation</td>
<td>[F8]</td>
<td>Project</td>
</tr>
<tr>
<td>Generate HTML Report</td>
<td>[Shift]+[F8]</td>
<td>Project</td>
</tr>
<tr>
<td>Generate Diagrams-only Report</td>
<td>[Ctrl]+[Shift]+[F8]</td>
<td>Project</td>
</tr>
<tr>
<td>Generate package source code</td>
<td>[Ctrl]+[Alt]+[K]</td>
<td>Project</td>
</tr>
<tr>
<td>Synchronize package contents</td>
<td>[Ctrl]+[Alt]+[M]</td>
<td>Project</td>
</tr>
<tr>
<td>Import source directory</td>
<td>[Ctrl]+[Shift]+[U]</td>
<td>Project</td>
</tr>
<tr>
<td>Package Build Scripts</td>
<td>[Shift]+[F12]</td>
<td>Project</td>
</tr>
<tr>
<td>Build</td>
<td>[Ctrl]+[Shift]+[F12]</td>
<td>Project</td>
</tr>
<tr>
<td>Test</td>
<td>[Ctrl]+[Alt]+[T]</td>
<td>Project</td>
</tr>
<tr>
<td>Run</td>
<td>[Ctrl]+[Alt]+[N]</td>
<td>Project</td>
</tr>
<tr>
<td>Deploy</td>
<td>[Ctrl]+[Shift]+[Alt]+[F12]</td>
<td>Project</td>
</tr>
<tr>
<td>Debug Run</td>
<td>[F6]</td>
<td>Project</td>
</tr>
<tr>
<td>Step Into</td>
<td>[Shift]+[F6]</td>
<td>Project</td>
</tr>
<tr>
<td>Step Over</td>
<td>[Alt]+[F6]</td>
<td>Project</td>
</tr>
<tr>
<td>Step Out</td>
<td>[Ctrl]+[F6]</td>
<td>Project</td>
</tr>
<tr>
<td>Debug Stop</td>
<td>[Ctrl]+[Alt]+[F6]</td>
<td>Project</td>
</tr>
<tr>
<td>Transform selected elements</td>
<td>[Ctrl]+[H] or</td>
<td>Project</td>
</tr>
<tr>
<td></td>
<td>[Ctrl]+[Alt]+[F]</td>
<td></td>
</tr>
<tr>
<td>Transform current package</td>
<td>[Ctrl]+[Shift]+[H]</td>
<td>Project</td>
</tr>
<tr>
<td>Validate Selected</td>
<td>[Ctrl]+[Alt]+[V]</td>
<td>Project</td>
</tr>
<tr>
<td>Manage locks applied by current user</td>
<td>[Ctrl]+[Shift]+[L]</td>
<td>Project</td>
</tr>
<tr>
<td>Configure package control</td>
<td>[Ctrl]+[Alt]+[P]</td>
<td>Project</td>
</tr>
<tr>
<td>Import package from XMI</td>
<td>[Ctrl]+[Alt]+[I]</td>
<td>Project</td>
</tr>
<tr>
<td>Export package to XMI</td>
<td>[Ctrl]+[Alt]+[E]</td>
<td>Project</td>
</tr>
<tr>
<td>Import and export to CSV files</td>
<td>[Ctrl]+[Alt]+[C]</td>
<td>Project</td>
</tr>
<tr>
<td>Manage Baselines</td>
<td>[Ctrl]+[Alt]+[B]</td>
<td>Project</td>
</tr>
<tr>
<td>Diagram properties</td>
<td>[F5]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Save</td>
<td>[Ctrl]+[S]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Function</td>
<td>Shortcut</td>
<td>Category</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Save image to file</td>
<td>[Ctrl]+[T]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Save image to clipboard</td>
<td>[Ctrl]+[B]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Visible Relations</td>
<td>[Ctrl]+[Shift]+[I]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Locate in Project Browser</td>
<td>[Shift]+[Alt]+[G]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Repeat last element</td>
<td>[Shift]+[F3] or [Ctrl]+[click]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Repeat last connector</td>
<td>[F3]</td>
<td>Diagram</td>
</tr>
<tr>
<td>Element Properties</td>
<td>[Alt]+[Enter]</td>
<td>Element</td>
</tr>
<tr>
<td>Add Tagged Value</td>
<td>[Ctrl]+[Shift]+[T]</td>
<td>Element</td>
</tr>
<tr>
<td>Linked Document</td>
<td>[Ctrl]+[Alt]+[D]</td>
<td>Element</td>
</tr>
<tr>
<td>Display Attribute Properties dialog</td>
<td>[F9]</td>
<td>Element</td>
</tr>
<tr>
<td>Display Operation Properties dialog</td>
<td>[F10]</td>
<td>Element</td>
</tr>
<tr>
<td>Space elements evenly horizontally</td>
<td>[Alt]+[-]</td>
<td>Element</td>
</tr>
<tr>
<td>Space elements evenly vertically</td>
<td>[Alt]+[=]</td>
<td>Element</td>
</tr>
<tr>
<td>Add attribute</td>
<td>[Ctrl]+[Shift]+[F9]</td>
<td>Element</td>
</tr>
<tr>
<td>Add operation</td>
<td>[Ctrl]+[Shift]+[F10]</td>
<td>Element</td>
</tr>
<tr>
<td>Add other type</td>
<td>[Ctrl]+[F11]</td>
<td>Element</td>
</tr>
<tr>
<td>Auto-size selected elements</td>
<td>[Alt]+[Z]</td>
<td>Element</td>
</tr>
<tr>
<td>Generate code from element</td>
<td>[Ctrl]+[G] or [F11]</td>
<td>Element</td>
</tr>
<tr>
<td>Align bottom edges of selected elements</td>
<td>[Ctrl]+[Alt]+[Down]</td>
<td>Element</td>
</tr>
<tr>
<td>Align top edges of selected elements</td>
<td>[Ctrl]+[Alt]+[Up]</td>
<td>Element</td>
</tr>
<tr>
<td>Align selected elements on left boundaries</td>
<td>[Ctrl]+[Alt]+[Left]</td>
<td>Element</td>
</tr>
<tr>
<td>Align selected elements on right boundaries</td>
<td>[Ctrl]+[Alt]+[Right]</td>
<td>Element</td>
</tr>
<tr>
<td>Configure element default appearance</td>
<td>[Ctrl]+[Shift]+[E] or [F4]</td>
<td>Element</td>
</tr>
<tr>
<td>Edit selected</td>
<td>[F2]</td>
<td>Element</td>
</tr>
<tr>
<td>Manage embedded elements</td>
<td>[Ctrl]+[Shift]+[B]</td>
<td>Element</td>
</tr>
<tr>
<td>Insert new feature after current selection</td>
<td>[Insert]</td>
<td>Element</td>
</tr>
<tr>
<td>Locate in browser</td>
<td>[Alt]+[G]</td>
<td>Element</td>
</tr>
<tr>
<td>New element</td>
<td>[Ctrl]+[M]</td>
<td>Element</td>
</tr>
<tr>
<td>View source code in default editor</td>
<td>[Ctrl]+[E] or [F12]</td>
<td>Element</td>
</tr>
<tr>
<td>Operation</td>
<td>[F10]</td>
<td>Element</td>
</tr>
<tr>
<td>Override inherited features</td>
<td>[Ctrl]+[Shift]+[O]</td>
<td>Element</td>
</tr>
<tr>
<td>Configure element properties</td>
<td>[Alt]+[Enter]</td>
<td>Element</td>
</tr>
<tr>
<td>Select alternative image</td>
<td>[Ctrl]+[Shift]+[W]</td>
<td>Element</td>
</tr>
<tr>
<td>Specify which element features are visible on a diagram</td>
<td>[Ctrl]+[Shift]+[Y]</td>
<td>Element</td>
</tr>
<tr>
<td>Set element parent or implement interface(s)</td>
<td>[Ctrl]+[I]</td>
<td>Element</td>
</tr>
<tr>
<td>Set references to other elements and diagrams</td>
<td>[Ctrl]+[J]</td>
<td>Element</td>
</tr>
<tr>
<td>Create Workbench Instance</td>
<td>[Ctrl]+[Shift]+[J]</td>
<td>Element</td>
</tr>
<tr>
<td>View element usage</td>
<td>[Ctrl]+[U]</td>
<td>Element</td>
</tr>
</tbody>
</table>
### Display Keyboard Accelerator Map

To display the key combinations for the menu functions within Enterprise Architect, select the **Help | Keyboard Accelerator Map** menu option. The *Help Keyboard* dialog displays.

![Help Keyboard](image)

To list the shortcuts in a particular category (see the **Command** column in the above table), click on the drop-down arrow in the **Category** field and select the appropriate category.
3.17 Project Discussion Forum

The Enterprise Architect Project Discussion Forum can be used to discuss the development and progress of a project. The Discussion Forum window consists of three main areas:

- The message thread area, located in the left pane, is used to create new categories and topics and to create, edit, reply to, and delete messages.
- The message contents section, located in the top right hand section of the discussion forum, is used to view discussion topics.
- The linked objects area, located in the lower right hand portion of the discussion forum, is used to associate model elements and diagrams of interest with the forum message.

To access the Discussion Forum, either:

- Select the View | Project Discussion Forum menu option.
- Press [Ctrl]+[Alt]+[U], or
- Click on the Discussion Forum button in the Other Views toolbar.

The icons beside the messages have the following meanings:

- Post read
- Post unread
- Post Reply
- Unread Reply
- Category
- Category unread
- Topic read
- Topic unread.
3.17.1 Add 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 displays 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 clicking on the OK button. New topics can now be added to the category; see Add a New Topic.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Category</td>
<td>Create a new category.</td>
</tr>
<tr>
<td>New Topic</td>
<td>Create a new topic under the selected category in the tree.</td>
</tr>
<tr>
<td>Edit</td>
<td>Edit the currently select item, either a category, topic or post.</td>
</tr>
<tr>
<td>Copy Path To Clipboard</td>
<td>Copy the path to the currently selected item to the clipboard.</td>
</tr>
<tr>
<td>Refresh Category</td>
<td>Refresh the currently selected category.</td>
</tr>
<tr>
<td>Reload Discussion Forum</td>
<td>Reload the discussion forum. Used when multiple users are connected to a model on a server.</td>
</tr>
<tr>
<td>Options</td>
<td>Display the Options dialog.</td>
</tr>
<tr>
<td>Delete Category</td>
<td>Delete the category from the tree.</td>
</tr>
</tbody>
</table>
3.17.2 Add 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 displays 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 clicking on the OK button. New posts can now be added to the topic; see Add a New Post.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Topic</td>
<td>Create a new topic under the selected category in the tree.</td>
</tr>
<tr>
<td>Edit</td>
<td>Edit the currently select item, either a category, topic or post.</td>
</tr>
<tr>
<td>Copy Path To Clipboard</td>
<td>Copy the path to the currently selected item to the clipboard.</td>
</tr>
<tr>
<td>Refresh Category</td>
<td>Refresh the currently selected category.</td>
</tr>
<tr>
<td>Reload Discussion Forum</td>
<td>Reload the discussion forum. Used when multiple users are connected to a model on a server.</td>
</tr>
<tr>
<td>Options</td>
<td>Display the Options dialog.</td>
</tr>
<tr>
<td>Delete Category</td>
<td>Delete the currently selected category from the tree.</td>
</tr>
</tbody>
</table>
Add 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.

The Create New Post dialog displays. Enter the name and any relevant details into the text field, as well as the name of the author, before clicking on the OK button. You can also drag in element entries from the Model Search dialog, the Model Views window, or the Element List.

Other users can now reply to the post; see Reply to a Post.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Post</td>
<td>Create a new post under the selected topic in the tree.</td>
</tr>
<tr>
<td>Edit</td>
<td>Edit the currently select item, either a category, topic or post.</td>
</tr>
<tr>
<td>Copy Path To Clipboard</td>
<td>Copy the path of the currently selected item to the clipboard.</td>
</tr>
<tr>
<td>Refresh Topic</td>
<td>Refresh the currently selected topic.</td>
</tr>
<tr>
<td>Reload Discussion Forum</td>
<td>Reload the discussion forum. Used when multiple users are connected to a model on a server.</td>
</tr>
<tr>
<td>Options</td>
<td>Display the Options dialog.</td>
</tr>
<tr>
<td>Delete Topic</td>
<td>Delete the currently selected topic from the tree.</td>
</tr>
</tbody>
</table>
3.17.4 Reply to a Post

To reply to a post, right-click on the post in the message thread window. Select **Reply** from the context menu.

![Reply to Post dialog](image)

The **Reply to Post** dialog displays. Enter the name and any relevant details into the text field, as well as the name of the author, before clicking on the **OK** button.
3.17.5 Edit 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; alternatively press [Ctrl]+[E].

![Edit dialog](image)

The Edit dialog displays. Modify any relevant details in the text field. You cannot edit the name of the author. Click on the OK button.
3.17.6 Message Dialog

The project discussion forum Message dialogs for Create New Category, Create New Topic, Create New Post, Edit Post, and Reply to Post all share the same functionality.

<table>
<thead>
<tr>
<th>Name</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Discussion Forum</td>
<td>Frederick Waber</td>
</tr>
</tbody>
</table>

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 Topics
- Create new Posts and reply to Posts
- Format messages
- Link EA elements to the discussion.

The table below describes the operation of each option available for the dialogs.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the message category or message topic.</td>
</tr>
<tr>
<td>Author</td>
<td>Click on the drop-down arrow and select the message author or type in a new name if the Author name is not present in the list. (You cannot change this field when editing a topic.) The Authors in the drop-down list are defined in the Project Authors list. For more information see the Project Authors topic.</td>
</tr>
<tr>
<td>Formatting Tools</td>
<td>These are standard formatting options for text.</td>
</tr>
<tr>
<td>OK</td>
<td>Confirm the forum message.</td>
</tr>
</tbody>
</table>
3.17.7 Add Object Links

In the project Discussion Forum you can create hyperlinks to elements and diagrams that are associated with the messages (Posts and replies). This enables rapid navigation to the objects in the Project Browser, access to the element properties and, with diagrams, the ability to open the diagram directly from the forum. To associate an element or diagram with the forum message, drag the object from the Project Browser window or the Model Search dialog into the linked elements panel underneath the message text panel.

To access the navigation options of each element in the linked elements section, right-click on the object to display the navigation context menu. The options are outlined in the table below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Open the diagram.</td>
</tr>
<tr>
<td>Properties</td>
<td>Display the element properties for the selected element.</td>
</tr>
<tr>
<td>In Diagrams</td>
<td>Open the diagram in which the element is used, or display a list of several diagrams in which the element has been used.</td>
</tr>
<tr>
<td>Delete Link</td>
<td>Delete the Association between the message and the element.</td>
</tr>
</tbody>
</table>

**Note:**

You cannot associate elements and diagrams with Topics or Categories, only with Posts and replies.
3.17.8 Delete an Item

To delete a Category, Topic or Post, right-click on the item in the message thread window. Select Delete Post from the context menu.

A confirmation dialog displays. Click on the OK button.
### 3.17.9 Context Menu

The **Discussion Forum** context menu enables you to access the following functions.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reply</td>
<td>Open the <strong>Message Dialog</strong> to create a <strong>reply</strong> to the selected post.</td>
</tr>
<tr>
<td>New Topic</td>
<td>Add a <strong>New Topic</strong> or <strong>New Post</strong> to the forum.</td>
</tr>
<tr>
<td>New Post</td>
<td></td>
</tr>
<tr>
<td>Edit</td>
<td><strong>Edit an Item</strong></td>
</tr>
<tr>
<td>Copy Path to Clipboard</td>
<td><strong>Copy the path</strong> of the currently-selected post to the clipboard.</td>
</tr>
<tr>
<td>Refresh Category ‘xyz’</td>
<td>Refresh the named category, topic or post, getting new posts and topics.</td>
</tr>
<tr>
<td>Refresh Topic ‘xyz’</td>
<td></td>
</tr>
<tr>
<td>Refresh Post ‘xyz’</td>
<td></td>
</tr>
<tr>
<td>Reload Discussion Forum</td>
<td>Reload the entire Discussion Forum, getting new posts and topics.</td>
</tr>
<tr>
<td>Mark branch as read</td>
<td>Mark this topic and all sub-topics and posts as read.</td>
</tr>
<tr>
<td>Mark branch unread</td>
<td>Mark this topic and all sub-topics and posts as unread.</td>
</tr>
<tr>
<td>Mark ‘xyz’ unread</td>
<td>Mark this topic as unread.</td>
</tr>
<tr>
<td>Forum Connections...</td>
<td>Access <strong>other discussion forums</strong> from other Enterprise Architect models or models located on servers.</td>
</tr>
<tr>
<td>Forum Options...</td>
<td>Change the <strong>loading behavior</strong> of the forum.</td>
</tr>
<tr>
<td>Delete Category &lt;xyz&gt;</td>
<td>Delete this category, topic or post and all sub-topics and sub-posts.</td>
</tr>
<tr>
<td>Delete Topic &lt;xyz&gt;</td>
<td></td>
</tr>
<tr>
<td>Delete Post &lt;xyz&gt;</td>
<td></td>
</tr>
</tbody>
</table>
3.17.10 Copy Path to Clipboard

To copy the current path in the discussion forum tree to the clipboard, right-click on any element in the tree and select the Copy Path To Clipboard context menu option. Alternatively, press [Ctrl]+[C].

The clipboard now contains the path to the selected item in the tree. In the above example, the clipboard contains Project::Project Stages::Stage 1 Requirements.
3.17.11 Forum Options

The Forum Options dialog enables you to change the loading behavior of the forum.

From here you can:

- **Clean History** - Clears the history log, which keeps track of which posts you have read
- **Load forum data when required** - This is the fastest loading option. Forum data is only loaded on demand; for example, when you read a post
- **Preload all forum data** - This caches the entire contents of the forum on load. This takes longer to load but, once completed, maneuvering the forum is faster.
3.17.12 Forum Connections

Forum Connections enables you to access other discussion forums from other Enterprise Architect models, or models located on servers.

To switch to another discussion forum, follow the steps below:

1. Right-click on an item in the tree and select the Forum Connections context menu option. The Forum Connections dialog displays.

2. Click on the New button and select the Enterprise Architect model in which to access the discussion form. Click on the Open button; the model displays in the Forum Connections panel.

3. Select the check box against the model in the Forum Connections panel.

4. Click on the Open Selected Forum button. The connection now switches to the forum in the selected model from the list.

5. Click on the OK button. The Discussion Forum now shows the discussion in the selected forum.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Name</td>
<td>The connection name becomes the name of the model you selected.</td>
</tr>
<tr>
<td>Connection Type</td>
<td>The type of Enterprise Architect model: a local .EAP file or a model on a remote server.</td>
</tr>
<tr>
<td>Target Model</td>
<td>The path to the selected model.</td>
</tr>
<tr>
<td>New</td>
<td>Creates a new Discussion Forum connection.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves the connection to the Forum Connections list.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the currently selected connection from the Forum Connections list.</td>
</tr>
<tr>
<td>Forum Connections</td>
<td>Lists all forum connections created.</td>
</tr>
<tr>
<td>Prompt for connection</td>
<td>If you select this checkbox, each time you select the Discussion Forum option from the main menu, the Forum Connections dialog displays.</td>
</tr>
<tr>
<td>Open Selected Forum</td>
<td>Switches the Discussion Forum to the one selected in the Forum Connections list.</td>
</tr>
</tbody>
</table>
3.18 Spell Checking

Enterprise Architect provides a powerful spell checking facility. This operates at the project level and enables you to quickly spell check an entire project.

See Also
- Using the Spell Checker
- Correcting Words
- Select Language
3.18.1 Using the Spell Checker

Enterprise Architect has an inbuilt spell checker.

Notes:

- Enterprise Architect 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.
- In the Corporate edition of Enterprise Architect, if security is enabled you must have Spell Check permission to spell check a package and set the spell check language.

To perform a spell check, follow the steps below:

1. Select the Tools | Spell Check Project or Tools | Spell Check Current Package menu option, depending on which level of spell check you require. The Spell Check dialog displays.

2. Select the checkbox against each of the items to spell check within your model.

3. Click on the Start button to begin the spell check.

4. As the spell check proceeds, the text being checked displays in the visible edit area. If an error is detected, the Check Spelling dialog displays, offering several options to correct the error.
<table>
<thead>
<tr>
<th>Not in Dictionary:</th>
<th>Ignore</th>
<th>Ignore All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suggestions:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spar</td>
<td>Change</td>
<td></td>
</tr>
<tr>
<td>Spars</td>
<td></td>
<td>Change All</td>
</tr>
<tr>
<td>Spear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pari</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add words to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>userdict.lx</td>
<td>Add</td>
<td>Suggest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options...</td>
<td></td>
<td>Help</td>
</tr>
<tr>
<td>Dictionaries...</td>
<td></td>
<td>Cancel</td>
</tr>
<tr>
<td>Undo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.18.2 Correcting Words

As the spell check progresses, Enterprise Architect highlights any errors or unknown words in the Check Spelling dialog. This enables you to correct the spelling of a word, ignore the error, add the word to a user dictionary, suggest alternatives or otherwise assist in the spelling correction process.

The inbuilt spell check stores user-defined words in the User Dictionary (userdict.tlx) stored in the Enterprise Architect installation directory. During the spell check process, if you add a word, it is written into this file for later reference.

To correct the current word you can:

- Modify the spelling by hand and click on the Change or Change All button to change the word to that spelling
- Click on a suggested alternative and click on the Change or Change All button to change the word to that spelling
- Click on the Ignore or Ignore All button to exclude the word from the spell check
- Click on the Add button to add the word to the current user dictionary
- Click on the Suggest button to list alternative spellings or words
- Click on the Cancel button to abort the spell check entirely.
3.18.3 Select a Different Language

Enterprise Architect is supplied with two dictionaries, for US English and British English. Additional dictionaries are available for download from the registered area of the Sparx Systems website. Once these have been downloaded and installed, you can select another language in which to perform the spell check.

**Note:**

Before selecting a language as described below, ensure you have downloaded the additional language pack and installed it in the Enterprise Architect install directory. Language packs are available from: [http://www.sparxsystems.com/registered/reg_ea_down.html](http://www.sparxsystems.com/registered/reg_ea_down.html).

To select another language for the spell checker, follow the steps below:

1. Select the **Tools | Spelling Language** menu option. The Spell Check Language dialog displays.

   ![Select Language](image)

2. Click on the radio button for the required language dictionary to use.

3. Click on the **OK** button. The selected language remains the current language until changed.
4 UML Tool Project Roles

Enterprise Architect performs a number of tasks that are suited to a variety of professions. The way you use Enterprise Architect can depend on your role within a project. This topic describes some common working practices with Enterprise Architect for a range of project roles. There are tools for the roles of:

- Business Analyst
- Software Developer
- Software Architect
- Software Engineer
- Project Manager
- Tester
- Database Administrator
- Implementation Manager
- Technology Developer

Click on the appropriate role title to explore how Enterprise Architect can assist you in carrying out your role within a model driven project.
4.1 Business Analysts

A Business Analyst can use Enterprise Architect to create high-level models of business processes. These include business requirements, activities, work flow, and the display of system behavior. Using Enterprise Architect, 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 2.1.1 Activity diagrams and are less formal than other diagram types, but they provide a useful means for expressing essential business characteristics and requirements.

**Model Requirements**

Gathering requirements is typically the first step in developing a solution, be it for developing a software application or for detailing a business process. It is an important step in the implementation of a project. Enterprise Architect enables you to define the Requirement elements, connect Requirements to the model elements for implementation, connect Requirements together into a hierarchy, report on Requirements, and move Requirements 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 can use an Interaction Overview diagram, which provides an overview of sub activities that are involved in a system.
Display System Behavior

In displaying the behavior of a system as a Use Case diagram, Enterprise Architect gives the Business Analyst an easily understood tool for mapping the functional requirements and behavior of a system.
4.2 Software Architects

Software Architects can use Enterprise Architect to map functional requirements with Use Cases, perform real-time modeling of objects using Interaction diagrams (Sequence, Timing, Communication or Interaction Overview), 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 modeled 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

Component diagrams enable 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 can be brought in from existing projects and from third-party vendors.

See Also

- Analysis Diagrams
- Modeling with Enterprise Architect
- XMI Import and Export
- XML Technologies
4.3 Software Engineers

Software Engineers using Enterprise Architect can map Use Cases onto 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
In Enterprise Architect the Software Engineer can study the Use Cases developed by the Software Architect, and with that information create Classes that fulfill the objectives defined in the Use Cases. A Class is one of the standard UML constructs that is used to detail the pattern from which objects are produced at runtime. To record the relationships between Use Cases and Classes, the Software Engineer can create diagrams linking the elements with Realization connectors, and/or map the Realization connectors in the Relationship Matrix.

Detail Interaction Between Classes
Interaction diagrams (Sequence and Communication diagrams) enable the Software Engineer to model the dynamic design of the system. Sequence diagrams are used to detail the messages 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 can 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
The Software Engineer can use Package diagrams to detail the software architecture. Package diagrams are used to organize diagrams and elements into manageable groups, declaring the dependencies.

See Also
- Modeling with Enterprise Architect
4.4 Developers

Developers can use Enterprise Architect to perform round trip code engineering, which includes reverse engineering of existing code and generation of code from UML Class elements.

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 software from existing source code to UML 2.1 models 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

Enterprise Architect enables 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 the ability to reverse engineer code, Enterprise Architect offers the developer the option of forward engineering code (code generation). This enables the developer to make changes to their model with Enterprise Architect and have these changes implemented in the source code.

Determine the System State

To visualize the state of the system the developer can use State Machine diagrams to describe how elements move between states, classifying their 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 can 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 diagrams and elements into manageable groups, and to declare their dependencies.

**Follow the Flow of Code**

Activity diagrams are used to enable a better understanding of the flow of code. Activity diagrams illustrate the dynamic nature of the system. This enables modeling of the flow of control between Activities and represents the changes in state of the system.

**See Also**
- XML Technologies
- MDA Transformations
- Debug and Profile
4.5 Project Managers

Enterprise Architect provides support for the management of projects. Project Managers can use Enterprise Architect to assign resources to elements, measure risk and effort, and estimate project sizes. Enterprise Architect also helps them manage element status, change control and maintenance.

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 enables the Project Manager or Development Manager to assign resources directly to model elements and track progress over time.

Risk Management

The Project Management window can be used to assign Risk to an element within a project. The Risk Types enable the Project Manager to name the risk, define the type of risk, and give it a weighting.

Maintenance

Enterprise Architect enables the Project Manager to track and assign maintenance-related items to elements within Enterprise Architect. This enables rapid capture and record keeping for items such as issues, changes, defects and tasks. They can also create and maintain a project Glossary of processes, procedures, terms and descriptions.
4.6 Testers

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

You can assign test cases to individual model elements. requirements and constraints. You can add scenarios to model elements, and use element defects to report problems associated with model elements.

For more detailed information on testing, see the Introduction to Testing in Enterprise Architect topic.

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, Enterprise Architect enables the tester 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 can be Pre-conditions (states which must be true before an event is processed), Post Conditions (events that 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
Enterprise Architect provides the facility to generate high quality test documentation. Enterprise Architect produces test documentation in the industry-standard .RTF file format.

Element Defect Changes
Defect tracking enables you to allocate defect reports to any element within the Enterprise Architect model. This enables all who are involved in the project to quickly view the status of defects, to see which defects have to be addressed and which have been dealt with.
4.7 Implementation Manager

Using Deployment diagrams in Enterprise Architect, you can model the tasks involved in the rollout of a project, including network deployment and workstation deployment. Users involved in project deployment can add maintenance tasks to the diagram elements.

Deployment diagrams provide a static view of the run-time configuration of nodes on the network or of workstations, and the components that run on the nodes or are used in the workstations.

Maintenance

Enterprise Architect enables you to track and assign maintenance-related items to elements within Enterprise Architect. This enables you to rapidly capture and keep records of maintenance tasks such as issues, changes, defects and tasks. By providing a centralized facility for each element involved in the deployment process Enterprise Architect offers a powerful solution for tracing the maintenance of the items and processes involved in system deployment.
4.8 Technology Developers

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

The following illustration shows a customized view created using an Add-In.

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 can 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 can 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 Enterprise Architect. This enables the generation of code languages not specifically supported by Enterprise Architect and enables you to define the way Enterprise Architect generates source code to comply with your own company style guidelines.

Tagged Value Types

Tagged Values are used in Enterprise Architect 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 can be used to create a logical collection of resources that can contain UML Profiles, Patterns, Code Templates, Image files and Tagged Value types that can be accessed from a single point in the Resources window.

Enterprise Architect Add-Ins

Enterprise Architect Add-Ins enable you to build your own functionality into Enterprise Architect, creating your own mini programs that can extend the capabilities of Enterprise Architect, defining your own menus, and creating your own Custom Views.

The Enterprise Architect Software Developers' Kit

This topic gives a brief introduction to many of the features identified above, explaining how they are used within Enterprise Architect. However, SDK for Enterprise Architect explains in greater detail how to develop and integrate these facilities.
4.9 Database Administrators

Enterprise Architect supports a range of features for the development of databases, including modeling database structures, importing database structures from an existing database and generating DDL for rapidly creating 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. Enterprise Architect currently supports JET-based databases; DB2; InterBase; Informix; Ingres; MySQL; SQL Server; PostgreSQL; Sybase Adaptive Server Anywhere and Adaptive Server Enterprise; and Oracle 9i, 10g and 11g.

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 enables the DBA to quickly document their work and create a diagrammatic account of a complex database through the graphical benefits of UML.

See Also
- Set Up a Database Repository
Part V
5 What Is UML Modeling?

In relation to using Enterprise Architect, UML modeling can be described as graphically representing a business process or software system. The resulting model can be used to emphasize a certain aspect of the system being represented and to record, document and communicate its detail. A study of such a model can enable insight or understanding of the system.

The Enterprise Architect Modeling Platform

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. For information on UML, see the UML Dictionary topic.

For examples of the UML models that Enterprise Architect can help you build, see the Model Templates topic.

Building a Model

Using Enterprise Architect, you can quickly build a model using a hierarchy of packages to represent the structure and organization of the model. Each package can contain:

- Other packages
- Diagrams that represent various aspects of the equipment, environment and business processes of the system
- Elements that represent the objects and actions within the system or process, arranged in an organization defined by relationships represented by UML connectors.

The Create a Project - Quick Start topic briefly shows you how to create a diagram within a package, containing elements and connectors. Sparx Systems also provide a demonstration of quickly developing a Use Case model.

For specific details of configuring and combining the components of a model, see:

- Work With Packages
- Work With Diagrams
- Work With Elements
- Work With Connectors

Requirements Management

Gathering requirements is typically the first step in developing a solution, be it for developing a software application or for detailing a business process. Requirements are essentially 'what the system must do'. The requirements management built into Enterprise Architect provides full support for defining, organizing and managing the requirements that drive the project.

Relationship Matrix

The Relationship Matrix enables you to display and manage the relationships between the elements within selected packages. You can refine the display to show specific types of relationship between specific types of element. The Relationship Matrix is an effective and convenient method of visualizing relationships quickly and definitively.

UML Stereotypes

Stereotypes are an inbuilt mechanism for logically extending or altering the meaning, display and syntax of a model element. Different model elements have different standard stereotypes associated with them. You can also define your own stereotypes.
For further information on stereotypes, see the UML Stereotypes topic.

**UML Profiles**

UML Profiles are a means of extending UML, which enables you to build models in particular domains. A Profile is a collection of additional stereotypes and Tagged Values applied to elements, attributes, methods and connectors, which together describe some particular modeling problem and facilitate modeling constructs in that domain.

For further information on Profiles, see the UML Profiles topic.

**UML Patterns**

Patterns are groups of collaborating Objects/Classes that can be abstracted from a general set of modeling scenarios (that is, parameterized collaborations). They generally describe how to solve an abstract problem, and are an excellent means of achieving re-use and building in robustness.

For more information on Patterns, see the UML Patterns topic.

**MDG Technologies**

The Model Driven Generation (MDG) Technologies enable you to access and use the resources of a specific technology within Enterprise Architect. Interfaces to some technologies, such as BPMN and Iconix Process, are integrated with Enterprise Architect, whilst interfaces to others such as Eclipse and Visual Studio can be added separately. You can also link to technologies that you have created yourself.

For more information on MDG Technologies, see the MDG Technologies topic.

**Business Modeling**

Modeling the business process is an essential part of any software development process. It enables you to establish the broad outline and procedures that govern what it is a business does. As the Business Process Model typically has a broader range than just the software system being considered, it also enables you to clearly map what is in the scope of the proposed system and what is to be implemented in other ways.
5.1 Work With Packages

A Package is a container of model elements, and is displayed in the Project Browser using the ‘folder’ icon familiar to Windows users. This topic 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

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have Update Element permission to update or delete a package.
5.1.1 **Open Package in the Project Browser**

To open a package from the Project Browser, follow the steps below:

1. Double-click on a package; the contents display in the Project Browser.
2. Click on the + and - symbols next to the folder icon to open or close the package respectively.

**Tip:**

Package contents are arranged alphabetically and elements can be dragged from one package to another using the mouse.
5.1.2 Add a Package

To add a new package:

1. In the Project Browser, select the package or view under which to add a new package.
2. Right-click on the folder icon within the Project Browser. The context menu displays.
3. Select the Add | Add Package menu option. The New Model Package dialog displays.

4. In the Package Name field type the name of the new package.
5. To immediately create a diagram for the package, leave the Automatically add new diagram checkbox selected. To avoid creating a diagram, deselect the checkbox.
6. If you are adding a package to a parent package that is under version control, the Add to Version Control option displays, with the checkbox selected. Deselect the checkbox to exclude the new package from version control, otherwise leave it selected.
7. Click on the OK button. The new package is inserted into the tree at the current location and, if you left the Automatically add new diagram checkbox selected, the New Diagram dialog displays.
8. If you have selected to put the package under version control, the Package Control Options dialog displays. Complete this dialog as required.

Tip:
You can also add a package using the Enterprise Architect UML Toolbox and pasting a new package element into a diagram. In this case the package is created under the diagram's owning package, and is created with a default diagram of the same type as that in which the package is created.

Note:
In a multi-user environment, other users do not see the change until they reload their project.
5.1.3 Rename a Package

To rename a package:

1. Select the package to rename in the Project Browser.
2. Right-click to display the context menu.
3. Click on the Package Properties option.
4. In the Name field, type the new name.
5. Click on the OK button.

Alternatively, highlight the package to rename, and press [F2].

Note:
In a multi-user environment, other users do not see the change until they reload their project.
5.1.4 Drag a Package Onto a Diagram

You can drag a package element from the Project Browser onto the current diagram. This displays the package and any contents within. This is a useful feature to help organize the display and documentation of models.

The following illustration shows how a package is displayed in a diagram; note the child Actor and Use Case icons.

See Also
- Show or Hide Package Contents
5.1.5 **Show or Hide Package Contents**

To show or hide the contents of packages in a diagram, follow the steps below:

1. Load a diagram.
2. Double-click in the background area to open the **Diagram Properties** dialog.
3. Click on the **Elements** tab.

4. Select or clear the **Package Contents** checkbox as required.
5. Click on the **OK** button.
5.1.6 Delete a Package

To delete a package, follow the steps below:

1. Highlight the package in the Project Browser.
2. Right-click to open the context menu.
3. Click on the Delete option. A confirmation prompt displays.
4. Click on the OK button.

**Warning:**

Deleting a package also deletes all contents of the package, 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 do not see the change until they reload their project.
5.2 Work With Diagrams

Diagrams are collections of project elements laid out and inter-connected as required.

Enterprise Architect supports all of the 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 can have a parent object (optional). Diagrams can be moved from package to package.

The basic elements used in each type of diagram are shown below. After you have looked at these illustrations, go to the following topics:

- Diagram Context Menu
- Diagram Tasks

Tip:
If the diagram display is too small to read comfortably, click on the diagram, press and hold [Ctrl] and use the mouse wheel to temporarily expand or reduce the display magnification.
5.2.1 Diagram Context Menu

Open the required diagram and right-click on the diagram background to open the diagram context menu. Not all menu options shown below appear on all diagram context menus.

- **Add-Ins**
- **Properties...**
- **Swimlanes and Matrix...**
- **Lock Diagram**
  - **New Element or Connector**
  - **Paste Object(s) as Link** (Shift+Insert)
  - **Paste Object(s) as New** (Ctrl+Shift+V)
  - **Clear the Clipboard**
  - **Paste Image from Clipboard** (Ctrl+Shift+Insert)
- **Make All Elements Selectable**
- **Save as Profile...**
- **Import from source file(s)**
- **Import DB schema from ODBC**
- **Save Current Changes** (Ctrl+S)
- **Show as Element List**
- **Help...**

The diagram context menu enables you to:

- View the **Diagram Properties** dialog
- Add **Swimlanes** or a **Swimlanes Matrix** to the diagram
- Protect a diagram from inadvertent changes (**Lock Diagram**)

**Note:**
This does not apply in the Corporate edition if security is enabled. In that case, see the **Lock Model Elements** topic.

- Insert various elements into a diagram (see below)
- **Paste copied element** as a link or as new elements
- **Paste an image** held on the clipboard into the diagram
- Make all the elements on the diagram selectable. If an element is selectable, you can move it around the diagram and perform right-click operations. If an element is unselectable, you cannot move it around the diagram and the only right-click operation available is to make the element selectable. This option has no effect on double-click operations on the element, such as displaying child diagrams.
- Save the current diagram as a **Profile** in the **Resources** window
- **Import, or reverse engineer, source code** (not available in the Desktop edition)
- **Import database tables from an ODBC data source** (not available in the Desktop edition)
- Save any changes to the current diagram
- Display the diagram contents as an **Element List** instead of as a diagram
- View the Enterprise Architect Help on the type of diagram currently displayed.
Insert Items

When you click on the **New Element or Connector** context menu option, a list of elements and connectors displays, as shown below for a Timing diagram:

The structure of this list is as follows:

- **Other** - expands to offer options to select elements and connectors from diagram types other than either the current diagram type or pinned Enterprise Architect Toolbox pages
- The expanded list of elements and connectors for the current diagram type
- Collapsed lists of elements and connectors for pages that have been pinned in the Toolbox; if an MDG Technology:
  - is active
  - automatically pins Toolbox pages, and
  - has pages that redefine UML or Extended pages
  the MDG Technology pages override the UML or Extended pages, which are not shown
- (At the end) **Common** - expands to display a list of the common elements and connectors.
5.2.2 Diagram Tasks

This topic details many of the common tasks associated with managing diagrams.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Update Element** permission to update or delete items on a diagram, and **Manage Diagram** permission to create, copy or delete diagrams.

- Add New Diagrams
- Delete a Diagram
- Rename a Diagram
- Copy And Paste Diagram Element
- Diagram Navigation Hotkeys
- Convert Linked Element to Local Copy
- Z Order Elements
- Copy Image to Disk
- Copy Image to Clipboard
- Change Diagram Type
- Open a Package
- Duplicate a Diagram
- Feature Visibility
- Insert Diagram Properties Note
- Autosize Elements
- Drop Elements from the Project Browser
- Paste from the Project Browser
- Place Related Elements on Current Diagram
- Swimlanes
- Using the Image Manager
- Show Realized Interfaces for a Class
- Label Menu Section
- Pan and Zoom a Diagram
- View Last and Next Diagram
- Set Diagram Page Size
- Scale Image to Page Size
- Lock Diagram
- Manage Legend Elements
- Lay Out a Diagram
- Set Diagram Properties
- Undo Last Action
- Redo Last Action

5.2.2.1 Add New Diagrams

This topic explains how to add a UML diagram, Extended diagram or MDG Technology diagram to a model in Enterprise Architect.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Manage Diagram** permission to create new diagrams.

To add a new diagram to an existing package or element, follow the steps below:

1. In the Project Browser, select the appropriate package or element under which to place the diagram.
2. Do one of the following:
   - In the Project Browser toolbar click on the New Diagram icon
What Is UML Modeling?  | Work With Diagrams

- Right-click to open the context menu and select the Add | Add Diagram or Add | Add <type> Diagram menu option.
- Press [Insert] and select the Add | Add Diagram or Add | Add <type> Diagram menu option, or
- Select the Project | Add Diagram menu option.

The New Diagram dialog displays.

3. The Name field defaults to the name of the selected package or element; if necessary, type a different name for the new diagram.
4. In the Select From panel, click on the appropriate diagram category for the diagram. The Diagram Types panel displays a list of the diagram types within the selected category.
5. In the Diagram Types panel, click on the type of diagram to create.
6. Click on the OK button to create your new diagram.

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 (e.g. a Sequence diagram under a Use Case).

5.2.2.2 Lay Out a Diagram

Enterprise Architect provides the facility to layout diagrams automatically. This creates a tree-based structure from the diagram elements and relationships in a diagram. Owing to the complexity of many diagrams, you might then have to do some manual 'tweaking'.

Notes:
- This facility is available for Structural diagrams and Extended diagrams, but not for Behavioral diagrams (see the UML Diagrams topic for a description of the diagram types). However, the facility is also available for Sequence diagrams generated by the Enterprise Architect Debugger.
- Dynamic and Analysis diagrams are NOT suited to this form of layout - please ensure first that the diagram type you are laying out benefits from the action.
- If you dislike the autolayout, you can reverse it before saving the diagram. Click [Ctrl]+[Z].
Layout a Diagram
To layout a diagram, follow the steps below:

1. Select a diagram.
2. Click on either:
   - The Diagram | Layout Diagram option, or
   - The Auto Layout button on the diagram toolbar.

Access 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. Double-click on the background of the diagram to display the Diagram Properties dialog.
2. Click on the Diagram tab, then click on the Set Layout Style button. The Diagram Layout Options dialog displays.
3. When you have made the required changes, click on the OK button to save the changes.

You can alter any of the following settings on the Diagram Layout Options dialog to refine your layout:

- **Cycle Remove Options**
  - Greedy
  - Depth First Search

- **Crossing Reduction Options**
  - Iterations
  - Aggressive

- **Layering Options**
  - Longest Path Sink
  - Longest Path Source
  - Optimal Link Length

- **Initialize Options**
  - Naive
  - Depth First Search Outward
  - Depth First Search Inward

- **Set as Project Default**

- **Spacing**
  - Layer Spacing
  - Column Spacing

- **Direction**
  - Up
  - Left
  - Down
  - Right

You can alter any of the following settings on the Diagram Layout Options dialog to refine your layout:

- **Cycle Remove Options** panel - these settings remove cycles in the element organization (where element X is the source of a path but also becomes the target of a branch of the path), by reversing the connectors that impose the cycling and then reorganizing the diagram and reinstating the reversed relationships. This identifies the primary source element in the diagram.
- **Greedy** - Select to use the Greedy Cycle Removal algorithm, which minimizes the number of connectors reversed.
- **Depth First Search** - Select to use the Depth First Search Cycle Removal algorithm, which establishes the longest linear sequence possible, before establishing parallel sequences and branches. This algorithm is less effective in large and/or complex diagrams, but produces a more natural layout than the Greedy algorithm.

- **Crossing Reduction Options** panel - these options determine how long the routine should look for ways of reorganizing the layout to avoid crossed relationships:
  - **Iterations** - Type the number of iterations to be used during cycle removal (more than 8 does not usually provide any improvement).
  - **Aggressive** - Select to use an aggressive (detailed and time-consuming) crossing reduction step.

- **Layering Options** panel - these settings determine how elements are organized in layers during layout:
  - **Longest Path Sink** - Select to use the Longest Path Sink Layering algorithm, where the final target
elements (sinks, which have no relationships issuing from them) are arranged in a layer at the top of the diagram, and the relationship paths built downwards from there in as many layers as there are nodes in the longest path.

- **Longest Path Source** - Select to use the *Longest Path Source Layering* algorithm, where the original source elements (those with no relationships entering them) are arranged in a layer at the bottom of the diagram and the relationship paths built up from there in as many layers as there are nodes in the longest path.

- **Optimal Link Length** - Select to use the *Optimal Link Length Layering* algorithm, which organizes the elements into whichever layers minimize the total source-to-sink relationship chain; in this layout you can have both source elements and sink elements at various levels of the diagram.

**Layout Options** panel

- **Layer Spacing** - Type the default number of logical units between layers of elements (vertical spacing).
- **Column Spacing** - Type the default number of logical units between elements within a layer (horizontal spacing).
- **Up, Down, Left, Right** - Select the direction in which directed connectors should point, to set the position of the primary source element and the overall flow of the diagram.

**Initialize Options** panel - the autolayout routine inserts line waypoints and connectors into relationship paths to help plot the direction of relationships. The routine then assigns an index number to every node, such that nodes in the same layer are numbered left to right. The settings in this panel determine how those index numbers are assigned.

- **Naive** - Select to use the *Naive Initialize Indices* algorithm, which assigns index numbers to nodes as they are encountered in a sweep and tends to place all waypoints to the right of real nodes (and therefore long relationships between a small number of elements to the right of chains of short relationships between several elements).

- **Depth First Search Outward** - Select to use the *Depth First Out Initialize Indices* algorithm, which assigns index numbers to nodes as they are encountered in a depth first search from source nodes outwards (and would therefore place longer relationship chains to the left of shorter chains, with the primary source node at the start of the diagram flow).

- **Depth First Search Inward** - Select to use the *Depth First In Initialize Indices* algorithm, which also assigns index numbers to nodes as they are encountered in a depth first search, but from sink nodes inwards (and would therefore place longer relationship chains to the left of shorter chains, with the ultimate target node at the end of the diagram flow).

**Set as Project Default** checkbox

- Select this checkbox to apply the diagram layout settings to all diagrams in the project. If you later check this box and click on the **OK** button for a different diagram, the new settings override the settings saved earlier.

The following is an example of an automatically laid out diagram, with the following options set:

- **Depth First Search**
- **Optimal Link Length**
- **Depth First Search Outward**
- **Direction - Up.**
5.2.2.3 Delete Diagram

**Warning:**
In Enterprise Architect there is no *Undo* feature for deleting diagrams, so be certain that you want to delete a diagram before you do so.

**Note:**
When you delete a diagram, you do not delete the elements in the diagram from the *model*.

To delete a diagram from your model, follow the steps below:

1. In the *Project Browser*, right-click on the diagram to delete. The context menu displays.
2. Select the *Delete `<diagram name>`* menu option. A confirmation prompt displays.
3. Click on the *OK* button to confirm the delete.
5.2.2.4 Rename Diagram
To rename a diagram, follow the steps below:
1. Open the Diagram Properties dialog by double-clicking on the diagram background, or by selecting the Diagram | Properties menu option.
2. In the Name field on the General tab, type the new name for your diagram.
3. Click on the OK button to save changes.

5.2.2.5 Copy And Paste 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 the Copy menu option. Alternatively, press [Ctrl]+[C].
3. For single elements, select the Edit | Copy menu option or alternatively press [Ctrl]+[C].

Paste Diagram Elements
To paste diagram elements, follow the steps below:
1. Open the diagram to paste into.
2. Right-click on the diagram background to open the diagram context menu.
3. Select either the Paste Object(s) as New menu option (completely new element) or the Paste Object(s) as Link menu option (reference to the existing element).

5.2.2.6 Diagram Navigation Hotkeys
The diagram hotkeys enable you to quickly navigate to and select elements within a diagram. The following table details the key combinations and their functionality.

<table>
<thead>
<tr>
<th>Hotkey Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Arrow], Element(s) selected</td>
<td>Move the selected element(s).</td>
</tr>
<tr>
<td>[Arrow], No element selected</td>
<td>Scroll around the diagram.</td>
</tr>
<tr>
<td>[Esc]</td>
<td>Clears the current selection.</td>
</tr>
<tr>
<td>[Tab]</td>
<td>Selects the first element in the diagram if none currently selected.</td>
</tr>
<tr>
<td>[Shift]+click</td>
<td>Adds the clicked element to the current selection.</td>
</tr>
<tr>
<td>[Ctrl]+click</td>
<td>Adds the clicked element to the current selection.</td>
</tr>
<tr>
<td>[Ctrl]+[Shift]+drag</td>
<td>Pans the diagram.</td>
</tr>
<tr>
<td>[Alt]+[G]</td>
<td>Selects the item in the Project Browser and gives it focus.</td>
</tr>
</tbody>
</table>

5.2.2.7 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.
To copy a diagram image to file, follow the steps below:
1. Open the diagram to save.
2. Select the Diagram | Save Image menu option, or press [Ctrl]+[T].
3. When prompted, enter a name for the file and select an image format.
4. Click on the OK button.

**Note:**

Enterprise Architect clips the image size to the smallest bounding rectangle that encompasses all diagram elements.

### 5.2.2.8 Copy Image to Clipboard

You can copy diagram images onto the MS Windows clipboard and paste them directly into MS Word or other applications.

To copy an image to the clipboard, follow the steps below:

1. Open the diagram to copy.
2. Select the **Diagram | Copy Image** menu option, or press `[Ctrl]+[B]`.
3. Click on the OK button.

The diagram has been copied to the clipboard and can now be pasted into compatible applications or into another diagram. You can set the clipboard format on the **Options** dialog ([Tools | Options menu option, General page]). Enterprise Architect supports bitmap or metafile format.

### 5.2.2.9 Change Diagram Type

If necessary, you can change one type of diagram to another type. 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.

**Note:**

Some diagram types do not transfer to others; for example you cannot change a Class diagram into a Sequence diagram.

To change a diagram type, follow the steps below:

1. Open the diagram to change.
2. Select the **Diagram | Change Type** menu option. The **Change Diagram Type** dialog displays.

   ![Change Diagram Type Dialog]

   **UML Class Diagrams** capture the logical structure of a system as a series of classes, their features and the relationships between them.

   ![Diagram Types]

3. Select the required diagram type.
4. Click on the **OK** button to save changes.

**5.2.2.10 Z Order Elements**

Z Order refers to an element's depth in the diagram perspective, and thus influences which elements appear in front of others and which appear behind.

To set the Z Order of an element, follow the steps below:

1. Right-click on the element in the **Diagram View**.
2. Select the **Z order** menu option. The following submenu displays:

   ![Z Order Menu](image)

3. Select the operation to perform. The element is moved to the new position in the diagram perspective.

**5.2.2.11 Copy Diagram**

Enterprise Architect makes it easy to duplicate a complete diagram, either with links back to the original diagram elements (**shallow mode**), or with complete copies of all elements in the diagram (**deep mode**).

When you copy a diagram in shallow mode, the elements in the new diagram are linked to the originals, so if you change the properties of one, the other reflects 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.

**Procedure**

To duplicate a diagram, follow the steps below:

1. In the **Project Browser**, select the diagram to copy.
2. Right-click to display the context menu and select the **Copy Diagram** menu option.
3. Navigate to the package to host the new diagram, and right-click to open the context menu.
4. Select the **Paste Diagram** menu option. The **Copy Diagram** dialog displays.

   ![Copy Diagram Dialog](image)

5. In the **Name** field, type the name for the new diagram.
6. In the **Type of copy** panel, click on the radio button for the type of copy you require; either linked elements (shallow copy) or complete copies of the originals (deep copy).
7. Click on the **OK** button.

Enterprise Architect automatically creates the new diagram, links or creates new elements and arranges them as in the original diagram. All links are also copied between diagram elements where appropriate.
5.2.2.12 Open Package From Diagram

To open a package from within a diagram follow the steps below:

1. Open a diagram that shows the package to open.
2. Right-click on the package element to open the context menu.
3. Select the Open Package option. Alternatively, press [Ctrl]+[K].

Note:
Enterprise Architect finds the package default diagram and opens it for you. This is the first available diagram in the package, selected in alphabetical order; for example, a diagram called Alpha in a child package or element several levels down opens before a diagram called Beta immediately under the selected package.

5.2.2.13 Feature Visibility

Enterprise Architect enables you to set the visibility of attributes and operations per Class or per diagram, or on a package diagram. For example, you can hide all protected attributes, all private operations or any other combination of attributes and operations. The visibility you set applies only to the current diagram, so a Class could appear in one diagram with all elements displayed, and in another with elements hidden.

It is possible to show inherited attributes, operations, requirements, constraints and Tagged Values for elements that support those features. When Enterprise Architect 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 on the diagram object to display the context menu and select the Advanced Settings | Use Rectangle Notation option.

Customize Feature Visibility

To customize feature visibility, follow the steps below:

1. Either:
   - Click on the element in the diagram and either click on the Element | Feature Visibility menu option or press [Ctrl]+[Shift]+[Y], or
   - Right-click on the element in the diagram to display the context menu and click on the Feature Visibility option.

   The Feature Visibility dialog displays.
2. Select the checkbox against each feature that should be visible and clear the checkbox against each of those that should not.

3. In the **Show Element Compartments** panel, select the compartments to display for the elements on the diagram.

   The **Fully Qualified Tags** checkbox enables you to display the full provenance of a Tagged Value, where the same Tagged Value can be used several times in different contexts with different values. The description in the Tagged Value compartment reads: `<Profile>::<Stereotype>::<Tagged Valuename>=<Value>`, for example: `BPMN::Activity::Activity Type = Task`. (Only for Tagged Values created in Enterprise Architect release 7.1 or later.)

   If you select the **Notes** checkbox, the Notes compartment on each element in the diagram displays the text that has been typed into the **Notes** field of the **Element Properties** dialog. This checkbox also enables the **maximum chars** field, which defaults to 1000 as the number of characters of notes text displayed. Overtype this value to display less text or more text.

   The change only applies to the selected elements on the diagram, so you can display full notes for a selected element whilst the other elements on the diagram have no notes text.

   **Note:**

   If you have selected the **Notes** checkbox, you can select the **Render Formatted Notes** checkbox to display the text on the diagram, formatted using the [Rich Text Notes](#) toolbar.

4. In the **When Resizing Elements** panel, select the appropriate option for resizing the Class, object or table to prevent very wide diagram objects.

   The selected option defaults to **Resize to longest Feature**, so that the minimum width for a diagram object is determined by its longest displayed attribute, method or other compartment value. If necessary, you can change the option to **Wrap Features** (so that any longer features are wrapped onto multiple lines) or **Truncate Features** (so that longer features are not displayed in full).

5. If required, in the **Inherited Features** panel, select one or both checkboxes to set whether Enterprise Architect should display inherited features as well as directly owned ones.

6. Click on the **OK** button to save changes. Enterprise Architect redraws the diagram with the appropriate level of feature visibility.
Suppress Features in Diagram

Enterprise Architect enables you to set the visibility of attributes and operations - where shown - per diagram for a Class. You can hide attributes and operations by scope, or you can hide attributes and operations individually by clicking on the Custom button. The visibility you set applies only to the current diagram, so a Class can appear in one diagram with all elements displayed, and in another with elements hidden.

### Feature Type

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute1</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute2</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute3</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute4</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute5</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute6</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute7</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute8</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute9</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>attribute10</td>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>

5.2.2.14 Insert Diagram Properties Note

Properties of a diagram can 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. To create the note, click on the button and click on the diagram.

Alternatively, select the Diagram | Property Note menu option.

**Note:**

This is not the same as the diagram details note, which displays in the top left corner of the diagram if the Show Diagram Details checkbox is selected on the Diagram Properties dialog. You cannot move the diagram details, nor change the appearance. To hide the diagram details, deselect the checkbox.
5.2.2.15 Autosize Elements

You can autosize an element or group of elements in a diagram to the minimum size for revealing the information (such as attributes, operations and notes) they display. Each element is reduced or expanded as necessary to just reveal the information displayed. The size change effectively operates around the mid point of each element, so the layout and size of the diagram do not change. To automatically change the layout of a diagram, see the Lay Out a Diagram topic.

To autosize elements, follow the steps below:

1. Select the elements to resize (press [Ctrl]+[A] to select all).
2. Either:
   - Right-click on any of the elements and, on the context menu, select the Autosize menu option, or
   - Press [Alt]+[Z].

Note:
- Not all elements resize; elements such as Events remain the same; Timing and Sequence diagrams (where position is crucial) are unchanged; and elements added from a profile or shape script maintain any size definitions imposed by the profile.
- The minimum size Enterprise Architect can set is the default size for the element, usually around 100 x 100, but different for certain elements.
- With an element image created with a Shape Script that contains a defSize command, Autosize returns the element to the defSize value and not the element default size.

5.2.2.16 Paste from Project Browser

You can paste an element from the Project Browser into the current diagram.

If you press and hold [Ctrl] while you drag an element from the Project Browser onto the current diagram, Enterprise Architect prompts you to select the type of paste action to carry out.

Three options are available:

1. Paste the element as a simple link. In this case the element displays in the current diagram as a simple reference to the original source element. Changes to the element in the diagram affect all other links to this element.
2. Paste as an instance of the element. If the element can have instances such as an Object, Sequence instance or Node instance, you can drop the element in 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 Communication diagram.
3. Create as a child of the source element. This automatically creates a new Class - which you are prompted to name - with a Generalization connector back to the source. This is very useful when you have a Class library or framework from which you inherit new forms; for example, you can paste a Hashtable as "MyHashtable" which automatically becomes a child of the original Hashtable. Used with the Override parent operations and features, this is a quick way to create new structures based on frameworks such as the Java SDK and the .NET SDK.
5.2.2.16.1 Paste Multiple Items

You can paste multiple elements from the Project Browser into the current diagram.

To select multiple elements, click on the selected items from the Project Browser while pressing and holding:

- [Ctrl] to add single items to the selection of multiple elements, or
- [Shift] to select all the elements between the first and last selected items in the Project Browser.

You can then drag the selected elements from the Project Browser onto the current diagram, pressing and holding [Ctrl]; Enterprise Architect prompts 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 displays in the current diagram as a simple reference back to the original source element. Changes to the element in the diagram affect all other links to this element.

2. Paste as an instance of the element. If the element can have a classifier (such as an Object, Sequence instance or Node instance) 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 automatically creates a new Class (which Enterprise Architect prompts you to name) and creates a Generalization connector back to the source. This is very useful when you have a Class library or framework from which you inherit new forms (e.g. a Hashtable can 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:**

To make use of the *Only show this dialog when [Ctrl]+Mouse drag is used* checkbox, you must first select the *Auto Instance* checkbox on the Diagram Behavior page of the Options dialog (select the Tools | Options | Diagram | Behavior option).

## 5.2.2.16.2 Paste Composite Elements

Several additional options are available to you when pasting Composite elements from one diagram to another.

When you drag a Composite element from the Project Browser onto the current diagram with [Ctrl] held down, Enterprise Architect prompts you to select the type of paste action to carry out with the Composite element.

![Paste Element into Diagram](image)

Two advanced options are available for pasting Composite elements; these require the *include Embedded Elements* checkbox to be selected:

1. The *All Embedded Elements* option, which pastes all of the Composite element's embedded elements.
2. The *Based on instance* option, which pastes only the elements contained in a specific instance of the Composite element. Click on the drop-down arrow and select the appropriate instance.

## 5.2.2.16.3 Paste Activities

You can paste an Activity from the Project Browser into the current diagram.

When you hold [Ctrl] down and drag an Activity from the Project Browser onto the current diagram, Enterprise Architect prompts you to select the type of paste action to carry out.
Two options are available:

- Paste the Activity as a link: in this case the Activity appears in the current diagram as a simple reference to the original source Activity. Changes to the Activity in the diagram affect all other links to this Activity.
- Paste as an invocation of the Activity.

**Note:**
To make use of the **Only show this dialog when [Ctrl]+Mouse drag is used** checkbox, you must first select the **Auto Instance** checkbox on the **Diagram Behavior** page of the Options dialog (select the **Tools | Options | Diagram | Behavior** option).

### 5.2.17 Drop Elements from 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 can be dropped either as 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 displays `<InstanceName>:<ElementName>`.

If you are working on an instance diagram, such as a Communication diagram, you can quickly drag and drop Classes and elements from the **Project Browser** onto a diagram as an instance of the classifier. If you are working in a Class diagram, you might drop links to the classifier itself. There are a couple of settings available to simplify this process.

**Note:**
Enterprise Architect displays a warning about this behavior (see below), indicating what is happening. Select the **Hide this message in future** checkbox to prevent this message displaying again.
There are two important things to remember here:

- Hold down [Ctrl] while dragging and dropping an element into a diagram (see Paste from the Project Browser).
- Hold down either [Ctrl] or [Shift] to select multiple elements into a diagram (See Paste Multiple Items from the Project Browser).

To change the default behavior, select the Diagram Behavior page of the Options dialog (select the Tools | Options | Diagram | Behavior option). To enable or disable Auto Instance, select or clear the Auto Instance checkbox. Remember: pressing [Ctrl] inverts the current default.

**See Also**
- Implementation
- Create Object From Attribute
- Make Linked Element A Local Copy

### 5.2.2.18 Place Related Elements on Diagram

To find and place related elements on the current diagram, use the Relationships window (View | More Windows | Relationships).

Right-click on any connector in the list to open the context menu.
If an element is not present in the current diagram, the context menu contains 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 cursor to the required position in the diagram and click to place the element. Alternatively, press `[Esc]` to cancel the action.

### 5.2.2.19 Swimlanes

Enterprise Architect 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 (e.g. the User, or the back end Repository) are placed within a containing swim lane to indicate their association.

To manage swimlanes, select the **Diagram** | **Swimlanes and Matrix** menu option to display the **Swimlanes**.
and Matrix dialog. The dialog defaults to the Swimlanes tab.

This dialog enables you to set the orientation (vertical or horizontal), line color and width of the swimlanes, and lock the swimlanes to prevent further movement. You can also specify the font color and bold font, hide names, hide the classifier and show the name in the title bar. Use the New, Modify and Delete buttons to change aspects of the selected swimlane. Use the (up and down) buttons to change the order of swimlanes within the diagram.

If you set a background color for a swimlane, it takes on the same shading profile as the main diagram background.

See Also
- Swimlanes Matrix
5.2.2.20 Swimlanes Matrix

Enterprise Architect diagrams support a Swimlanes Matrix for all diagram types, based on the Zachman Framework.

<table>
<thead>
<tr>
<th>The Zachman Framework</th>
<th>DATA Model (Things)</th>
<th>FUNCTION (Processes)</th>
<th>NETWORK (Location)</th>
<th>PEOPLE (Roles)</th>
<th>TIME (Time)</th>
<th>MOTIVATION (Why)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUSINESS MODEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM MODEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TECHNOLOGY MODEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DETAILED REPRESENTATIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNCTIONING ENTERPRISE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Swimlanes Matrix divides the diagram into cells of vertical columns and horizontal rows. The cell in the top left corner of the Swimlanes Matrix contains the heading of the matrix. The first cell at the top of each column contains the column title text. The first cell at the left of each row contains the row title text.

Set up Swimlanes Matrix

To set up and manage the Swimlanes Matrix, select the Diagram | Swimlanes and Matrix menu option to display the Swimlanes and Matrix dialog. Click on the Matrix tab.
Activate the Matrix
To activate the Swimlanes Matrix, select the Active check box.
At the same time, you can define the line width for all lines on the matrix; in the Line Widths field, click on the drop-down arrow and select the appropriate width.

Create the Heading of the Swimlanes Matrix
To define the heading for the matrix, follow the steps below.

1. Click on the New button.
2. In the Type field in the Details of New Column panel, click on the drop-down arrow and select Heading.
3. In one or more of the Title fields, type the heading name. You can enter up to three text strings as heading text.
4. If necessary, click on the Color, Font and Back options and select the heading text font, color and background color.
5. Click on the Save button in the Operations panel. The Heading cell displays on the diagram.

Note:
The heading is the first item in the list; you create only one heading.

Create Columns and Rows:
To define the column and row headings for the matrix, follow the steps below.

1. Click on the New button.
2. In the Type field in the Details of New Column panel, click on the drop-down arrow and select either Column or Row as appropriate.
3. In one or more of the Title fields, type the column or row name. You can enter up to three text strings as title text.
4. If necessary, click on the Color, Font and Back options and select the title text font, color and background color.
5. Click on the Save button in the Operations panel. The column or row heading cell and column or row...
lines display on the diagram.

**Note:**
When you define columns and rows, you define the header or title cells. The properties of these cells do not reflect on the matrix cells themselves. For example, the intersection cell of a column and row has a transparent background and therefore takes the color and shading effect of the diagram background.

**Lock the Matrix**
To lock the matrix so that it cannot be edited on the diagram, on the Swimlanes and Matrix dialog select the Lock checkbox.

**Edit items in the list:**
As you create the heading, column and row title cells, they are added to the list in the bottom of the dialog. To edit an item, follow the steps below.

1. Click on the required item in the list.
2. Make the relevant changes in the Edit Selected ... panel.
3. Click on the Save button in the Operations panel.

**Delete items from the list:**
To delete the heading or a column or row from the matrix, follow the steps below.

1. Click on an item in the list.
2. Click on the Delete button in the Operations panel.

**Model Profiles:**
After creating a Swimlane Matrix, you can save it into a Model Profile and apply it to other diagrams. Model Profiles are available on any diagram in your model.

**Save a Model Profile:**
To save a Model Profile, follow the steps below.

1. In the Model Profiles panel, click on the Save button. The Save Model Profile dialog displays.
2. In the Name field, type the name of your profile.
3. Click on the OK button.

The profile is now visible in the profile name drop-down list here and on other diagrams.

**Note:**
You can also transport all the matrix profiles between models (as Diagram Matrix Profiles), using the Export Reference Data and Import Reference Data options on the Tools menu.

**Apply a Model Profile:**

**Note:**
By applying a Model Profile, you replace the current profile. Save the current profile to avoid losing it.

To apply a Model Profile to a diagram, follow the steps below.

1. In the Model Profiles panel, click on the drop-down arrow of the profile name field, and select the required profile from the list.
   The list contains a predefined Zachman profile, as well as an empty profile should you want to replace the current profile with one that you create on the spot.
2. A confirmatory prompt displays. Click on the OK button to display the profile details on the Swimlanes and Matrix dialog.
3. Click on the OK button at the bottom of the Swimlanes and Matrix dialog to apply the profile to the matrix on the diagram.
Size the Matrix
To size the rows and columns, drag the row and column borders on the diagram.
Elements placed inside each cell are shifted when sizing. To prevent the elements shifting, press and hold [Ctrl] while sizing.

See Also
- Swimlanes

5.2.2.21 Using the Image Manager
The Image Manager dialog enables you to insert alternative images in diagrams, rather than inserting standard UML elements. For example, you might want to place a custom background image on a diagram, or display a custom image such as a Router or PC on a UML element.

Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Configure Images permission to configure alternative element images.

To display the Image Manager dialog, either:
- Right-click on the element within the diagram and, from the context menu, select the Appearance | Alternate Image option, or
- Select the element in the diagram and press [Ctrl]+[Shift]+[W].

To locate and display an image, click on individual image filenames, or press [↑] and [↓] to scroll through the list of images. As you highlight each image filename, the Preview panel changes to reflect the image. Double-click on the required image filename to display the image in full size.

On the Image Manager dialog, the following buttons are available:
What Is UML Modeling?  | Work With Diagrams

### Option & Function Keys

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>View [Alt]+[V]</td>
<td>Display the selected image in full size.</td>
</tr>
<tr>
<td>Add New [Alt]+[A]</td>
<td>Browse appropriate directories to search for and import new images. You can import images in .BMP, .PNG, .EMF, .WMF, .TGA, .PCX or .JPG format. Internally, Enterprise Architect stores the images in .PNG or metafile format to conserve space.</td>
</tr>
<tr>
<td>Update Selected [Alt]+[U]</td>
<td>Refresh the selected image; for example, after it has been modified.</td>
</tr>
<tr>
<td>Delete [Alt]+[D]</td>
<td>Delete the selected image. A message displays to indicate how many elements use the image. Click on the Continue button to delete information about the image from those elements, which then revert to their previous appearance.</td>
</tr>
<tr>
<td>Close</td>
<td>Close the Image Manager dialog.</td>
</tr>
<tr>
<td>OK [Alt]+[O]</td>
<td>Confirm selection of the alternative image for the element selected in the diagram.</td>
</tr>
</tbody>
</table>

**Notes:**

- If you are creating many elements of the same type that have a particular image, you should use a custom stereotype with an associated metafile.
- You can transport image files between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

### 5.2.2.21.1 Create Custom Diagram Background

Enterprise Architect diagrams have a single-color ‘wash’ background that you can set to a solid color or a fade gradient down the screen. You set the color and whether to have a fade gradient on the Standard Colors page of the Options dialog (select the Tools | Options | Standard Colors option). Alternatively, using the Image Manager dialog, you can create a non-tiled background for diagrams. To perform this operation follow the steps below:

1. Create a Boundary object from the Use Case Elements page of the Enterprise Architect UML Toolbox. Do not use the Boundary element from any other section of the Toolbox.
2. Stretch the Boundary to a size that can contain all of the elements you intend to place on the diagram, and drag it to the edges of the diagram workspace.
3. Right-click on the Boundary element. The context menu displays.
4. Select the Z-Order | Send to Bottom menu option. This ensures that the Boundary is not displayed in front of any other element in the diagram.
5. Either:
   - Press [Ctrl][Shift][W], or
   - Right-click on the Boundary to display the context menu, and select the Appearance | Alternate Image menu option.
6. On the Image Manager dialog, select an appropriate image as the diagram background and ensure that the image size is large enough to span the required size of the diagram background.
7. When you have selected the required image, click on the OK button.

### 5.2.2.21.2 Import Image Library

Using the Image Library enables you to create attractive diagrams with custom images. A bundled clip art collection of UML-based images is available as an Imported Image Library, from www.sparxsystes.com/resources/image_library.html. Image libraries enable you 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, are only available for use in conjunction with Enterprise Architect, and are supplied on the understanding that they are not used under any other circumstance.

Import an Image Library
To import an Image Library you must have a suitable Image Library file. To import the Image Library, follow the steps below:

2. Select the Tools | Import reference data menu option. The Import Reference Data dialog displays.
3. Locate the XML Image Library file to import using the Select File button. The file name is ImageLibrary.xml in the directory in which you saved the file.
4. Select the data set containing the Image Library. Then click on the Import button.

Use the Image Library
To use the images contained within the Image Library, follow the steps below:

1. Create a diagram to associate with the images contained in the Image Library.
2. Select the element to change from the default appearance to one of the images contained within the library.
3. Press [Ctrl]+[Shift]+[W], or right-click on the selected element to display its context menu and then select the Appearance | Select Alternate Image option.
4. On the Image Manager dialog, in the Name field highlight the appropriate image name and then click on the OK button.
5.2.2.22 Show Realized Interfaces of 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 as represented by the interface nodes protruding from the Class. Class1 is dependent on these two interfaces, which is shown by the Dependency connectors 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 Embedded Elements | Show Realized Interfaces. This setting only applies to the selected Class, and can be changed at any time.

5.2.2.23 Label Menu Section

You can add labels to both connectors and elements, using the element or connector context menu as follows:

- **Element** - Select the Embedded Elements menu option and either the Add <element> option or the Embedded Elements option; the label is the embedded element name
- **Connector** - Select the Properties option and define the connector name, stereotype, constraints and/or source and target roles.
Once you have these labels, you can edit and format them using the **Labels** context menu.

To display the **Labels** context menu, right-click on a label.

**Note:**

As labels can be concentrated on and around the element or connector, make sure that you click on a section of the required label that is clear of any other label or structure.

### Element Labels

The **Labels** menu associated with embedded elements provides the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Label Color</td>
<td>Specify a color for the label.</td>
</tr>
<tr>
<td>Hide Label</td>
<td>Hide the label; to unhide the label use the <strong>Show label</strong> option in the <strong>Embedded Elements</strong> context menu.</td>
</tr>
<tr>
<td>Bold</td>
<td>Set the label font to bold.</td>
</tr>
<tr>
<td>Text Alignment</td>
<td>Align the text within the label text area. The options available from the submenu enable you to specify left, center and right alignment.</td>
</tr>
<tr>
<td>Label Rotation</td>
<td>Orient the label in the horizontal or vertical planes, with the vertical plane offering the option of clockwise or anti-clockwise position.</td>
</tr>
<tr>
<td>Default Position</td>
<td>Move the label to the initial default location.</td>
</tr>
<tr>
<td>Default Color</td>
<td>Set the label color to the default color.</td>
</tr>
</tbody>
</table>

### Connector Labels

The **Labels** menu associated with connectors provides the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Label Color</td>
<td>Specify a color for the label.</td>
</tr>
<tr>
<td>Hide Label</td>
<td>Hide the label; to unhide the label use the <strong>Set Label Visibility</strong> option on the connector context menu.</td>
</tr>
<tr>
<td>Bold</td>
<td>Set the label font to bold.</td>
</tr>
<tr>
<td>Text Alignment</td>
<td>Align the text within the label text area. The options available from the submenu enable you to specify left, center and right alignment.</td>
</tr>
<tr>
<td>Label Rotation</td>
<td>Orientate the label horizontally or vertically and, if vertically, in a clockwise or anti-clockwise position.</td>
</tr>
<tr>
<td>Direction</td>
<td>Set a small arrow at the end of the label pointing to either the label source or the destination dependent upon selection from the available options.</td>
</tr>
<tr>
<td>Default Position</td>
<td>Move the label to the default location.</td>
</tr>
<tr>
<td>Default Color</td>
<td>Set the label color to the initial default color.</td>
</tr>
</tbody>
</table>

#### 5.2.2.24 Lock Diagram

You can lock a diagram against inadvertent changes, such as moving or sizing elements.

To lock a diagram, follow the steps below:

1. Open the diagram to lock.
2. Right-click on the background to open the diagram context menu.
3. Click on the **Lock Diagram** option to prevent further changes.
4. Click on the **OK** button.
5.2.2.25 Undo Last Action

When editing diagrams, Enterprise Architect supports multiple undo levels for moving, re-sizing and deleting elements, and for deleting connectors.

There are three ways to undo the last action:

- Press [Ctrl]+[Z]
- Select the Edit | Undo menu option
- Click on the Undo button in the Default Tools toolbar.

Warning:
Currently you cannot undo element additions or connector moves.

5.2.2.26 Redo Last Action

When editing diagrams, Enterprise Architect supports multiple undo levels for moving, re-sizing and deleting elements, and for deleting connectors. If an Undo action is in error, you can redo the action to reverse the Undo.

There are three ways to redo the last action:

- Press [Ctrl]+[Y]
- Select the Edit | Redo menu option
- Click on the Redo button in the Default Tools toolbar.

5.2.2.27 View Last and Next Diagram

Enterprise Architect enables you to step backwards and forwards through the currently-open diagrams, including the Start Page.

To view the previous or next diagram use the Previous or Next buttons on the Diagram toolbar.

Use the Home button to display the default project diagram (if one has been specified).

5.2.2.28 Diagram Properties

You can set several properties of a diagram using the diagram Properties dialog. Some properties influence the display and some are logical attributes that appear in the documentation.

Note:

You can also set the default diagram background color and gradient, and the element fill color and gradient, on the Standard Colors page of the Options dialog (select the Tools | Options | Standard Colors option).
There are several options for opening the diagram Properties dialog for a given diagram:

- Select the **Diagram | Properties** menu option 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 **Properties** from the context menu
- Double-click in the background of the open diagram.

In the Diagram Properties dialog you can set 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, click on the **OK** button to save and exit.

See the following topics:

- **General Tab**
- **Diagram Tab**
- **Elements Tab**
- **Features Tab**
- **Connectors Tab**

### 5.2.2.28.1 General Tab

The General tab of the diagram Properties dialog enables you to define characteristics of the overall diagram, such as its title, version and modification date.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Update Diagrams** permission to update diagram properties.
### Field Guide: Diagram Properties

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type the name of the diagram (defaults to the name of the parent package).</td>
</tr>
<tr>
<td>Author</td>
<td>Type or select the name of the person who created the diagram.</td>
</tr>
<tr>
<td>Version</td>
<td>Type the version number of the diagram (defaults to 1.0).</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Type or select the name of the stereotype for the diagram.</td>
</tr>
<tr>
<td>Created</td>
<td>Automatically display the date the diagram was created.</td>
</tr>
<tr>
<td>Modified</td>
<td>Type the date and time on which the diagram was last modified (defaults to the current date and time).</td>
</tr>
<tr>
<td>Notes</td>
<td>Type any additional notes about the diagram. You can format the notes using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
</tbody>
</table>

![Diagram Properties](image_url)

Documentation for the **Communication** package.
5.2.28.2 Diagram Tab

The Diagram tab of the diagram Properties dialog enables you to define the structure of the diagram.

### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Alias if Available</td>
<td>Display the element alias as the name if the alias is specified.</td>
</tr>
<tr>
<td>Show Additional Parents</td>
<td>Show the name of all parents not in the current diagram for all Classes and interfaces.</td>
</tr>
<tr>
<td>Show Page Border</td>
<td>Show a page border to align elements with.</td>
</tr>
<tr>
<td>Show Diagram Details</td>
<td>Show diagram details in a note in the top left corner of the diagram. (Deselect to hide the diagram details.)</td>
</tr>
<tr>
<td>Show Sequence Notes</td>
<td>Show the Sequence Notes on the current diagram.</td>
</tr>
<tr>
<td>Show Namespace</td>
<td>Show the namespace of each element on the diagram, under the element.</td>
</tr>
<tr>
<td>Always show as Element List</td>
<td>Always display the diagram contents as an Element List rather than as a diagram.</td>
</tr>
</tbody>
</table>

### Page Setup

- A4 Sheet, 210 mm by 297 mm
- Landscape

### RTF Document Options

- Exclude image from RTF Documents
- Divide Diagram into Multiple Pages
- Document each contained element in RTF
- Rotate Images

### General Options

- **Appearance**
  - Use Alias if Available
  - Show Additional Parents
  - Show Page Border
  - Show Diagram Details
  - Show Sequence Notes
  - Show Namespace
  - Always show as Element List

- **Page Setup**
  - A4 Sheet, 210 mm by 297 mm
  - Landscape

- **Print Page Header and Page Footer**

- **Set Layout Style**

---

© 1998-2008 Sparx Systems Pty Ltd
### What Is UML Modeling?

**Work With Diagrams**

#### Rotate Images
- **Use to:** Rotate each diagram image by 90 degrees in the RTF document.
- **Note:** Only valid for bitmap (.bmp) images.

#### 5.2.2.28.3 Elements Tab

The Elements tab of the diagram Properties dialog enables you to define what components of the elements should be displayed on the diagram.

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
</table>
| Use Stereotype Icons| For elements that have whole shapes drawn by Enterprise Architect (such as Analysis stereotypes), draw the alternative shape (if defined).
| Show Element Stereotypes| For elements that have an icon displayed in the top right corner, (such as an Artifact element) if Show Element Stereotypes is selected, display the stereotype icon instead of the stereotype text. |
| Show Table Owner    | Display the Table Owner. For more information, see the Set Table Owner topic. |
| Show Element Property String | Show the advanced property string for all elements; e.g. {leaf}. |
| Show Compartments   | Enable the following compartments to be shown or hidden for any element |
What Is UML Modeling? | Work With Diagrams

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>using rectangle notation:</td>
<td>- Attributes</td>
</tr>
<tr>
<td></td>
<td>- Operations</td>
</tr>
<tr>
<td></td>
<td>- Tags (Tagged Values)</td>
</tr>
<tr>
<td></td>
<td>- Requirements</td>
</tr>
<tr>
<td></td>
<td>- Constraints</td>
</tr>
<tr>
<td></td>
<td>- Testing (<a href="#">Testing Scripts</a>)</td>
</tr>
<tr>
<td></td>
<td>- Maintenance (<a href="#">Maintenance Scripts</a>)</td>
</tr>
<tr>
<td></td>
<td>- Package Contents</td>
</tr>
<tr>
<td></td>
<td>- Notes.</td>
</tr>
</tbody>
</table>

### 5.2.2.28.4 Features Tab

The **Features** tab of the diagram Properties dialog enables you to define the features of the diagram.

<table>
<thead>
<tr>
<th>Field Options</th>
<th>Visible Class Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Qualifiers and Visibility Indicators</td>
<td>Public</td>
</tr>
<tr>
<td>Show Stereotypes</td>
<td>Protected</td>
</tr>
<tr>
<td>Show Property String</td>
<td>Private</td>
</tr>
<tr>
<td>Show Operation Return Type</td>
<td>Package</td>
</tr>
<tr>
<td>Suppress Brackets for Operations without Parameters</td>
<td>Property Methods</td>
</tr>
</tbody>
</table>

**Show Attribute Details**

- **Name and Type**

**Show Parameter Detail**

- **Type Only**

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Qualifiers and Visibility Indicators</td>
<td>Show or hide the qualifiers and visibility indicators on the diagram.</td>
</tr>
<tr>
<td></td>
<td><strong>Qualifiers</strong> include such things as the 'derived' symbol (/) and the public key symbol (PK).</td>
</tr>
<tr>
<td></td>
<td><strong>Visibility indicators</strong> include such things as +, -, # and ~, which indicate the scope of access of the item (such as an attribute, operation or role).</td>
</tr>
<tr>
<td>Show Stereotypes</td>
<td>Show the stereotypes on all features.</td>
</tr>
<tr>
<td>Show Property String</td>
<td>Show the advanced property string for all element features, e.g. (readOnly).</td>
</tr>
<tr>
<td>Show Operation Return Type</td>
<td>Display the return data type of operations.</td>
</tr>
<tr>
<td>Suppress Brackets for Operations Without Parameters</td>
<td>Suppress brackets on operations that have no parameters; i.e. Opn; rather than Opn().</td>
</tr>
</tbody>
</table>
What Is UML Modeling? | Work With Diagrams

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible Class Members</td>
<td>Hide Class members according to their scope and methods that specify properties. See the Visible Class Members topic.</td>
</tr>
<tr>
<td>Show Attribute Detail</td>
<td>Select whether to show both the attribute name and type or the attribute name only.</td>
</tr>
<tr>
<td>Show Parameter Detail</td>
<td>Control the display of method parameters. See the Visible Class Members topic.</td>
</tr>
</tbody>
</table>

5.2.28.5 Connectors Tab

The Connectors tab of the diagram Properties dialog enables you to define the appearance of the connectors on the diagram.

Field Controls:

- **Show Relationships**: Show relationships in the current diagram.
- **Show Collaboration Numbers**: Show numbering in Communication diagrams.
- **Show Non-Navigable Ends**: Indicate when an Association end is not navigable; a cross is presented at the Association connector.
- **Show Connector Property String**: Show the property string for connectors.
- **Suppress All Connector Labels**: Hide all connector labels.

**Connector Notation**

- **UML 2.1**: use the standard UML 2.1 notation for connectors.
- **Information Engineering**: use the Information Engineering (IE) connection style; for more information see the [http://www.agiledata.org/essays/dataModeling101](http://www.agiledata.org/essays/dataModeling101) page.
- **IDEFX1**: use the Integrated Definition Methods IDEFX1 connection style; for more information see the [http://www.idef.com/IDEF1X.html](http://www.idef.com/IDEF1X.html) page.

Enterprise Architect User Guide
5.2.2.28.6 Visible Class Members

On the Features tab of the diagram Properties dialog, the Visible Class Members panel enables you to hide Class members by their scope and methods that specify properties. Use the checkboxes to define the visibility of Class members.

Show Parameter Detail

The Show Parameter Detail field enables you to control the display of method parameters with the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No details shown.</td>
</tr>
<tr>
<td>Type Only</td>
<td>Shows only the type of parameter.</td>
</tr>
<tr>
<td>Full Details</td>
<td>Shows all of the details for parameters.</td>
</tr>
<tr>
<td>Name Only</td>
<td>Shows the name of the parameter only.</td>
</tr>
</tbody>
</table>

5.2.2.29 Set the Default Diagram

A project might have a default diagram. If set, this diagram loads when Enterprise Architect first opens the model. It is often convenient to place hyperlinks to other diagrams and resources on the default diagram, thus creating a Home Page for your model.

To set the currently active diagram as the model default, select the Diagram | Make Model Default menu option. (Also use this option to cancel the default setting.)

Tip:

Once you have specified a default diagram, the Home icon on the Diagram toolbar takes you to that diagram.

5.2.2.30 Create Legends

A Legend shape identifies colors and styles you have used to group other elements on the diagram. You can use the Legend to assist in distinguishing different elements, connectors or systems on the diagram. For example, the Legend could show that all elements concerned with the management system are shaded in blue, and all outcomes connectors are shown in red. The Legend displays as a key to the diagram, with the filled shape styles first and the lines and connector styles underneath.
You add a Legend to the diagram, then edit it to add Legend elements, which define the colors and styles used in the diagram.

**Add a Legend**

To add a Legend to a diagram, click on the **New Diagram Legend** icon ( ) on the **UML Elements** toolbar. The **Legend** dialog displays.

Click on the **OK** button. The Legend displays on the diagram as a simple rectangle.

**Edit a Legend**

To edit the Legend follow the steps below:

1. Either:
   - Double-click on the Legend, or
   - Right-click on the Legend and select the **Properties** option from the context menu. The **Legend** dialog displays.
What Is UML Modeling? | Work With Diagrams

Note:
The Legend dialog enables you to add, delete, modify or re-sequence Legend elements. Use the Fill tab to define the Legend elements for shapes, then click on the Line tab to define Legend elements for lines and connectors.

2. In the Name field, type the name of the Legend element; for example, Management System or Help.
3. Use the drop-down arrows to select the fill color, line color and line thickness for the Legend element.
4. Click on the Save button to save the Legend element. The element displays in the Fill or Line tab, as appropriate.
5. Click on the New button to add another Legend element.

**Style Options**

Click on the Style Options button [...] to display the Style Options dialog, on which you can modify a Legend title, font size, background color and border color. If you choose default options for the colors, the Legend automatically assumes colors based on the diagram background color.

<table>
<thead>
<tr>
<th>Legend Title:</th>
<th>Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Size:</td>
<td>16</td>
</tr>
<tr>
<td>Font Color:</td>
<td></td>
</tr>
<tr>
<td>Background:</td>
<td></td>
</tr>
<tr>
<td>Border Color:</td>
<td></td>
</tr>
</tbody>
</table>

Click on the OK button on the Style Options dialog and again on the Legend dialog. The Legend displays on the diagram.
5.2.2.31 Scale Image to Page Size

When you print a diagram, the default setting is to scale the image to fit the size of the printer paper you have defined in the page set-up. The image is not scaled up to fill the page, but it is scaled down if it exceeds the current page boundary. The image retains its current proportions; that is, it is scaled down equally in the X and Y dimensions. For a large diagram, this can mean that the components of the diagram are small and hard to read.

Alternatively, you can print a multi-page image; that is:

- allow the diagram image to print on as many printer pages as it naturally occupies, (no scaling), or
- scale the diagram image to exactly fit a specified number of pages.

In all three cases you also define the paper size and orientation.

Scale Images

To turn off or customize image scaling options, follow the steps below:

1. Select the diagram to scale.
2. Double-click on the diagram background to display the <type> Diagram: <name> dialog, or right-click on the background and select the Properties option from the context menu.
3. Click on the Diagram tab and, in the Page Setup panel click on the Advanced button.

The Print Advanced dialog displays.

From the Print Advanced dialog the following options are available:

- **None**: select to print on as many pages as the diagram image covers
- **Scale to 1 page**: select to scale the diagram image to fit on the currently selected page
- **Custom**: select to specify the width and height of the diagram images across a specified number of pages
- **Page Setup**: click to select the page size and alignment

Note:

Before printing, make sure you have selected the required page layout using the Page Setup button.
5.2.2.32 Set Diagram Page Size

You can change the size of the diagram area (or scrollable/printable area) using the Diagram Properties dialog.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Update Diagrams** permission to change diagram page setup.

To set the page size, follow the steps below:

1. Load a diagram.
2. Double-click on the background to open the Diagram Properties dialog.
3. Click on the Diagram tab and, in the Page Setup section, click on the Advanced button. The Print Advanced dialog displays.

5. In the Size field, click on the drop-down arrow and select an appropriate page size. In the Orientation panel click on the radio button for the orientation of the page to print.
6. Click on the OK button.

The scrollable area for your diagram is 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 Diagram page of the Options dialog (select the Tools | Options | Diagram menu option). Once the paper size is set there, all new diagrams have that as the default size.

See the **Configure Local Options - Diagram** topic.
5.2.2.33 Pan and Zoom a Diagram

**Pan**

Pan the Diagram View in the following ways:

- Use [−], [−], [↑], [↓], [Page Up], [Page Down], [Home] and [End] when the Diagram View is selected
- Use the scrollbars
- Use the middle mouse button
- Use the Pan & Zoom window.

**Zoom**

You can zoom into and out from a diagram using the zoom buttons on the diagram toolbar, or by using the Diagram | Zoom submenu.

Change the zoom level by 10% by clicking on either the **Zoom In (+)** or **Zoom Out (-)** buttons. Alternatively, select the **Zoom In** or **Zoom Out** options from the Diagram | Zoom submenu.

**Note:**

This sub-menu is a tear off menu.

There are three ways to return the diagram to 100%:

- Click on the button
- Select **Zoom to 100%** from the Diagram | Zoom submenu
- [Ctrl]+middle-click the mouse.

**Tip:**

You can zoom in and out of the main window dynamically by holding [Ctrl] and rolling the mouse wheel.

**Note:**

At high levels of zoom, element features 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.
5.3 Work With 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 2.1.1 are depicted in the following diagrams:
What Is UML Modeling? | Work With Elements

Enterprise Architect User Guide
5.3.1 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 is displayed.

The element context menu is split into a number of sections and submenus:

- **Properties**
- **Add**
- **Find**
- **Transform** ([Ctrl]+[H]) - Transform the selected element from one domain to another
- **Embedded Elements**
- **Features**
- **Generate DDL** - Generate DDL for a table, procedure or view Class
- **Code Engineering**
- **Appearance**
- **UML Help** - display the Enterprise Architect Help topic for the UML element type
- **Delete** ([Ctrl]+[D]) - delete the element.

**Note:**
Context menus vary between element types. The **Code Engineering** options won’t display for a Use Case element, for example.

### Example Context Menu for a Class:

```
Add-In...
Properties...  Alt+Enter
Advanced
Linked Document...  Ctrl+Alt+D
Add
Find
Transform...  Ctrl+H
Embedded Elements
Attributes...
Operations...
Set Feature Visibility...  Ctrl+Shift+V
Lock Element...
Selectable
Appearance
Z-Order
UML Help
Delete 'Activity1'  Ctrl+D
```

### Example Context Menu for an Activity:
5.3.1.1 Properties Menu Section

The Properties section of the element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties [Ctrl]+[Enter]</td>
<td>Open the Properties dialog for the selected element. For State Lifeline and Value Lifeline elements, display the Configure Timeline dialog.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Open the Advanced sub-menu.</td>
</tr>
<tr>
<td>Other Properties</td>
<td>For State Lifeline and Value Lifeline elements, display the Properties dialog for the selected element.</td>
</tr>
<tr>
<td>Create (or Edit) Linked Document [Ctrl]+[Alt]+[D]</td>
<td>(Corporate Edition) Create an RTF document linked to the element.</td>
</tr>
<tr>
<td>Delete Linked Document</td>
<td>Delete an existing linked document for the element.</td>
</tr>
</tbody>
</table>
5.3.1.1 Advanced Submenu

The Advanced submenu on an element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Properties [Ctrl]+[Shift]+[Enter]</td>
<td>Open the Custom Properties dialog.</td>
</tr>
<tr>
<td>Parent [Ctrl]+[I]</td>
<td>Set the element parent.</td>
</tr>
<tr>
<td>Instance Classifier [Ctrl]+[L]</td>
<td>Set the instance classifier for the element.</td>
</tr>
<tr>
<td>Classifier Properties [Ctrl]+[Alt]+[Enter]</td>
<td>Open the Properties dialog for the classifier of the selected element.</td>
</tr>
<tr>
<td>Make Composite</td>
<td>Set the element as a Composite element.</td>
</tr>
<tr>
<td>Change to State (Value) Lifeline</td>
<td>Switch one type of Lifeline element to the other.</td>
</tr>
<tr>
<td>Show Composite Diagram</td>
<td>Display a mini-picture of the contents of a composite element within that element.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>Define the multiplicity for the element.</td>
</tr>
<tr>
<td>Edit Extension Points</td>
<td>For an extended Use Case, display the Use Case Extension Points dialog, which you use to insert the point at which the behavior should be inserted.</td>
</tr>
<tr>
<td>Association Class</td>
<td>Connect the Class to a new Association (if the element is a Class).</td>
</tr>
<tr>
<td>Use Rectangle (Circle) Notation</td>
<td>Use rectangle notation for the element.</td>
</tr>
<tr>
<td>Partition Activity</td>
<td>Define an Activity Partition.</td>
</tr>
<tr>
<td>Set Run State</td>
<td>Add a new instance variable to the element using the Define Run State dialog.</td>
</tr>
<tr>
<td>Set Property Value</td>
<td>Set the property value for the Part, using the Set Property Values dialog.</td>
</tr>
<tr>
<td>Override Attribute Initializers [Ctrl]+[Shift]+[R]</td>
<td>Pre-define initial values for attributes that can be used to override existing defaults.</td>
</tr>
<tr>
<td>Convert to Instance</td>
<td>Convert this classifier to an instance.</td>
</tr>
<tr>
<td>Convert Linked Element To Local Copy</td>
<td>Convert the occurrence of the element on this diagram from a link to the original element to a local copy of the element.</td>
</tr>
<tr>
<td>Make Sender/Receiver</td>
<td>Toggle the element from a sender to a receiver and vice versa.</td>
</tr>
<tr>
<td>Accept Time Event</td>
<td>Change the notation for an Accept Event action to a Accept Time Event action.</td>
</tr>
<tr>
<td>Set Object State [Ctrl]+[Shift]+[S]</td>
<td>Set the state of an instance classifier based on the child states for that object.</td>
</tr>
<tr>
<td>Define Concurrent Substates</td>
<td>Define a set of substates that can be held simultaneously within that composite state.</td>
</tr>
<tr>
<td>Use State Label Notation</td>
<td>Display State Label Notation for a State object (the element name is displayed on a box on top of the element rather than inside it).</td>
</tr>
<tr>
<td>Deep History</td>
<td>Change the type of pseudo-state to a Deep History. Applies only when right-clicking on a History pseudo-state.</td>
</tr>
<tr>
<td>Set Attached Links</td>
<td>Attach the selected Note element to a connector, or several connectors.</td>
</tr>
<tr>
<td>Link to Diagram Note</td>
<td>Convert a Note to be connected to an element feature (via the connector context menu) to display the element feature as the note text. The option simply deletes any current text and blocks the note from being edited other</td>
</tr>
</tbody>
</table>
What Is UML Modeling?  | Work With Elements

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>than through the feature Properties dialog.</td>
</tr>
</tbody>
</table>

**Note:**

Not all menu options shown here are present on all element context menus. Context menus vary between element types. The Partition Activity option only displays for an Activity element, for example.

### 5.3.1.1.2 Custom Properties Dialog

Certain elements and connectors feature the Custom Properties option in their context menu. The following example shows the CustomProperties dialog for an Activity element. Properties differ between the various types of element or connector.

As shown above, you can change the values of properties either by selecting the value from the property's drop-down list or by typing the value in the field to the right of the property.

### 5.3.1.2 Add Submenu

The Add submenu enables you to add supporting elements and diagrams to the selected element.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tagged Value</td>
<td>Add a Tagged Value</td>
</tr>
<tr>
<td>Related Elements</td>
<td>Open the Insert Related Elements dialog.</td>
</tr>
<tr>
<td>Note</td>
<td>Create and attach a blank Note element to the current element.</td>
</tr>
<tr>
<td>Constraint</td>
<td>Create and attach a blank Constraint element to the current element.</td>
</tr>
<tr>
<td>Activity</td>
<td>Add an Activity element as a child of the current Classifier element, with either an Activity diagram or an Interaction Overview diagram.</td>
</tr>
<tr>
<td>Interaction</td>
<td>Add an Interaction element as a child of the current Classifier element, with either a Sequence diagram, a Communication diagram or a Timing diagram.</td>
</tr>
<tr>
<td>State Machine</td>
<td>Add a State Machine element as a child of the current Classifier element, with a State Machine diagram.</td>
</tr>
<tr>
<td>Add Diagram</td>
<td>Add a child diagram to the Classifier element, using the New Diagram dialog.</td>
</tr>
</tbody>
</table>
What Is UML Modeling? | Work With Elements

Note:
Not all menu options shown here are present on all element context menus. Context menus vary between element types. The options relating to Classifiers, for example, are not available for Object elements.

5.3.1.2.1 Insert Related Elements

The Insert Related Elements dialog can be accessed from the Add | Related Elements option on most element context menus. This dialog enables you to insert connected elements from elsewhere in the model into the current diagram.

You can specify the following details:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert elements to: «x» levels</td>
<td>Select the level down to which to insert connected elements, between levels 1 and 5. You can select levels 4 or 5 to see how far the element/relationship hierarchy extends, but as this can produce a complicated and tangled diagram, it is better to use level 1 or 2 on selected elements in turn.</td>
</tr>
<tr>
<td>For Link Type</td>
<td>Select a type of connector to limit the inserted elements to those connected by that relationship type.</td>
</tr>
<tr>
<td>With Link Direction</td>
<td>Select whether the connectors are to be a single direction or bi-directional.</td>
</tr>
<tr>
<td>Limit to Element Type</td>
<td>Select a type of element to limit the inserted elements to those of that element type.</td>
</tr>
<tr>
<td>Layout Diagram When Complete</td>
<td>Select whether Enterprise Architect should layout the diagram after the elements have been inserted.</td>
</tr>
<tr>
<td>Limit to this Namespace</td>
<td>Select a specific namespace from which the inserted elements are to come.</td>
</tr>
</tbody>
</table>

Note:
If no elements have been added, this option has no effect. Elements have to be added for Enterprise Architect to adjust the layout.

5.3.1.3 Find Submenu

The Find submenu on the element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Project Browser [Alt]+[G]</td>
<td>Highlight the currently selected element in the Project Browser.</td>
</tr>
<tr>
<td>Locate Classifier In Project</td>
<td>Highlight the classifier for the currently-selected object, in the Project</td>
</tr>
</tbody>
</table>
### Menu Option & Function Keys

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser [Ctrl]+[Alt]+[G]</td>
<td>Browser.</td>
</tr>
<tr>
<td>In Diagrams [Ctrl]+[U]</td>
<td>Open the Element Usage dialog.</td>
</tr>
<tr>
<td>Custom References [Ctrl]+[J]</td>
<td>Set up Cross References.</td>
</tr>
<tr>
<td>Add to Favorites</td>
<td>Add the element to the Favorites folder in the Resources window.</td>
</tr>
</tbody>
</table>

### 5.3.1.4 Embedded Elements Submenu

The Embedded Elements submenu on the element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Port</td>
<td>Add an embedded Port to the element.</td>
</tr>
<tr>
<td>Add Required Interface</td>
<td>Add an embedded Required Interface to the element.</td>
</tr>
<tr>
<td>Add Provided Interface</td>
<td>Add an embedded Provided Interface to the element.</td>
</tr>
<tr>
<td>Add Action Pin</td>
<td>Add an embedded Action Pin to the element.</td>
</tr>
<tr>
<td>Add Expansion Node</td>
<td>Add an embedded Expansion Node to the element.</td>
</tr>
<tr>
<td>Add Object Node</td>
<td>Add an embedded Object Node to the element.</td>
</tr>
<tr>
<td>Add Activity Parameter</td>
<td>Add an embedded Activity Parameter to the element.</td>
</tr>
<tr>
<td>Add Entry Point</td>
<td>Add an embedded Entry Point to the element.</td>
</tr>
<tr>
<td>Add Exit Point</td>
<td>Add an embedded Exit Point to the element.</td>
</tr>
<tr>
<td>Embedded Elements</td>
<td>Open the Embedded Elements window.</td>
</tr>
<tr>
<td>Show Realized Interfaces</td>
<td>Display each interface directly realized by a Class.</td>
</tr>
<tr>
<td>Show Dependent Interfaces</td>
<td>Display each dependency relationship for that model element as a lollipop style node attached to its left-hand side.</td>
</tr>
</tbody>
</table>

**Note:**

Not all menu options shown here are present on all element context menus. Context menus vary slightly between element types. Of the Add options, only Add Port displays for a Class element, for example.
5.3.1.4.1  Embedded Elements Window

The Embedded Elements dialog enables 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, click on the New button to create a new embedded element. Enter details such as type, name and stereotype, and click on the OK button. The embedded element now shows 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.

To incorporate inherited or owner properties, select the Show Owned/Inherited checkbox.

The name of the embedded element is a label, which you can edit using the Labels context menu.
5.3.1.5 Features Menu Section

The Features section of the element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Open the Attributes dialog.</td>
</tr>
<tr>
<td>Operations</td>
<td>Open the Operations dialog.</td>
</tr>
<tr>
<td>Feature Visibility [Ctrl]+[Shift]+[Y]</td>
<td>Open the Feature Visibility dialog.</td>
</tr>
</tbody>
</table>

Note:

Not all menu options shown here are present on all element context menus. Context menus vary slightly between element types. The Attributes and Operations options won't display for an Action element, for example.

5.3.1.6 Code Engineering Menu Section

The Code Engineering submenu on the element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Source Code [F12]</td>
<td>Open the source editor if a file exists for that selected element.</td>
</tr>
<tr>
<td>Create Workbench Instance [Ctrl]+[Shift]+[J]</td>
<td>Create a workbench instance for the Debug Workbench (if a debug command has been configured for the parent package).</td>
</tr>
</tbody>
</table>

Note:

Not all menu options shown here are present on all element context menus. Context menus vary slightly between element types. These Code Engineering options won't appear for a Use Case element, for example.

5.3.1.7 Appearance Menu Section

The Appearance section of the element context menu can contain the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock Element</td>
<td>Lock the element so it can’t be edited. To unlock the element, select Lock Element again.</td>
</tr>
</tbody>
</table>

Note:

This does not apply in the Corporate edition when security is enabled; in that situation, see the Lock model elements topic.

Selectable

Toggle whether the element is selectable or not. If an element is selectable, you can move it around the diagram and perform right-click operations. If an element is unselectable, you cannot move it around the diagram and the only right-click operation available is to make the element selectable. This option has no effect on double-click operations on the element, such as displaying child diagrams or Properties dialogs.

Appearance

Display the Appearance submenu; see the table below.

Z-Order

Set the Z-Order of the element.
What Is UML Modeling?  | Work With Elements

Note:
You can also change the appearance (and other aspects) of several selected elements at once.

Appearance Sub-Menu

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Appearance [F4]</td>
<td>Override the element’s default appearance (which you set for all elements globally on the Options dialog, Standard Colors page - select the Tools</td>
</tr>
<tr>
<td>Apply Image From Clipboard</td>
<td>Paste the image held on the clipboard onto the selected element.</td>
</tr>
<tr>
<td>Alternate Image [Ctrl]+[Shift]+[W]</td>
<td>Select an alternative image using the image manager.</td>
</tr>
<tr>
<td>Set Font</td>
<td>Change the font type, size, color and effects for the text in an element.</td>
</tr>
<tr>
<td>Copy Appearance to Painter</td>
<td>Copy the default element appearance (set using the Default Appearance option, above) to the painter. You then paste the default appearance using the Paste Appearance option on the Diagram toolbar.</td>
</tr>
<tr>
<td>Copy Image to Clipboard</td>
<td>Copy the element image to the clipboard.</td>
</tr>
</tbody>
</table>

Note:
Not all menu options shown here are present on all element context menus. Context menus vary slightly between element types. The Alternate Image option won’t display for a Lifeline element, for example.

5.3.1.7.1 Set Element Font

You can change the appearance of the text within an element, by selecting the Appearance | Set Font context menu option for one or more selected elements. The Font dialog displays.

Select the font, style, size, color and effects, and (if necessary) the script type. Click on the OK button to save your changes.
### 5.3.1.8 Element Multiple Selection Menu

You can perform operations on two or more elements on a diagram at once. To select the required elements, either click and drag the cursor over the group to highlight them, or press [Shift] and click on each element. Right-click on an element to display the following context menu:

<table>
<thead>
<tr>
<th>Add Ins</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Align Left</td>
<td>Ctrl+Alt+Left</td>
</tr>
<tr>
<td>Align Right</td>
<td>Ctrl+Alt+Right</td>
</tr>
<tr>
<td>Align Top</td>
<td>Ctrl+Alt+Up</td>
</tr>
<tr>
<td>Align Bottom</td>
<td>Ctrl+Alt+Down</td>
</tr>
<tr>
<td>Align Centres</td>
<td></td>
</tr>
<tr>
<td>Space Evenly</td>
<td></td>
</tr>
<tr>
<td>Same Height</td>
<td></td>
</tr>
<tr>
<td>Same Width</td>
<td></td>
</tr>
<tr>
<td>Same Height and Width</td>
<td></td>
</tr>
<tr>
<td>Feature Visibility...</td>
<td>Ctrl+Shift+Y</td>
</tr>
<tr>
<td>Add Tagged Value...</td>
<td></td>
</tr>
<tr>
<td>Autosize</td>
<td>Alt+Z</td>
</tr>
<tr>
<td>Appearance</td>
<td></td>
</tr>
<tr>
<td>Make Non Selectable</td>
<td></td>
</tr>
<tr>
<td>Transform...</td>
<td>Ctrl+H</td>
</tr>
<tr>
<td>Copy</td>
<td></td>
</tr>
<tr>
<td>Layout Selected Elements</td>
<td></td>
</tr>
<tr>
<td>Delete Selected Elements</td>
<td>Ctrl+D</td>
</tr>
</tbody>
</table>

This menu enables you to do the following:

**Note:**

Where elements are made the same, they are matched to the element you right-clicked on.

- Align elements (by left edge, right edge, top, bottom, center in a column or center in a row)
- Space elements evenly (across or down)
- Standardize the dimensions of the selected elements
- Specify the visibility of features for all selected elements
- Add the same Tagged Value to all selected elements
- Automatically resize elements to match (element content permitting)
- Set the default appearance and font 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 the Make All Elements Selectable option
- Generate code for all selected elements at once, or synchronize the code against the selected elements
- Transform the selected elements
- Copy all selected elements to the clipboard
- Automatically adjust the layout of the selected elements on the diagram
• Delete all selected elements.

**Tip:**
It is much faster to assign an appearance or characteristic to a group of elements than to one element at a time.
5.3.2 Element Tasks

This topic describes the following common UML tasks that you can perform on elements in Enterprise Architect:

- **Create Elements**
- **Add Elements Directly to Packages**
- **Use Auto Naming and Auto Counters**
- **Set Element Parent**
- **Show Element Usage**
- **Set Up Cross References**
- **Move Elements Between Packages**
- **Move Elements Within Diagrams**
- **Change Element Type**
- **Align Elements**
- **Resize Elements**
- **Delete Elements**
- **Customize Visible Elements**
- **Create Notes and Text**
- **Set an Element's Default Appearance**
- **Get/Set Project Custom Colors**
- **Use Element Templates**
- **Highlight Context Element**
- **Make Linked Element a Local Copy**
- **Copy Features (Attributes and Operations) Between Elements**
- **Move Features Between Elements**

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Update Element** permission to update element properties or delete an element.

5.3.2.1 Create 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.

Create Elements on a Diagram

The fastest and simplest ways to create elements directly on a diagram are via the Quick Linker and the Enterprise Architect UML Toolbox. The following topics describe these and other approaches for creating elements on a diagram:

- **Create Elements In Place Using the Quick Linker**
- **Create Elements Using the Enterprise Architect UML Toolbox**
- **Create Elements Using the Diagram Context Menu**
- **Create a Group of Elements Using UML Patterns**
- **Create Domain Specific Elements from UML Profiles**

**Tip:**

If you are creating several elements of one type, after creating the first just press **[Shift]+[F3]** or **[Ctrl]+click** to create the next element of that type.

Re-Use Existing Elements

Be aware that once you have created elements, you can re-use them by dragging them from the Project Browser and dropping them onto your diagrams.
Add 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 window and is explained in the following topic:

- Add Elements Directly to a Package

See Also
- Behavioral Diagram Elements - summary of all elements used in Behavioral diagrams
- Structural Diagram Elements - summary of all elements used in Structural diagrams

5.3.2.2 Add Elements Directly To Packages

You can quickly add new elements to a package without the necessity of adding a diagram element at the same time. This is particularly useful in defining a group of Requirements, Changes, Issues, base Classes or other element types that might not require diagrammatic representation in the model.

To add a new element to a package, follow the steps below:

1. In the Project Browser, right-click on the appropriate package. The context menu displays.
2. Select the Add | Add Element menu option. The New Element dialog displays.
3. In the Type field, click on the drop-down arrow and select the element type.
4. In the Name field, type the name of the element.
5. If required, in the Stereotype field either type the stereotype name or click on the drop-down arrow and select the stereotype.
6. Select the Open Properties Dialog on Creation checkbox if the Properties dialog is to open immediately after the element is created.
7. Deselect the Close dialog on OK checkbox to add multiple elements in one session.
8. Click on the OK button to create the element.

Note:
If you have a diagram open at the time, the element is added to the diagram as well. If you do not want the element in that diagram, click on the element in the diagram and press [Delete].

5.3.2.3 Use Auto Naming and Auto Counters

The Auto Element Naming dialog enables 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 the steps below:

1. Select the Settings | Auto Name Counters option from the main menu. The Auto Name Counters dialog displays.
2. In the Type field, click on the drop-down arrow and select the element type (e.g. Activity).
3. In the Name panel:
   - In the Prefix field, type a prefix for the new name (optional).
   - In Counter field, type the counter value; use a many 0’s as required to pad the name.
   - In the Suffix field, type a suffix for the new name (optional).
   - If required, click on the Active checkbox to turn auto naming on for this element type.
4. In the Alias panel:
   - In the Prefix field, type a prefix for the new alias (optional).
   - In Counter field, type the counter value; use a many 0’s as required to pad the alias.
   - In the Suffix field, type a suffix for the new alias (optional).
   - If required, click on the Active checkbox to turn alias auto naming on for this element type.
5. Click on the Save button.

New elements of this type now have an automatically-generated name and/or alias with an incrementing counter value.

Note:
If an Alias is active then auto naming applies; however, to view the Alias in a diagram requires that the option Use Alias if Available is selected in Diagram Properties.

5.3.2.4 Set Element Parent
You can manually set an element's parent or an interface it realizes, using the Type Hierarchy dialog.

To set the element parent, follow the steps below:

1. Select a generalizable element in a diagram.
2. Select the Element | Advanced | Set Parents and Interfaces menu option. Alternatively:
   - Press [Ctrl]+[I] or
   - Right-click and select the Advanced | Parent context menu option.

The Set Parents and Interfaces dialog displays.
3. You can elect to enter a parent or interface name by either manually typing it in, or clicking on the Choose button to locate the element within the current model.

4. Set the Type of relationship (Implements or Generalizes) from the drop-down list.

5. Click on the Add button to add the relationship.

6. Click on the Delete Selected button to remove the current selected relationship.

**Note:**

If Parents are not in the same diagram as their corresponding related element, the parentage is shown in the top right corner of the child element, as shown below:

5.3.2.5 **Show Element Usage**

You can display the usage of an element using the Element Usage dialog. This lists all occurrences of the element throughout the model, and enables you to easily navigate to any occurrence.

To show element usage, follow the steps below:

1. Select an element in a diagram.

2. Select the Element | Find in Diagrams menu option. Alternatively, press [Ctrl]+[U]. If the element exists in other diagrams, the Element Usage dialog displays, listing all occurrences of the current element in the model.
3. Double-click a line item to open the relevant diagram and display the selected element.

**Note:**
You can also access this feature from the Project Browser; select an element in the tree and select the Element | Find in Diagrams menu option. If there is only one instance of the element in any diagram, that diagram displays instead.

### 5.3.2.6 Set Up Cross References

It is possible to set up a cross reference (or Custom Reference) from one element in Enterprise Architect to another. Conversely, you can view existing cross references on an element, using the Hierarchy window.

To set up a cross reference, follow the steps below:

1. In the Project Browser, locate the target element or diagram (that is, the subject of the cross reference).
2. Open a diagram that contains the element(s) that are to have the currently selected element as a reference.
3. Right-click on the element. The context menu displays.
4. Select the Add custom reference... menu option.
5. In the Set up references dialog, select the checkbox against each element to include in the explorer as a reference.
6. Optionally, in the Comment field, type some text 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. Select the **Element** | **Custom References** menu option. Alternatively, either press [Ctrl]+[J], or right-click on the element and select the **Find** | **Custom References** context menu option.
3. The **Custom References** dialog displays, showing a list of elements that have been set as cross references.

   ![Custom References dialog](image)

4. You can open a selected element by highlighting it and clicking on the **Open** button.
5. If you have a diagram cross reference, you can open that diagram.
6. If you have a string of diagram links, click on the **Home** button to return to the original diagram.

**Note:**
You can delete a cross reference by selecting it on the **Custom References** dialog and clicking on the **Delete** button. Cross references are also automatically deleted if the source or target element in the reference is deleted.

### 5.3.2.7 Move Elements Within Diagrams
Any one of the following options enables 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 the element to the required position (the cursor switches to the four-arrow icon as shown below)

![Four-arrow icon](image)
- Hold down [Shift] and use the arrow keys to move the element to the required position
- Use the **Left**, **Right**, **Up** and **Down** options in the **Element** | **Move** submenu
- Align multiple elements using the **Element** | **Alignment** submenu, the **Alignment** options in the right-click context menu, or the **Alignment** buttons on the **Diagram** toolbar

![Diagram toolbar](image)
Confirm Possible Parent Elements

As you organize the elements within a diagram, you can drag any element over another and, provided the dragged element is within and on top of the possible parent, it is always encapsulated by the lower element and moved within the lower element. However, the lower element might not be a valid parent.

You can confirm that a possible parent element is able to accept a selected child element. When you drag the child element over the potential parent, the target element border changes to a dashed line if it can accept the selected element as a child. If the border does not change, the selected element cannot be a child to the target element.

Notes:

- The Support for Composite Objects checkbox must be selected on the Objects page of the Options dialog (select the Tools | Options | Objects option). If this option is not selected, the dashed border does not show and the child element cannot be embedded on the parent in the diagram.
- Both elements must already exist on the diagram; an element border does not change if you drag a potential child element over it from the Toolbox or Project Browser.
- The child element must have equal or higher z-order placement than the parent; that is, the parent element must be level with or behind the child.
- The child element borders must be completely within the parent element borders.

For example, if you drag a Signal over a Class, the Class border changes; a Class element can be a parent to a Signal. If you drag a Class element over a Signal element, the Signal border does not change. A Signal cannot be a parent to a Class.

When you embed a child element on its parent, the child element becomes part of the parent element hierarchy in the Project Browser. Similarly, if you drag the child element out of the parent, the child element becomes independent and is no longer embedded in the parent element hierarchy.

5.3.2.8 Move Elements Between Packages

Elements and packages can be moved from one package to another by dragging and dropping the element to the target destination in the Project Browser. Note that if you move a package, ALL the child packages and their contents are moved to the new location also.

To move an element between packages, follow the steps below:

1. Click on the element in the Project Browser.
2. Drag the element so that the cursor is over the target package icon.
What Is UML Modeling? | Work With Elements

3. Release the mouse button. The element is automatically moved.

Tip:
You can also drag the element under a host element in the new package; for example, drag an element under a Class.

Notes:
- Moving an element has no effect on any relationships that the element might have.
- Moving an element in the Project Browser has no effect on the use of that element in any diagram.
- Moving a diagram generally does not affect the location of elements in packages. If you move a diagram out of one package into another, all the elements in the diagram remain in the original package. However, certain elements (such as Decision, Initial and Final elements) are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram. Therefore, if you move a diagram containing these elements, they are moved to the new parent package with the diagram.

Warning:
In a multi-user environment, if one person moves or updates the Project Browser structure, other users must 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.

5.3.2.9 Change Element Type

To change an element type, follow the steps below:

1. In the Diagram view, click on the element to change.
2. Select the Element | Advanced | Change Type menu option. The Select Element Type dialog displays.
3. In the **Element Type** field, click on the drop-down arrow and select the required element type.
4. Click on the **OK** button.
   
   The target is transformed into the required type.

### 5.3.2.10 Align 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 them one by one by holding down [Ctrl] and clicking on each element).
2. Right-click on the element in the group to align others to. The context menu displays.
3. Select the alignment function you require.

All selected elements are aligned to the one beneath the cursor.

**Tip:**

You can also use the Diagram toolbar.

![Diagram toolbar](image)

The first four buttons are used to align elements, and are made available when more than one element is selected in a diagram.

You can also select the **Element | Alignment** menu option.

### 5.3.2.11 Resize Elements

Any one of the following options enables you to resize an element. Select an element or group of elements in the diagram view, then:

- Use the resize handles that appear at each corner and side to resize the element(s) by dragging with the mouse (the cursor switches to the double-ended arrow as shown below)
• Press and hold [Ctrl] and use the arrow keys to resize as required
• Use the Wider, Narrower, Taller and Shorter options in the Element | Move submenu
• Autosize selected element(s) using the option in the Element | Appearance submenu, or by pressing [Alt] +[Z]. (With multiple elements selected, Autosize also appears in the right-click context menu)
• 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.

Resize a Set of Objects to a Specific Size
If you right-click a selected set of objects, you can resize them to the same dimensions (height, width or both). When you select multiple elements using [Ctrl]+click, then resize the dimensions, the dimensions of the selected hatched object are used to set the dimensions of the other selected objects.

For example, in the diagram below, the Message Class height and width are 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 required size (e.g. Message as above).
2. Select all other elements (e.g. Account and Message Folder as above).
3. Right-click on the pre-sized element (e.g. Message).
4. Select your resizing option (such as same height, width) from the context menu.
5.3.2.12 Delete Elements

Delete an Element From a Diagram

Follow the steps below:
1. In the active diagram, click on the element to delete.
2. Either:
   - Press [Delete], or
   - Right-click to display the context menu and select the Delete <element name> option.

Note:
This does not delete the element from the model, only from the current diagram.

Delete an Element From the Model

Follow the steps below:
1. In the Project Browser, right-click on the element to delete. The context menu displays.
2. Select the Delete <element name> option. A confirmation prompt displays.
3. Click on the Yes button.
   Alternatively:
   1. Click on the element in the Project Browser and press [Ctrl]+[Delete].

The element is completely removed from the model.

Delete Multiple Elements From a Diagram

Follow the steps below:
1. In the active diagram, [Ctrl]+click on each element to delete.
2. Either:
   - Press [Delete], or
   - Right-click to display the context menu and select the Delete selected elements option.

Delete Multiple Elements From a Diagram and Model

Follow the steps below:
1. Open the diagram containing the elements to remove from the model.
2. Press [Ctrl]+[A] to select all of the elements in the diagram, or use [Ctrl]+click to select specific elements.
3. Press [Ctrl]+[Delete] to completely remove the elements from the model.

Delete Multiple Elements from the Project Browser and Model

Follow the steps below:
1. In the Project Browser, press and hold either [Shift] or [Ctrl] and click on the required items.
2. Press [Ctrl]+[Delete] to completely remove the elements from the model.

Note:
If you delete an element in this way, you delete all its properties and connectors as well.
5.3.2.13 Customize Visibility of Elements

Some elements are hidden from view in packages and in RTF documents by default. These include 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.

To show additional elements, follow the steps below:

1. Select the **Tools | Options | Objects** menu option. The **Objects** page of the **Options** dialog displays.
2. Click on the **Advanced** button. The **Advanced Settings** dialog displays.
3. Select the checkbox for each type of element to show in packages and in RTF documents.
4. Click on the **Close** button on each dialog.
5. **Reload** the current diagram if required.

The package from the example above now shows the Event and Decision elements it contains:

---

5.3.2.14 Create Notes and Text

You can create both notes and text on a diagram; the two are slightly different.

Create a Note

To create a note, follow the steps below:

1. Select the **New Note** icon (the text page) on the **UML Elements** toolbar and click on the diagram.
   - If you have the **Edit Object On New** checkbox deselected, the Note element displays on your
What Is UML Modeling? | Work With Elements

To create text, follow the steps below:

1. Click on the New Text Element icon (the letter A) in the UML Elements toolbar, and click on the diagram. The Notes window displays.

2. Type your text, then click on the OK button to save it. Your text displays on the diagram in the following format, with no border:

Note:
You can also create a note by dragging the Note icon from the Common page of the Enterprise Architect UML Toolbox.
5.3.2.15 Set an Element’s Default Appearance

You can set the global appearance of all elements throughout a model, using the Options dialog. Select the Tools | Options menu option, then select Standard Colors from the options tree.

To override the default appearance of a specific element on all diagrams on which it is found, right-click on the element and select the Appearance | Default Appearance menu option. The Default Appearance dialog displays.

To change the appearance of an element on the current diagram only, use the Format toolbar. If the Format toolbar is not displayed, select the View | Toolbars | Format Tool menu option.

**Note:**

You can adjust several elements at the same time. Select all of the required elements, right-click on one of them and select the Default Appearance menu option, or use the Format toolbar.

---

**Change a Background, Font or Border Color**

To reset the background color, font color or border color, follow the steps below:

1. On the Default Appearance dialog, select the Background Color, Font Color or Border Color radio button as appropriate.

**Note:**

You have further options for changing the font of element text in the Set Font menu option.

2. Deselect the Use Default Color checkbox, to enable the Select button.
3. Click on the Select button. The Color dialog displays.
4. Select the required color (click on the Define Custom Colors>> button and define a specific color if necessary) and click on the OK button. You return to the Default Appearance dialog, on which the Preview panel indicates the selected color for the element.

Note:
To change to a different color, click on the Select button again, or to return to the default color, select the Use Default Color checkbox.

5. Click on the OK button. The new color is applied to the selected element or elements.

**Change the Border Thickness**
To change the border thickness, follow the steps below:

1. On the Default Appearance dialog, in the Border Thickness field, type the number of points to apply. Alternatively, click on the scroll arrows to increase or decrease the number.

   The Preview panel indicates the effect of the change in border thickness.
2. Click on the OK button. The new border thickness is applied to the selected element or elements.

**5.3.2.16 Get/Set Project Custom Colors**

If more than one person is working on a project, each person might want to use specific colors for elements within the project. The **Settings | Colors | Set Project Custom Colors** and **Get Project Custom Colors** options enable you to set specific colors and subsequently 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 to be colored.
2. Select the **Element | Appearance | Default Appearance...** menu option. The **Default Appearance** dialog displays.

3. Deselect the **Use Default Color** checkbox to enable the **Select** button.
4. Click on the **Select** button. The **Color** dialog displays.
5. Click on the **Define Custom Colors** button.

6. Create the color in the color mixer panel on the right of the dialog.

7. Click on the **Add to Custom Colors** button to add the color to the **Custom colors** blocks on the left hand side of the dialog.

8. Click on the **OK** button to close the **Color** dialog, then click on the **OK** button to close the **Default Appearance** dialog.

9. Select the **Settings | Colors | Set Project Custom Colors** menu option to save the custom color you have created.

**Get a Project Custom Color**

To get your project's custom colors, follow the steps below.
1. Select the Settings | Colors | Get Project Custom Colors menu option. This applies any saved custom colors to this project.

2. Click on an element to be colored and select the Element | Appearance | Default Appearance menu option. The Default Appearance dialog displays.

3. Deselect the Use Default Color checkbox to enable the Select button.

4. Click on the Select button to view the applied custom colors. They appear as circled below:

5. Click on the required color and click on the OK button to close the Color dialog, then click on the OK button to close the Appearance dialog. The element changes to the required color.

5.3.2.17 Set Element Templates Package

In building up a model, you might want to represent or emphasize certain characteristics of elements in the appearance of those elements, or select a particular display option as the standard. For example, you could make new Interface elements a different default color to new Class elements, ensure all new Activity Partitions are vertical rather than horizontal, or set a specific group of display options for new diagrams. You could also define a set of characteristics to use for each development stage of a project.

To do all this, you create a diagram with all the characteristics you require, and store it in an element Templates package. Enterprise Architect then checks this folder whenever you start to create an element in a diagram and, if it finds a template for that diagram type, applies the settings in that template to the new element or to the display options of the diagram. For example, you could save a diagram under the name ClassTemplate, to apply a set of display characteristics to all new Class elements.

You should create the Templates package in an administrative View of the project file, rather than in any work area. This prevents the package from being changed or lost in any project development work.

There are two other ways in which you can modify the appearance of elements in diagrams:

- You can define the default appearance of elements (and other structures) grouped in a diagram by using UML Profiles. These provide a means of extending the UML Language, which enables you to build UML models in particular domains. They are based on additional stereotypes and Tagged Values that are applied to elements, attributes, methods, connectors, connector ends and so on.

- You can modify the appearance of elements (and connectors) of a specific type using stereotypes. Stereotypes take precedence over templates; if you drop an unsterotyped element - a Class, for example - onto a diagram, Enterprise Architect searches the Templates package for a Class diagram that defines an unsterotyped Class, and applies that definition to the new Class. If you drop a stereotyped Class onto a diagram, the stereotype defines the Class appearance so the template is not accessed. Stereotypes are much more flexible for defining the appearance of an element under different scenarios.

Procedure

To set up the element Templates package, follow the steps below:

1. Create a new package in the appropriate administration View. You can give this package any name;
Templates is an unambiguous option.

2. Within the Templates package create new diagrams, one for each type of diagram to template. Give them easily recognized names; for example ClassTemplate for the template for Class diagrams.

3. Add new elements to the template diagrams from the Enterprise Architect UML Toolbox and configure the size, appearance, notes, version and other properties.

4. Select the Settings | Template Package menu option to set the templates as the default element templates. The Browse Project window displays.

5. Locate and click on the Templates package, and click on the OK button to set the package as the default element template.

Now each new element you add to your project is created with the settings from the appropriate Template diagram.

Note:
If you decide not to use the default element template, set the default element template to <none> in the Browse Project window.

5.3.2.18 Highlight Context Element

You can show a hatched border around a selected element by selecting the Always Highlight Context Element checkbox on the Diagram Behavior page of the Options dialog (select the Tools | Options | Diagram | Behavior menu option). If you have selected this checkbox, the selected element displays similarly to the following example:

If you have not selected the Always Highlight Context Element checkbox, the selected element does not have a hatched border around it.
Multiple Selections

Whether you have selected the **Always Highlight Context Element** checkbox or not, if you select multiple elements one of the elements you select always has a hatched border. If you align the elements, this element is the one used to align the other elements against.

For example, if the elements in the diagram below are aligned, the top element aligns to the bottom element (the element showing a hatched border).

---

**Change the Element to Align Against**

To change which element has a hatched border in a selected group (and thus the element that is aligned with) click on the element that the other elements are to align with.

---

**5.3.2.19 Make Linked Element a Local Copy**

To convert a linked element to a local copy, follow the steps detailed below:

1. Open the diagram with the linked element.
2. Select the linked element and right-click on it to display its context menu.
3. Select the **Convert Linked Element to Local Copy** menu option.

The element changes to a local copy and is placed in the appropriate package.

---

**5.3.2.20 Copy Features Between Elements**

Using drag and drop, you can copy attributes and/or operations from an element in the Project Browser on to another element in a diagram.

To move attributes and operations, see **Move Attributes and Operations Between Elements**.

---

**Copy an Element Feature**

To copy an element feature, follow the steps below:

1. Open a diagram that contains the target element (in the example below, the AccountItem Class is the target and Customer element is the donor).
2. 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.

Copy Multiple Element Features
To copy multiple element features, follow the steps below:

1. Open a diagram that contains the target element (in the example above, the AccountItem Class is the target and Customer element is the donor).
2. Hold down [Ctrl] (separate features) or [Shift] (select a range) and click on the attributes and/or operations to copy, then drag the selected features to the target element.
3. Release the mouse button.

5.3.2.21 Move Features Between Elements
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.

To copy attributes and operations, see Copy Attributes and Operations Between Elements.

To move element features, follow the steps below:

1. In the Project Browser, locate the attributes and/or operations to move from the target element and select them while holding down [Ctrl] (single item select) or [Shift] (multiple item select).
2. Holding down the mouse button, drag the attributes and/or operations to the target element. Only the last selected element feature displays during the move; however all of the selected features are moved.

3. Release the mouse button. The image below shows the final stage of the attribute and operations move between the Account element and the Class element.
5.3.3 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* can be attributes. Attributes have several important characteristics, such as type, scope (visibility), static, derived and notes.

![Customer Class Diagram]

### Create and Modify Element Attributes

**Note:**

This facility is only available if the element supports attributes.

1. In the **Diagram** view, either:
   - Right-click on the element to be edited, and from the context menu select the Attributes menu option
   - Click on the element and press **[F9]**, or
   - Drag the attribute from the Project Browser onto the element.
What Is UML Modeling?  | Work With Elements

2. The `<Element name> Attributes` dialog displays.
What Is UML Modeling? | Work With Elements

Notes:

- If you make changes and do not save them, the Cancel button prompts you to save or cancel the changes, whilst the Close button closes the dialog immediately and does not save the changes.

- If you are creating many attributes, go to the Attribute/Operations page of the Options dialog (Tools | Options | Source Code Engineering | Attribute/Operations) and select the After save, re-select edited item checkbox. Now, when you create an attribute and click on the Save button, the dialog fields clear ready for you to enter the details of the next attribute. This helps you when you want to create attributes quickly and might not necessarily want to fully define each one as you create it.

See the topics on the Attributes dialog General, Detail and Constraints tabs.

See Also

- Attribute Tagged Values
- Create Properties
- Display Inherited Attributes
- Create Object From Attribute

5.3.3.1 Attributes Dialog - General Tab

The General tab of the Attributes dialog is shown below:

To review an existing attribute, click on the attribute name in the Attributes panel.

To delete an existing attribute, click on the attribute name in the Attributes panel and click on the Delete button.

To create a new attribute, either:

- Click on the New button, or
- Click on an existing attribute name in the Attributes panel, and click on the Copy button.

Review, edit or complete the fields as indicated in the following table.
<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Display the name of the attribute. For a new attribute, type the name (with no spaces).</td>
</tr>
<tr>
<td>Alias</td>
<td>Display an optional alias for the attribute. If necessary, type in a new alias.</td>
</tr>
<tr>
<td>Type</td>
<td>Display the attribute's data type. If necessary, click on the drop-down arrow and select a different type.</td>
</tr>
<tr>
<td>(Select) button</td>
<td>Open the Select Attribute Type dialog, which you use to select or define a different attribute type.</td>
</tr>
<tr>
<td>Scope</td>
<td>Define the attribute as Public, Protected, Private or Package. If necessary, click on the drop-down arrow and select a different scope.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Define the optional stereotype of the attribute. If necessary, either type a different stereotype name or click on the drop-down arrow and select a stereotype.</td>
</tr>
<tr>
<td>Containment</td>
<td>Define the containment type (by reference, by value or not specified). If necessary, click on the drop-down arrow and select a different containment type.</td>
</tr>
<tr>
<td>Derived</td>
<td>Indicate that the attribute is a calculated value.</td>
</tr>
<tr>
<td>Static</td>
<td>Indicate that the attribute is a static member.</td>
</tr>
<tr>
<td>Property</td>
<td>Indicate that the attribute has automatic property creation.</td>
</tr>
<tr>
<td>Const</td>
<td>Indicate that the attribute is a constant.</td>
</tr>
<tr>
<td>Is Literal</td>
<td>(For Enumeration elements) Defaults to selected, to define the attribute as an enumeration literal. Deselect to define the attribute as a normal element attribute. In the Attributes compartment on the diagram, the enumeration literals are listed separately, above the normal attributes. (Ensure that the Stereotype field for the normal attribute is not set to enum.)</td>
</tr>
<tr>
<td>Initial</td>
<td>Display an optional initial value. If necessary, type in a new initial value.</td>
</tr>
<tr>
<td>Notes</td>
<td>Enter any free text notes associated with the attribute. You can format the notes text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
</tbody>
</table>

To change the position of an attribute in the list in the Attributes panel, click on the Scroll Up or Scroll Down (hand) buttons.

**Note:**

By default, the attributes are listed in alphabetical order. Before changing this sequence, you must deselect the Sort Features Alphabetically checkbox on the Objects page of the Options dialog (Tools | Options | Objects).

If you have changed the attribute details, click on the Save button to save the changes.
5.3.3.2 Attributes Dialog - Detail

To define additional details relating to collections, click on the Detail tab of the Attributes dialog.

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplicity</td>
<td></td>
</tr>
<tr>
<td>Lower bound</td>
<td>Define a lower limit to the number of elements allowed in the collection.</td>
</tr>
<tr>
<td>Upper bound</td>
<td>Define an upper limit to the number of elements allowed in the collection.</td>
</tr>
<tr>
<td>Ordered Multiplicity</td>
<td>Indicate that the collection is ordered.</td>
</tr>
<tr>
<td>Collection</td>
<td>Code the attribute as an array, so that it can contain multiple concurrent values rather than a single value.</td>
</tr>
<tr>
<td>Attribute is a Collection</td>
<td>Indicate that the attribute is a collection (array).</td>
</tr>
<tr>
<td>Allow Duplicates</td>
<td>Indicate that duplicates are allowed. Maps to the UML property isUnique, value FALSE).</td>
</tr>
<tr>
<td>Container Type</td>
<td>Enter the name of the container type.</td>
</tr>
<tr>
<td>Transient</td>
<td>(For Java code) indicate that the attribute can change regardless of what the code is performing.</td>
</tr>
</tbody>
</table>

When you have completed these fields, click on the Save button.
5.3.3.3 Attributes Dialog - Constraints

Attributes can also have constraints associated with them. Typically these indicate such things as maximum value, minimum value and length of field.

Select the Constraints tab of the Attributes dialog to define these constraints.

To review an existing constraint, click on the constraint name in the panel at the bottom of the dialog.
To delete an existing constraint, click on the constraint name in the panel and click on the Delete button.
To create a new constraint, click on the New button.
Review, edit or complete the fields as indicated in the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint</td>
<td>Type the constraint name.</td>
</tr>
<tr>
<td>Type</td>
<td>Select the constraint type.</td>
</tr>
<tr>
<td>(Notes)</td>
<td>Type any comments or notes concerning the constraint.</td>
</tr>
</tbody>
</table>

If you have created or edited the data, click on the Save button to save the changes.

5.3.3.4 Attribute Tagged Values

You can define Tagged Values for an attribute. 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.

Note:
Tagged Values are supported for attributes, operations, objects and connectors.
Add a Tagged Value

To add a Tagged Value to an attribute, follow the steps below:

1. Either:
   - Select the View | Tagged Values menu option or
   - Press [Ctrl]+[Shift]+[6].
   The Tagged Values window displays.

2. Double-click on the attribute, either in the diagram or in the Project Browser. The attribute name is displayed as selected in the Tagged Values window.

3. Either click on the New tag button ( ) or press [Ctrl]+[N]. The Tagged Value dialog displays.

   ![Tagged Value Dialog](image)

4. In the Tag field, type the tag name or click on the drop-down arrow and select a custom defined tag.

5. In the Value field, type the text associated with the tag.

6. Click on the OK button to confirm the operation. The tag name and value are displayed under the attribute in the Tagged Values window.

Note:
You can define custom tags by creating a Custom Tagged Value Type. For more information see [SDK for Enterprise Architect](https://www.sparxsystems.com/en/download-sdk).
5.3.3.5 Create Properties

Enterprise Architect has capabilities for automatically creating properties in various languages. Property creation is controlled from the General tab of the Attribute dialog.

Select the Property checkbox. The Create Property Implementation dialog immediately displays.

The Language panel defaults to the Class language; however, you can change this and generate the properties 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
- Delphi creates get and set functions as well as a specialized Delphi property Tagged Value.

Type in the required details and click on the OK button. Enterprise Architect generates the required operations and properties to comply with the selected language.

Note that get and set functions are stereotypes with «property get» «property set» making it easy to recognize property functions. You can also hide these specialized functions by deselecting the Property Methods checkbox in the Features tab of the Diagram Properties dialog for a specific diagram (select the Diagram | Properties menu option). This makes it easier to view a Class, uncluttered by many get and set methods.
Note that for Delphi you must enable the Tagged Values compartment to see the generated properties. See Compartments for the steps for doing this.
5.3.3.6 **Display Inherited Attributes**

When displaying a Class with attributes in a diagram, you can 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 for elements that have attributes, you can also override an inherited attribute's initial value, using the element context menu option Advanced | Override Attribute Initializers. This displays the Override Attribute Initializers dialog.
In the Override Attribute Initializers dialog, select the variable name and enter a new initial value. If required, you can type a note in the Note field. When you display inherited attributes, Enterprise Architect merges the list of attributes from all ancestors and merges the attribute initializers, so that the final child Class displays the correct attribute set and initial values.

5.3.3.7 Create Object From Attribute

If you drag an attribute from the Project Browser onto an Activity diagram, the attribute generates an Object element of the same name. This is very useful for creating connectors between elements and specific attributes. For example, a Class element of stereotype table defines its fields as attributes; Requirement elements that define requirements for those fields can then be linked to the appropriate table fields via the attribute Object elements.

In the following diagram, the billingAddress Object was generated by dragging the BillingAddress attribute from the Account Class in the Project Browser onto the diagram. Realize relationships were then created between the Account element and the billingAddress element, and between the billingAddress element and the REQ028 Requirement element.
5.3.4 Operations

Operations are features of a Class or other element that represent the behavior or services an element supports. For a Customer Class, \texttt{UpdateCustomerName} and \texttt{GetCustomerAddress} can 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 right-click context menu contains the Operations menu item. Select this to open the Operations dialog. Alternatively, press [F10].

How Operations Appear in Diagrams

Elements with operations (typically Classes) display their features in diagrams in the manner shown in the following diagram. As the diagram illustrates, some characteristics display in shorthand form; for example, static displays as $\$, abstract as $\ast$.

Operations in the Project Browser

Classes with operations 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 them the same operations.
5.3.4.1 Operations Dialog - General

The Operations dialog has four tabs:

- General, from which you can also define operation parameters.
- Behavior
- Pre-conditions and Post conditions (that is, Constraints).

The General tab of the Operations dialog enables you to define new operations and set the most common properties, including name, access type and return.

Note:

The General tab can vary according to the type of element you are adding an operation to. If defining operations for a data modeling table, see the Indexes, Triggers and Check Constraints topic. The following illustrations are for the operations of an Object element and a State element.
What Is UML Modeling? | Work With Elements

**Option**  | **Use to**
--- | ---
Name | Display the selected operation name.
Parameters | Display the parameter list. See Operation Parameters for information regarding what this string can contain.
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Parameters</td>
<td>Open the Parameters dialog.</td>
</tr>
<tr>
<td>Return Type</td>
<td>Display the data type returned by the operation.</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>[ ... ] (Return Type</td>
<td>Open the Set Element Classifier dialog.</td>
</tr>
<tr>
<td>Build button)</td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Action</td>
<td>Define the action of the operation: do, exit or entry.</td>
</tr>
<tr>
<td></td>
<td>(For State or State Machine elements.)</td>
</tr>
<tr>
<td>Scope</td>
<td>Select Public/Protected/Private/Package.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>Specify an optional stereotype for this operation.</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Set the concurrency of the operation.</td>
</tr>
<tr>
<td>Alias</td>
<td>Define an optional alias for the operation.</td>
</tr>
<tr>
<td>Notes</td>
<td>Enter free text notes. You can format this text if necessary, using the</td>
</tr>
<tr>
<td></td>
<td>Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Virtual/Abstract</td>
<td>If the operation's language is set to C++, map to the C++ Virtual</td>
</tr>
<tr>
<td></td>
<td>keyword. Otherwise this option is Abstract, pertaining to an abstract</td>
</tr>
<tr>
<td></td>
<td>function. (Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Return Array</td>
<td>Indicate that the return value is an array.</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Synchronized</td>
<td>Specify a code engineering flag that relates to multi threading in Java.</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Static</td>
<td>Indicate that the operation is a static member.</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Const</td>
<td>Indicate that the return type of this method is constant.</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Pure</td>
<td>Indicate that C++ is pure virtual syntax - e.g.</td>
</tr>
<tr>
<td></td>
<td>virtual void myFunction( ) = 0;</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>IsQuery</td>
<td>Indicate that this method does not modify the object.</td>
</tr>
<tr>
<td></td>
<td>(Not shown for State or State Machine elements.)</td>
</tr>
<tr>
<td>Operations</td>
<td>List the defined operations.</td>
</tr>
<tr>
<td>Up/Down Buttons</td>
<td>Change the order of operations in the list.</td>
</tr>
<tr>
<td>New</td>
<td>Create a new operation.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy the currently selected operation.</td>
</tr>
<tr>
<td>Save</td>
<td>Save a new operation, or save modified details for existing operation.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the currently selected operation.</td>
</tr>
</tbody>
</table>
Note:

- If you make changes and do not save them, the Cancel button prompts you to confirm or cancel the changes, whilst the Close button closes the dialog immediately and does not save the changes.
- If you are creating many operations, go to the Attribute/Operations page of the Options dialog (Tools | Options | Source Code Engineering | Attribute/Operations) and select the After save, re-select edited item checkbox. Now, when you create an operation and click on the Save button, the dialog fields clear ready for you to enter the details of the next operation. This helps you when you want to create operations quickly and might not necessarily want to fully define each one as you create it.

### 5.3.4.1.1 Operation Parameters

The Parameters dialog enables you to define the parameters an operation has. The parameter list is reproduced in code in the order the parameters appear in the parameters list, so use the Up and Down buttons to move parameters into their required positions. Additionally, you can select the Add new to end checkbox to force new parameters to appear at the end of the list instead of the top.

**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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type the parameter name.</td>
</tr>
<tr>
<td>Type</td>
<td>Select the data type of the parameter. Alternatively, click on the [...] button and select the element classifier to define the type.</td>
</tr>
<tr>
<td>Default</td>
<td>Type an optional default value for the parameter.</td>
</tr>
</tbody>
</table>
What Is UML Modeling? | Work With Elements

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stereotype</td>
<td>Type a stereotype name, or click on the drop-down arrow and select a stereotype for the parameter.</td>
</tr>
<tr>
<td>Kind</td>
<td>Indicate the way a parameter is passed to a function:</td>
</tr>
<tr>
<td></td>
<td>- <em>In</em> = By Value</td>
</tr>
<tr>
<td></td>
<td>- <em>InOut</em> = By Reference</td>
</tr>
<tr>
<td></td>
<td>- <em>Out</em> is passed by Reference, but only the return value is significant.</td>
</tr>
<tr>
<td>Fixed</td>
<td>Set the parameter to <em>const</em>, even if passed by reference.</td>
</tr>
<tr>
<td>Alias</td>
<td>Type an optional alias for the parameter.</td>
</tr>
<tr>
<td>Add new to end</td>
<td>Place new parameters at the end of the list instead of the start.</td>
</tr>
<tr>
<td>Notes</td>
<td>Type any additional notes on the parameter.</td>
</tr>
</tbody>
</table>

**See Also**
- [Operation Parameter Tagged Values](#)
- [Operation Parameters by Reference](#)

### 5.3.4.1.1 Operation Parameter Tagged Values

Operation parameters can 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 values to them using this form.

Tagged Values are written to the XMI output, and can 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 a parameter, follow the steps below:

1. Select the View | Tagged Values menu option or press [Ctrl]+[Shift]+[6]. The Tagged Values window displays.
2. Click on the operation containing the parameter in a diagram or in the Project Browser. The Tagged Values window now has the operation and parameters selected.
3. Click on the required parameter in the Parameters compartment of the Tagged Values window, and either click on the New Tags button or press [Ctrl]+[N]. The Tagged Value dialog displays.
4. In the **Tag** field, type the tag name (or select a custom-defined tag from the drop-down list), then add the tag value in the **Value** field.
5. Click on the **OK** button to confirm the operation.

**Tip:**
Custom tags can be defined by creating a custom Tagged Value type. For more information see [SDK for Enterprise Architect](#).

### 5.3.4.1.1.2 Operation Parameters by Reference

You can select to highlight parameters declared as ‘Kind: inout’ with an additional user-defined prefix or suffix. On the **Objects** page of the **Options** dialog (select the **Tools | Options | Objects** menu option), the **Feature reference indicator** panel enables you to set whether references are highlighted or not.

If you select the **Highlight References** checkbox, you can also indicate whether a prefix or suffix should be used, and the actual reference character to use. In the example above, the & character has been set as a prefix.

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 is displayed in the **Diagram View**.

The example below shows that, in the `getName` operation, the parameter `strName` is a string reference, and is highlighted using the chosen character and position.

### 5.3.4.1.2 Operations Dialog - Behavior

The **Behavior** tab of the **Operations** dialog enables you to enter free text to describe the functionality that an operation has. Use pseudo-code, structured English or just a brief description.

You can also use this tab 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.
Show Behavior in a Diagram

To show behavior in a diagram, follow the steps below:

1. Create or locate the required operation.
2. Click on the Behavior tab of the Operations dialog.
3. Select the Show Behavior in Diagram checkbox.
4. Click on the Save button.

See Also
- Initial Code
5.3.4.1.2.1 Initial Code

On the Behavior tab of the Operations dialog, you can use the Initial Code field to enter code to be inserted into an operation body when the operation is first generated to file. After this point, forward code generation and synchronization do not replace the existing operation code with the Initial Code field.

By default, the Initial Code field also is not imported into the model during reverse engineering, but you can select to import the field by selecting the Include method bodies in model when reverse engineering checkbox on the Options dialog.

This field is most useful when combined with UML Patterns. Elements within a pattern often require the same stub code. Notice that the language specific patterns available from www.sparxsystems.com/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:

```csharp
if (instance == null)
    instance = new Singleton();
return instance
```

5.3.4.1.3 Operations Dialog - Constraints

Operations can 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.
5.3.4.2 Operation Tagged Values

Operations can 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 values to them using this form.

Tagged Values are written to the XMI output, and can 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, follow the steps below:

1. Select the **View** | **Tagged Values** menu option or press `[Ctrl]+[Shift]+[6]`. The Tagged Values window displays.
2. Click on the operation in a diagram or in the **Project Browser**. The Tagged Values window now has the operation selected.
3. Either click on the **New Tags** button or press `[Ctrl]+[N]`. The Tagged Value dialog displays.
4. In the Tag field, type the tag name (or select a custom-defined tag from the drop-down list), then in the Value field type the tag value.

5. Click on the OK button to confirm the operation.

Note:
You can define custom tags by creating a Custom Tagged Value Type. For more information see SDK for Enterprise Architect.

5.3.4.3 Override Parent Operations

In Enterprise Architect, you can automatically override methods from parent Classes and from realized Interfaces.

Select a Class that has a parent or realized interface and select the Element | Advanced | Overrides & Implementations menu option.

In the Override Operations/Interfaces dialog, check the operations/interfaces to automatically override and click on the OK button. Enterprise Architect generates the equivalent function definitions in your child Class.
You can configure Enterprise Architect to display this dialog each time you add a Generalization or Realization connector between Classes, and review their possible operations/interfaces to override/implement. Do this from the **Links** page of the **Options** dialog (select the **Tools** | **Options** | **Links** menu option).

5.3.4.4 **Display Inherited Operations**

You can configure an element in a diagram to display the complete operation set obtained from all ancestors in the element's type hierarchy, as well as those directly owned. To do this, use the **Element** | **Feature Visibility** function from the main menu, or press **[Ctrl]+[Shift]+[Y]**.

The following diagram illustrates this behavior when enabled for each element in a simple hierarchy.
5.3.5 Element In-place Editing Options

This topic explores the tasks that can be performed using in-place editing of elements on a diagram in Enterprise Architect. The tasks include:

- View Properties
- Edit Element Item Name
- Edit Stereotype
- Edit Scope
- Edit Attribute Keyword
- Edit Operation Parameter Keyword
- Insert Operation Parameter
- Edit Parameter Kind
- Insert New Attribute or Operation
- Add Maintenance Item
- Add Test Item
- Delete Selected from Model

5.3.5.1 In-place Element Item Tasks

To use the options that are available through the in-place editing menu, follow the steps below:

1. Open the diagram containing the element.
2. Click on the element, and on the item to manipulate within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.
3. Edit and manipulate the items in the element, either by pressing the appropriate keyboard keys or by right-clicking on the highlighted item and choosing a task from the Element Items menu. The following commands are available:

<table>
<thead>
<tr>
<th>To...</th>
<th>Select menu option...</th>
<th>Or press...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the name, scope or stereotype of the element or element item</td>
<td>Edit Selected</td>
<td>[F2]</td>
</tr>
<tr>
<td>Display the dialog containing details of the element</td>
<td>View Properties</td>
<td>[Enter]</td>
</tr>
<tr>
<td>Insert a new item in the element</td>
<td>Insert New After Selected</td>
<td>[Insert]</td>
</tr>
<tr>
<td>Add an attribute to the element</td>
<td>Add Attribute</td>
<td>[Ctrl]+[Shift]+[F9]</td>
</tr>
<tr>
<td>Add an operation to the element</td>
<td>Add Operation</td>
<td>[Ctrl]+[Shift]+[F10]</td>
</tr>
<tr>
<td>Insert a feature on the specific element item, such as Maintenance features and Testing features</td>
<td>Add Other</td>
<td>[Ctrl]+[F11]</td>
</tr>
<tr>
<td>Delete the selected item from the model</td>
<td>Delete Selected from Model</td>
<td>[Delete]</td>
</tr>
<tr>
<td>Navigate Diagram Selection, to navigate the diagram between elements without having to use the mouse</td>
<td>[Ctrl]+[Shift]+[arrow key]</td>
<td></td>
</tr>
</tbody>
</table>
What Is UML Modeling? | Work With Elements

<table>
<thead>
<tr>
<th>To...</th>
<th>Select menu option...</th>
<th>Or press...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle element highlight option on and off</td>
<td></td>
<td>[Shift]+[Enter]</td>
</tr>
</tbody>
</table>

Other options that are available while editing element attributes or operations in a diagram include:

<table>
<thead>
<tr>
<th>To...</th>
<th>Press...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept current changes</td>
<td>[Enter]</td>
</tr>
<tr>
<td>Accept current changes and open a new slot to add a new item</td>
<td>[Ctrl]+[Enter]</td>
</tr>
<tr>
<td>Abort the edit, without save</td>
<td>[Esc]</td>
</tr>
<tr>
<td>Display the context menu for in-place editing</td>
<td>[Shift]+[F10]</td>
</tr>
<tr>
<td>Invoke the Classifier dialog</td>
<td>[Ctrl]+[Space]</td>
</tr>
</tbody>
</table>

5.3.5.2 Edit Element Item Name

The in-place editing feature enables you to change the name of the element, or the name of an operation or attribute, directly from the diagram. To use this feature follow the steps below:

1. Open the diagram containing the element.
2. Click on the element and on the name to change within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.

3. Right-click on the item. The context menu displays.
4. Select the Edit Selected menu option (or press [F2]) to enable you to edit the item directly from the diagram. The name of the attribute or operation is highlighted.

5. Delete or type over the name. Press [Enter] to accept the change, or [Esc] to cancel the change.

5.3.5.3 Edit Feature Stereotype

You can use the in-place editing feature to change the stereotype of an operation or attribute directly from the diagram. To use this feature, follow the steps below:

1. Open the diagram containing the element.
2. Click on the element, and on the item to edit within the element. The item line is highlighted in a lighter...
shade (the default is white), to indicate that it has been selected.

3. Right-click on the item. The context menu displays.
4. Select the **Edit Selected** menu option (or press **[F2]**) to enable you to edit the attribute or operation directly from the diagram. The name of the item is highlighted.
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 name to change the stereotype name of the attribute or operation. Press **[Enter]** to accept the change or **[Esc]** to cancel the change. You can assign multiple stereotypes by including a comma-separated list inside the stereotype markers.

### 5.3.5.4 Edit Feature Scope

The in-place editing feature enables you to rapidly change the scope of an attribute or operation directly from the diagram. To use this feature follow the steps below:

1. Open the diagram containing the element.
2. Click on the element and on the item to edit within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.

3. Right-click on the item. The context menu displays.
4. Select the **Edit Selected** menu option (or press **[F2]**) to enable you to edit the attribute or operation directly from the diagram. The name of the item is highlighted.
5. Move the cursor to the scope of the item and delete the previous entry.
6. Reassign the entry by typing in one of the following symbols:
   - + indicates that the scope is Public
   - - indicates that the scope is Private
   - ~ indicates that the scope is Package
   - # indicates that the scope is Protected.

7. Press [Enter] to save the change, or [Esc] to cancel the change. The diagram is updated to reflect the changes. (Also see the catalogNumber attribute in the above screen illustrations.)

5.3.5.5 Edit Attribute Keyword

You can add features such as attribute keywords and classifiers directly to an element, using the Element Keywords and Classifiers menu. This enables you to rapidly assign details element item by element item, directly from a diagram. To use this feature, follow the steps below:

1. In Enterprise Architect, open the diagram containing the element.
2. Click on the element, and on the attribute to edit within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.
3. Right-click on the item. The context menu displays.
4. Select the Edit Selected menu option (or press [F2]) to enable you to edit the attribute directly from the diagram. The name of the attribute is highlighted.
5. Right-click on the attribute name to display the context menu.
6. From the context menu, you can:
   - Change the attribute classifier to static or fixed - select the **static** or **fixed** menu options as appropriate; the diagram is updated to reflect the changes.
   - Display the Class properties - click on the **Goto Definition** menu option; Enterprise Architect locates the Class in the **Project Browser** and opens its **Properties** dialog.

   If the data type is a raw data type, Enterprise Architect displays the message: *The data type is a raw data type.*

### 5.3.5.6 Edit Operation Parameter Keyword

You can directly edit operation classifiers by element, using the in-place editing menu. This enables you to rapidly assign parameter keywords. To use this feature, follow the steps below:

1. Open the diagram containing the element.
2. Click on the element, and on the operation to edit within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.

3. Right-click on the item. The context menu displays.
4. Select the **Edit Selected** menu option (or press **[F2]**) to enable you to edit the operation directly from the diagram. The name of the operation is highlighted.
5. Right-click on the data type of a parameter to display the context menu.

6. From the context menu you can:
   - Change the operation classifier by clicking on the appropriate menu option - **static**, **isquery**, **abstract** or **fixed**. The diagram is updated to reflect the changes.
   - Display the Class properties - click on the **Goto Definition** menu option.

   If the data type is Class, Enterprise Architect locates the Class in the **Project Browser** and opens its **Properties** dialog.

   If the data type is a raw data type, Enterprise Architect displays the message *This data type is a raw data type.*

   If the data type is not defined in the model, the message is: *The data type is not defined in the model.*
5.3.5.7 **Edit Parameter Kind**

You can edit operation parameter kinds such as `[in]`, `[inout]` and `[out]` directly from a diagram element by element, using the **Element Keywords and Classifiers** menu. This enables you to rapidly assign the parameter directly from a diagram. To use this feature follow the steps below:

1. In Enterprise Architect, open the diagram containing the element.
2. Click on the element, and on the operation to edit within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.
3. Right-click on the item. The context menu displays.
4. Select the **Edit Selected** menu option (or press `[F2]`) to enable you to edit the item directly from the diagram. The name of the item is highlighted.
5. Right-click on the item name to display the context menu.
6. Select the appropriate menu option for the parameter kind value: `[in]`, `[inout]` and `[out]`. The diagram is updated to reflect the change.

5.3.5.8 **Insert New Feature**

You can add attributes and operations to an element using the in-place editing options. To add attributes and operations to a Class diagram element, follow the steps below:

1. Open the diagram containing the element to which you are adding an attribute or operation.
2. Click on the element, and within the element on the item after which to insert the operation or attribute. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.
3. Press [Insert]. Alternatively, right-click on the selected element item to display the context menu and select the **Insert New After Selected** menu option.

   Enterprise Architect inserts a new data line in the diagram, underneath the selected item.

4. Type in the relevant information for the attribute or operation. Press [Enter] to accept the change or [Esc] to cancel the change. The diagram is updated to reflect the changes.

### 5.3.5.9 Insert Operation Parameter

You can add operation parameters to an operation through the in-place editing options, using hotkey commands or menu shortcuts. To add parameters to operations in a Class diagram element, follow the steps below:

1. Open the diagram containing the element.
2. Click on the element, and on the operation to update within the element. The item line is highlighted in a lighter shade (the default is white), to indicate that it has been selected.

![Diagram of LineItem class]

3. Press [F2], or right-click on the selected item to display the context menu and select the **Edit Selected** option.

4. Move the cursor inside the parameter brackets and click on the reference to the parameter (e.g. bks for a vector containing books). Either:
   - Type the name of the parameter or
   - Place the cursor after the reference, right-click the mouse to display the inline editing options menu and select the **Insert Classifier** option.

![Set Element Classifier dialog]

The **Set Element Classifier** dialog displays.
5. Click on the appropriate parameter from the list of available items, and click on the OK button. The parameter is displayed on the diagram.
6. Press [Enter] to accept the change or [Esc] to cancel the change. The diagram is updated to reflect the changes.

### 5.3.5.10 Insert Maintenance Feature

You can rapidly assign maintenance details such as Defects, Changes, Issues and Tasks directly to an element from a diagram, using the **Element Items** menu. To use this feature follow the steps below:

1. Open the diagram containing the element.
2. Click on the element name. The name is highlighted in a lighter shade (the default is white), to indicate that it has been selected.

3. Either:
   - Press [Ctrl]+[F11] or
   - Right-click on the highlighted name to display the context menu, and select the **Add Other** option.

   The **Insert Feature** dialog displays.
4. Click on the appropriate radio button option to associate the required maintenance feature with the element item.

5. Click on the OK button. The <Maintenance Feature> details for <element> dialog displays.

6. Complete the fields to define the maintenance activity, and then click on the Save button. To create a subsequent maintenance activity of this type, click on the New button.

7. When you have defined all of the maintenance activities of this type, click on the OK button. The maintenance details are added to the element.

To ensure that the maintenance items are visible in the diagram element, as shown in the example below, select the Show Maintenance checkbox in the diagram properties appearance options. For more information on diagram appearance options, see the Show Maintenance Scripts in Diagram topic.
5.3.5.11 Insert Testing Features

You can rapidly add testing features such as Unit, Integration, System, Acceptance and Scenario tests to an element directly from a diagram, using the Element Items menu. To use this feature follow the steps below:

1. Open the diagram containing the element.
2. Click on the element. The element name is highlighted in a lighter shade (the default is white), to indicate that it has been selected.
3. Either:
   - Press [Ctrl]+[F11] or
   - Right-click on the highlighted name to display the context menu and select the Add Other option.

   The Insert Feature dialog displays.

4. Click on the appropriate radio button option to associate the required testing feature with the element.
5. Click on the OK button. The Testing Window opens, showing the appropriate panel for the type of test selected.
6. **Complete the fields** to define the test activity, and then click on the **Save** icon in the window toolbar. The test is added to the element.

7. To create a subsequent test activity of this type, click on the **New** icon, or to add items for other types of test, click on the appropriate tab.

To ensure that the test items are visible in the diagram element, as shown in the example below, select the **Show Testing** checkbox in the diagram properties appearance options. For more information on diagram appearance options, see the **Show Test Scripts in Compartments** topic.
5.3.6 Properties

This topic area covers element properties and their settings, responsibilities, constraints, connectors, scenarios, Tagged Values, associated files, object files and classifiers, and boundary element settings.

To display the element Properties dialog:
- Select an element in the Diagram View and select the Element | Properties menu option
- Right-click on an element in the Diagram View, and select 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 display of the Properties dialogue when placing a new element, uncheck the Edit Object on New option on the Objects page of the Options dialog (Tools | Options | Objects).

Note:
There are three variations of the Properties dialog:
- The dialog for a Table or Stored Procedure element has slight differences on the General tab, and a Table (Stored Procedure) Details tab instead of a Details tab; see the Set Table Properties topic.
- The dialog for a Class element of a stereotype other than Table is as shown in General Settings.
- The dialog for an element of any other type does not have a Details tab.

The following topics describe each of the tabs in this dialog in detail.
- General
- Details
- Require
- Constraints
- Link
- Scenario
- Files

Follow the links for information on Tagged Values, Object files and Classifiers, and the Boundary element appearance.
### 5.3.6.1 General Settings

The General tab of the element Properties dialog is shown below:

![Element Properties Dialog](image)

Complete the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Change the element's name.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>(Optional) Type the name of a stereotype for the element, or click on the drop-down arrow and select one.</td>
</tr>
<tr>
<td>Abstract</td>
<td>Indicate that the element is abstract.</td>
</tr>
<tr>
<td>Author</td>
<td>Enter or select the name of the original author.</td>
</tr>
<tr>
<td>Status</td>
<td>Indicate the current status of the element (e.g. Approved, Proposed).</td>
</tr>
<tr>
<td>Scope</td>
<td>Indicate the element's scope (public, private, protected, package).</td>
</tr>
<tr>
<td>Complexity</td>
<td>Indicate the complexity of the element (used for project estimation). Assign Easy, Medium or Hard.</td>
</tr>
<tr>
<td>Alias</td>
<td>Enter an alias (alternative display name) for the object.</td>
</tr>
<tr>
<td>Language</td>
<td>Select the programming language for the object.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Enter free-text items such as keywords or context information. This can be filtered in Use Case Metrics and Search dialogs.</td>
</tr>
<tr>
<td>Phase</td>
<td>Indicate the phase this element is to be implemented in (e.g. 1, 1.1, 2.0 ...).</td>
</tr>
</tbody>
</table>

This Class declares an interface for operations that create abstract product objects.
### What Is UML Modeling? | Work With Elements

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Enter the version of the current element.</td>
</tr>
<tr>
<td>Notes</td>
<td>Enter any free text notes associated with the element. You can format the notes text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
</tbody>
</table>

Further facilities are made available by pressing the Advanced button. See Advanced Settings for details.

#### 5.3.6.1.1 Advanced Settings

Some elements support additional attributes. These are Generalizable elements, and by clicking on the Advanced button on the element Properties dialog you can set the following:

- **IsRoot** - the element is a root element and cannot be descended from another
- **IsLeaf** - the element is final and cannot be a parent for other elements
- **IsSpecification** - the element is a specification
- **IsActive** - the element is active; for example, an Active Class
- **Multiplicity** - multiplicity setting for the element.
5.3.6.2 Details

The Details tab of the element Properties dialog is shown below. It enables you to define the structural and processing details for the selected Class element.

Note:
When launched from MDG Integration, the Attributes and Operations buttons are not available.

<table>
<thead>
<tr>
<th>Field/Button</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
<td>Select the cardinality (number of elements in a set) for the Class.</td>
</tr>
<tr>
<td>Visibility</td>
<td>Select the visibility of the Class.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Define attributes for the Class. The Attributes Properties dialog displays.</td>
</tr>
<tr>
<td>Operations</td>
<td>Define operations for the Class. The Operations Properties dialog displays.</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Define how concurrent activities should be processed.</td>
</tr>
<tr>
<td>Collection Classes</td>
<td>Define Collection Classes (for generating code from Association connectors) that apply to this Class. The Collection Classes for Association Roles dialog displays.</td>
</tr>
<tr>
<td>Type</td>
<td>Select the type of Class template parameter to add or list. You can also edit or delete parameters. See the Parameterised Classes topic.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Select a parameter and type any required argument for that parameter.</td>
</tr>
</tbody>
</table>
5.3.6.3 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). Enterprise Architect shows both types, but you can only edit the internal type from this tab.

Internal requirements form the functional requirements of the system to be built. The meaning of the requirement can vary depending on which element is the host; for example, a business process requirement might mean something different to a Use Case requirement, which again might mean something different to a Class requirement. Use these as best suit your model.

Use the Specify Feature Visibility function to show the requirements for an element on the diagram directly (it is also possible to show inherited requirements in this way).

**Note:**

External requirements are those connected to this element using a Realization connector.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>Enter the name and high level detail of the requirement.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type; for example, Functional or Non-functional. Functional requirements are things that the system must do, such as identify franked, unfranked and total credit for a dividend; non-functional requirements are things that the system must be, such as reliable, cost effective.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify the current status of the requirement.</td>
</tr>
</tbody>
</table>
### 5.3.6.3.1 External Requirements

External requirements are those Requirement elements that have been connected to the current element using a **Realization** connector. By creating the connector from the element to the requirement, you create an expectation that the element must implement the requirement 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.
5.3.6.4 Constraints

The Constraints tab of the element Properties dialog is shown below.

Elements can 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 that 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 way).

Add 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. In the **Constraint** field, type the name of the constraint.
4. In the **Type** and **Status** fields, click on the drop-down arrow and select the appropriate constraint type (**Pre-condition**, **Post-condition** or **Invariant**) and status.
5. In the larger text field, type any additional notes required.
6. Click on the **Save** button.

Constraints are used in conjunction with responsibilities to define the conditions and rules under which an element operates and exists.
5.3.6.5 Links

The Links tab of the element Properties dialog displays a list of all relationships active for the current element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Element Stereotype</th>
<th>Type</th>
<th>Connection</th>
<th>Stereotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class1</td>
<td></td>
<td>Class</td>
<td>Generalize</td>
<td></td>
</tr>
</tbody>
</table>

The Relationships panel lists the relationships this element has. The:

- **Element** column identifies the elements this element is related to
- **Element Stereotype** column identifies the stereotype (if any) of the element
- **Type** column identifies the element type of the related element
- **Connection** column identifies the type of relationship
- **Stereotype** column identifies the stereotype (if any) of the relationship.

From the Links tab you can perform operations on a relationship, by right-clicking on the relationship to display the context menu.

To:

- Hide the relationship on the diagram, click on the **Hide Relation** menu option; the option then changes to **Show Relation**, which you select to redisplay the relationship on the diagram
- Display the relationship **Properties** dialog, click on the **Relationship Properties** menu option
What Is UML Modeling? | Work With Elements

- Highlight the related element in the Project Browser, click on the Locate Related Object menu option.
- Delete the relationship from the model and all diagrams, click on the Delete Relationship menu option; the system prompts you to confirm the deletion.

5.3.6.5.1 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 can be applied to any element and can describe functional behavior, business work flows and end-to-end business processes.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario</td>
<td>Type in the name of the scenario.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of scenario; for example, basic path, alternative path.</td>
</tr>
<tr>
<td>Notes</td>
<td>Record a textual description of the scenario, usually depicted in steps of how the user uses the current element. You can format the notes text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Scenarios</td>
<td>Display a list of defined scenarios.</td>
</tr>
</tbody>
</table>

5.3.6.6 Associated Files

The Files tab of the element Properties dialog is shown below. An element can be linked to files held somewhere. Use this tab to set associated files for the current element.

Note:

Linked files are a good way to link elements to additional documentation and/or source code.
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.

### File Path
- **Type**: Local File
- **Last Write**: 
- **Size**: 
- **Notes**: 

### Files
- **Filename**: C:\Temp\UseCaseDoc.tif
- **Type**: Local File

### Options
- **File path**: Type in the name of the file.
- **Type**: Specify the local file or web address.
- **Notes**: Type in free text about the file.
- **Attached files**: Display a list of files.
- **Launch**: Open the selected file. Local files open with their default application and web files open in the default browser.

### 5.3.6.7 Tagged Values
Tagged Values are a convenient way of adding additional information to an element outside that directly supported by UML. UML provides the Tagged Value element for just this purpose. Often Tagged Values 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, follow the steps below:

1. Select the View | **Tagged Values** menu option, or press [Ctrl]+[Shift]+[6]. The Tagged Values window displays.
2. Click on the required element either in a diagram or in the Project Browser. This selects the operation in the Tagged Values window.
3. Click on the **New Tag** button in the Tagged Values toolbar or press [Ctrl]+[N]. The Tagged Value dialog displays.

4. In the **Tag** field, type the tag name (or click on the drop-down arrow and select a custom-defined tag), then in the **Value** field type the appropriate value.

5. Click on the **OK** button to confirm the operation.

**Tip:**
Custom tags can be defined by creating a custom Tagged Value type. For more information see *SDK for Enterprise Architect*.

Tagged Values are the preferred method of extending the code generation capabilities of the CASE tool per element / per language.

### 5.3.6.7.1 Advanced Tag Management

Tagged Values can also be managed within a type hierarchy and with respect to element instances, using the Tagged Values window.

Using the Tagged Values window it is possible to:
- View Tagged Values inherited from parent Classes or realized interfaces or applied stereotypes
- Override Tagged Values derived from parents or applied stereotypes with a unique value for the current element
- Delete Tagged Values from the current element (if a parent version of the Tagged Value exists, it re-appears in the list after the override is deleted).
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 classifier's type hierarchy, additional stereotypes and realized interfaces.

### 5.3.6.7.2 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** menu option. (Alternatively, select one or more elements and press **[Shift]+[Ctrl]+[T]**). The **Tagged Values** dialog displays, which enables you to enter a **Name** and **Value** for the tag.
2. Click on the OK button 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 Properties dialog, go to the Tagged Values window and manually delete the item. There is currently no shortcut to delete tags from multiple elements at one time.

To add notes to the Tagged Value, go to the Tagged Values window, click on the Tagged Value name, and click on the Edit Notes button in the window toolbar. The Notes dialog displays.

Any Notes text you enter also displays in the Info section at the bottom of the Tagged Values window.

### 5.3.6.8 Object Classifiers

Many elements in UML model classifications (such as Classes and Actors), and other elements then model instances of such classifications (such as Objects, Actors again, and Sequence diagram objects). These instance elements represent real things in a run-time scenario; for example, a Person element named Joe Smith. In UML this is written as Joe Smith: Person.

You can define a classifier first, and then instances of that classifier. Alternatively, as a model develops from a rough sketch to a detailed design, many objects become examples of a defined Class, so in the early analysis phase you might model a Joe Smith and a Jane Smith, and later a Person Class from which Joe and Jane are instantiated.

Enterprise Architect enables you to associate an Object with its template element (its classifier), such as a Class. 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 you describe a Person Class with attributes such as Age, Name, Address and Sex, and functions such as GetAge and GetName, then when you associate your 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).
5.3.6.8.1 Using Classifiers

If you right-click on an Object in a diagram, the element context menu displays the Advanced | Instance Classifier menu option. Select this option to choose a single element (generally a Class) as the classifier or template for this Object.

The Set Element Classifier dialog is shown below. Use this to set the instance classifier.

Several Changes Occur if an Object has a Classifier

It is important to remember that an Object is only an instance of a classifier at runtime, so the appropriate attributes and operations are those of the classifier, not the Object. Therefore, in the context menu for the Object, if you select the Attributes or Operations menu options, the Attributes or Operations dialog displays for 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 come from the classifier, not the Object selected. This enables you to associate Sequence diagram objects with Classes and use the defined behavior of the Class to model actual behavior at run time.

You can also select a message for a State Flow connector. The same rules apply as for Sequence diagram objects.

Note that in the Message dialog you can also select to include messages defined in the target classifier’s inheritance hierarchy.
5.3.6.8.2 Set Element Classifier Dialog

This dialog is used primarily to set the base type or classifier for an Object or Lifeline, but is also used in setting classifiers for the return types for operations, the type and return type for operation parameters, Activities for Transitions, Pattern element defaults and Tagged Values. To set a classifier, follow the steps below:

1. The Set Element Classifier dialog displays.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Type</th>
<th>Package</th>
<th>Parent Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>StockItem</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>ShoppingBasket</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>OrderStatus</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>Order</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>LineItem</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>Account</td>
<td>Class</td>
<td>C# Model</td>
<td></td>
</tr>
<tr>
<td>&lt;none&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. If necessary, in the In Namespace field specify a namespace to reduce the list of classifiers. The list reduces to classifiers associated with that namespace.

3. From the list of available classifiers, click on the required type.

4. Click on the OK button.

If the available classifiers do not meet your requirements, you can sometimes create a new element and define the appropriate properties. Click on the Add New Element button. The New Element dialog displays, on which you define the required element.

**Note:**
The Add New Element button is not always available; for example, it is not available when setting classifiers for REFGUID Tagged Values.

5.3.6.9 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).

**Configure Boundary Elements**

Boundary elements can 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 (e.g. Client, Application and Database tiers could be represented in swim lanes).
5.3.7 Compartments

In addition to the attributes and operations compartments shown in a Class element, Enterprise Architect also supports other compartments that can optionally be displayed.

To set the visibility of the various compartments, see Feature Visibility.

Tag Compartment

The Tag compartment lists all Tagged Values for an element as entered in the Tagged Values window.

Or, in the fully qualified expanded format:

Note:
The fully-qualified option operates only on those Tagged Values that were created in Enterprise Architect release 7.1 or later. It does not expand Tagged Values created in earlier releases.

Responsibility Compartment

The responsibility compartment shows a list of responsibilities as entered on the Requirements tab of the element Properties dialog.

Constraint Compartment

The constraint compartment shows a list of element constraints as entered in the Constraint tab of the element Properties dialog.

Testing Compartment

The testing compartment lists all of the tests associated with an element as listed in the Testing window.
(select the View | Testing menu option).

**Maintenance Compartment**

The maintenance compartment lists all of the defects, changes, issues and tasks associated with an element, as listed in the Maintenance window (select the View | Maintenance menu option).
5.3.8 Link Note to Internal Documentation

It is possible to connect a Note element to another element's internal documentation. This enables you to externalize model documentation to the diagram level, and as Enterprise Architect 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 connected into an element’s internal structures. One is connected to an attribute, and displays the attribute name and notes. The other is connected to a constraint, showing the constraint name and documentation.

Procedure

To connect a Note element to a feature of another design element, follow the steps below:

1. Insert the target element into a diagram.

2. Drag the Note icon from the Common page of the Toolbox onto the diagram, next to the target element. The Notes dialog displays. Do not type any text, just click on the OK button.

3. Click on the Note Link icon from the Common page of the Toolbox, click on the Note, and drag across to the target element to create the connector.

4. Right-click on the Note Link to display the context menu.

5. Select the Link this Note to an Element Feature menu option. The Link note to element feature dialog displays.
6. In the **Feature Type** field, click on the drop-down arrow and select the type of feature to link to.
7. In the **Feature** list, click on the specific feature to link to.
8. Click on the **OK** button.

The note now automatically derives its contents from the target element.
5.3.9 Linked Documents

In the Corporate edition of Enterprise Architect, you can link an RTF document to any UML element in the model.

All three editions of Enterprise Architect provide an additional UML Artifact - Document Artifact - that can contain an RTF document internally.

You create linked documents from Linked Document Templates, which you define with the Document Template Editor; see the Create Linked Document Templates and Edit Linked Document Templates topics.

The Document Artifact and the Document Editor are illustrated below:

Documents created via the Document Artifact element are rendered into RTF Documentation by selecting the Model Document checkbox in the RTF Style Template Editor. See the Select Model Components For Documentation topic.
The **Model Document** checkbox is within the **Element** hierarchy, towards the end. Remember that checkboxes can be moved up and down the hierarchy (as has been done above) to position information in the generated document as you require.

The linked document is rendered into the RTF documentation at:

```
model document >
<model document
```

### Connections

<table>
<thead>
<tr>
<th>Connector</th>
<th>Source</th>
<th>Target</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dependency</strong> Link to Node</td>
<td>Public Artifact</td>
<td>Public Model</td>
<td></td>
</tr>
</tbody>
</table>

**Linked Document Editor**

The text of the document displays where ‘Linked Document Editor’ appears, above.

**See Also**

- Create Document Artifact
- Link Document to UML Element
- Edit Linked Documents
- Hyperlink From Linked Document
- Create Element From Document
- Replace or Delete Linked Documents
- RTF Report Dialog Options
5.3.9.1 Create Document Artifact

You create a Document Artifact element in a Component or Deployment diagram. Drag and drop the Document Artifact element from the Enterprise Architect UML Toolbox into your diagram.


In the Copy template field, click on the drop-down arrow and select a previously-created Linked Document Template. Click on the OK button.

For more information on how to create and edit Linked Document Templates, see Create Linked Document Templates and Edit Linked Document Templates.

5.3.9.2 Link Document to UML Element

(This operation is available in the Corporate edition only.) Click on an element in the Project Browser or diagram, and either:

- select the Element | Linked Document menu option or
- right-click and select the Linked Document option from the context menu.

The following dialog displays.

Select the previously-created template from which to create the document. Click on the OK button.

The Linked Document editor displays, in which you enter the text of the document.

For more information on how to create Linked Document Templates, see Create Linked Document Templates.
5.3.9.3 Edit Linked Documents

Enterprise Architect provides a Windows-like word processor to help you edit Linked Documents. This is a simplified version of the RTF Style Template Editor, and it provides the same convenient features.

The main difference between the two editors is that you access the RTF Style Template Editor features through a menu bar at the top of the screen, whilst you access the Linked Document Editor features through a context menu. To access the context menu, just right-click anywhere on the document.

This is the text of a linked document:

The following topics provide assistance on using the Document Editor.

- Scroll Through Text
- File and Print Options
- Cut and Paste Options
- Image and Object Imports
- Character Formatting
- Paragraph Formatting
- Tab Support
- Page Breaks and Repagination
- Insert Headers and Footers
- Insert Bookmarks
- Table Commands
- Sections and Columns
- Stylesheets and Table of Contents
- Text/Picture Frame and Drawing Objects
- View Options
- Search/Replace Commands
- Hyperlink From Linked Document
- Create Elements From Linked Documents
5.3.9.4 Hyperlink From Linked Document

Within a linked document, you can add hyperlinks to other objects (elements, packages, diagrams, attributes and operations) in the Enterprise Architect Project Browser.

To do this, click on the object in the Project Browser and drag it to the point at which to create the hyperlink. The linked document editor automatically creates the hyperlink, using the object name as the hyperlink text. You can edit this text if required.

When you next open the document, you can double-click on the hyperlink to locate and highlight the object in the Project Browser. You can then perform all normal operations on the object, including opening any linked document on the highlighted element.

You can also create a hyperlink to an external document or web page. See the Headers, Footers, Hyperlinks and Bookmarks topic.

5.3.9.5 Create Element From Document

Using the Linked Document Editor, you can create document-specific elements and diagrams in the Project Browser, with hyperlinks from the document to the created item. When you click on the hyperlink, the element or diagram is highlighted in the Project Browser. The element or diagram is created in the same package as the element for which the linked document was created.

You can create and link to any type of element or diagram, but the facility has specific options for the following element types:
- **Class**
- **Requirement**
- **Issue**

You can create the same arrangement with existing elements, diagrams and packages by dragging them from the Project Browser into the text of the document, creating a hyperlink with the item name as the text.

Create Item

To create an element or diagram in the Project Browser, whilst in a linked document, follow the steps below:

1. Open the linked document, either from a Document Artifact element or through the context menu (Corporate edition only).
2. Enter some text, including appropriate text to act as the link (such as the element or diagram name).
3. Highlight the appropriate text and right-click on it. The editor context menu displays.
4. Select the UML menu option, and the required submenu option.
5. If you select the:
   - **Add new Class**, **Add new Requirement** or **Add new Issue** option, the corresponding element is immediately created in the Project Browser.
   - **Add Element** option, the New Element dialog displays; specify the element type and - if appropriate - stereotype, and click on the OK button.
   - **Add Diagram** option, the New Diagram dialog displays; specify the diagram type and click on the OK button.
6. The highlighted text is now a hyperlink. Click on the link to highlight the new element or diagram in the Project Browser.

You can now edit or expand the element or diagram as required.

5.3.9.6 Replace or Delete Documents

If a linked document is out of date, you can either edit the text or replace the entire contents from another file. To replace the contents:

1. Click in the body of the document and press [Ctrl]+[A] to select all the document text.
2. Press [Delete].
3. Right-click and select the File | Import menu option. The Windows Open dialog displays, in which you can browse for the file to import into the document.
4. Click on the Save icon in the Linked Document screen toolbar.

Alternatively, you can delete the linked document. To do this:
1. Click on an element in the Project Browser or diagram, and either:
   - select the Element | Delete Linked Document menu option or
   - right-click and select the Delete Linked Document option from the context menu.
2. Enterprise Architect prompts you to confirm the deletion; click on the Yes button.
   If required, you can now create another linked document for the element.

5.3.9.7 Create Linked Document Templates

Linked Document templates can be created via the Resources window.

Under the Templates folder, right-click on the Linked Document Templates icon and click on the Create Template menu option. The following dialog displays.

Enter a name for your template, or select a previously-created template. Click on the OK button.

You can group your templates into folders. Right-click on your newly created template and select the Assign Template to Group menu option. Enter a category name and click on the OK button.

You can also modify and delete the templates using the context menu options.

Note:
You can transport these linked document templates between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
5.3.9.8 **Edit Linked Document Templates**

Double-click on a previously created template in the **Resource View** to invoke the **Linked Document Template Editor**.

The **Document Template Editor** is built into **Enterprise Architect**.

The following topics provide assistance on using the **Document Editor**.

- **Scroll Through Text**
- **File and Print Options**
- Cut and Paste Options
- Image and Object Imports
- Character Formatting
- Paragraph Formatting
- Tab Support
- Page Breaks and Repagination
- Headers, Footers, Hyperlinks and Bookmarks
- Table Commands
- Sections and Columns
- Stylesheets and Table of Contents
- Text/Picture Frame and Drawing Objects
- View Options
- Search/Replace Commands
5.4 Work With 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
- Connector Tasks
- Connector Properties

Off-Page Connector

UML, and therefore Enterprise Architect, does not have a connector that continues activity flow between two diagrams. If the need arises, in creating a model diagram, to continue flow to another diagram, you should consider revising and simplifying the structure of the process so that groups of Actions are captured in composite Activity elements, and each group of Actions is modeled within the child diagram of an Activity.

BPMN, however, does enable you to create off-page connectors. You can also use the Suppress Line Segments menu option to indicate continuation of flow in a large diagram that, when printed, occupies several pages. Be aware that these options are purely diagrammatic and do not indicate any diagram relationships in any of the relationship tools.
5.4.1 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 up to seven distinct sections:

- **Add-Ins** - displays in the first section only if you have Add-Ins installed and registered, such as Eclipse
- **Properties**
- **Type Specific**
- **Style**
- **Appearance**
- **UML Help** - Displays the Enterprise Architect Help topic for this connector type
- **Delete** - delete the connector with this option.

**Note:** Not all menu options are present on all connector context menus. Context menus vary slightly between connector types. The type specific menu options are not always included, for example.

**Example Context Menu for a Generalization:**

```
Add-Ins
Generalization Properties...
Advanced
Attach Note or Constraint...
Line Style
Pin End(s)
Appearance...
Visibility
Tidy Line Angles
UML Help
Delete Connector
```

**Example Context Menu for an Association:**

```
Add-Ins
Association Properties...
Advanced
Attach Note or Constraint...
Line Style
Pin End(s)
Bond Line at Cursor (Ctrl+Q)
Appearance...
Visibility
Tidy Line Angles
UML Help
Delete Connector
```

**Connector Role Context Menu**

For connectors with *Roles*, right-clicking a connector within up to 60 pixels of an end point raises a role-specific context menu.

The **Role** context menu has two additional menu options:

- **A Source/Target Role...** menu option that opens the connector specification dialog with the respective role page activated.
- **A Multiplicity** submenu that enables the multiplicity for the role to be set.

### 5.4.1.1 Properties Menu Section

The **Properties** section of the connector context menu contains the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Connector type&gt; Properties</code></td>
<td>Open the <strong>Properties</strong> window for the selected connector.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Display the <strong>Advanced</strong> menu.</td>
</tr>
<tr>
<td>Attach Note or Constraint</td>
<td>Attach a note or constraint to the connector.</td>
</tr>
</tbody>
</table>
Note:
Not all menu options are present on all connector context menus. Context menus vary slightly between
connector types. The type specific menu options are not always included, for example.

### 5.4.1.2 Type-Specific Menu Section

The *Type-Specific* section of the connector context menu is specific to the object, and only appears for a few
different connectors. Some examples are shown below:

<table>
<thead>
<tr>
<th>Connector</th>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>Message</td>
<td>Set the value of the Message.</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Set Aggregation to Composite</td>
<td>Change the Aggregation to composite.</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Set Aggregation to Shared</td>
<td>Set the Aggregation to shared. Appears after <em>Set Aggregation to Composite</em> has been selected.</td>
</tr>
<tr>
<td>Association</td>
<td>Specialize Associations</td>
<td>Specify how the properties of this Association specialize the properties of other Associations.</td>
</tr>
<tr>
<td>Dependency</td>
<td>Dependency Properties</td>
<td>Select a stereotype for the dependency.</td>
</tr>
<tr>
<td>Trace</td>
<td>Dependency Properties</td>
<td>Select a stereotype for the dependency.</td>
</tr>
<tr>
<td>Role Binding</td>
<td>Dependency Properties</td>
<td>Select a stereotype for the dependency.</td>
</tr>
<tr>
<td>Represents</td>
<td>Dependency Properties</td>
<td>Select a stereotype for the dependency.</td>
</tr>
<tr>
<td>Occurrence</td>
<td>Dependency Properties</td>
<td>Select a stereotype for the dependency.</td>
</tr>
</tbody>
</table>

Note:
Not all menu options are present on all connector context menus. Context menus vary slightly between connector types. The type-specific menu options are not always included, for example.

### 5.4.1.3 Advanced Menu Section

The *Advanced* section of the connector context menu contains the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Source and Target</td>
<td>Change the source and/or target of the connector.</td>
</tr>
<tr>
<td>Change Type</td>
<td>Change the connector type</td>
</tr>
<tr>
<td>Reverse Direction</td>
<td>Reverse the direction of the connector. For example, if the connector is an arrow, the arrowhead swaps to the other end.</td>
</tr>
<tr>
<td>Information Flows Realized</td>
<td>Realize any information items conveyed on an Information Flow connector between these same two elements.</td>
</tr>
</tbody>
</table>

Note:
Not all menu options are present on all connector context menus. Context menus vary slightly between connector types. The type specific menu options are not always included, for example. Certain connectors have Custom properties, which are predefined. However, you can change the values of these properties. See the *Custom Properties Dialog* topic.
### 5.4.1.4 Style Menu Section

The **Style** section of the connector context menu provides the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Line Style</strong></td>
<td>Set the connector line style - options are Direct, Auto Routing, Custom, Bezier, Tree (Horizontal) or Tree (Vertical).</td>
</tr>
<tr>
<td><strong>Pin End(s)</strong></td>
<td>Pin the start and/or connector ends to a position on the target element.</td>
</tr>
<tr>
<td><strong>Bend Line at Cursor [Ctrl]+[Q]</strong></td>
<td>Insert an anchor point on the line at the point of the cursor so you can change the shape of the line.</td>
</tr>
<tr>
<td><strong>Suppress Line Segment</strong></td>
<td>Hide a segment of a connector so that you can view a part of the diagram that it crosses. To reverse the change, right-click on the connector and select the <strong>Show All Line Segments</strong> menu option.</td>
</tr>
<tr>
<td><strong>Straighten Line at Cursor [Ctrl]+[Q]</strong></td>
<td>Remove an anchor point on the line at the point of the cursor. (This is the exact opposite of <strong>Bend Line at Cursor</strong>, and [Ctrl]+[Q] toggles the connector point between the options.)</td>
</tr>
</tbody>
</table>

**Note:**

Not all menu options are present on all connector context menus. Context menus vary slightly between connector types. The type specific menu options are not always included, for example.

### 5.4.1.5 Appearance Menu Section

The **Appearance** section of the connector context menu provides the following options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appearance</td>
<td>Set the line color and line thickness of the connector.</td>
</tr>
<tr>
<td>Visibility</td>
<td>Set connector visibility; see table below for sub-menu options.</td>
</tr>
<tr>
<td>Tidy Line Angles</td>
<td>Tidy the line angles of a custom connector.</td>
</tr>
</tbody>
</table>

**Visibility Sub-Menu**

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide Connector</td>
<td>Hide the connector. To show the connector again, follow the steps in the <strong>Hide/Show Connector</strong> topic.</td>
</tr>
<tr>
<td>Hide Connector in Other Diagrams</td>
<td>Hide or show the connector in other diagrams.</td>
</tr>
<tr>
<td>Hide All Labels</td>
<td>Hide or show all labels attached to the connector.</td>
</tr>
<tr>
<td>Set Label Visibility</td>
<td>Hide or show labels attached to the connector, individually.</td>
</tr>
</tbody>
</table>

**Note:**

Not all menu options are present on all connector context menus. Context menus vary slightly between connector types. The type specific menu options are not always included, for example.
5.4.2 Connector Tasks

This topic details some of the tasks associated with managing model connectors, such as:

- 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 Connector
- Use Tree Style Hierarchy
- Create a Connector
- Show Uses Arrow Head
- Set Association Specializations

Note:

In the Corporate edition, if security is enabled, you must have Update Element permission to update or delete a connector.

5.4.2.1 Connect Elements

Connect Elements on a Diagram

The fastest and simplest ways to create connectors are using the Quick Linker and using the Enterprise Architect UML Toolbox. The following topics describe these and other approaches for creating connectors on a diagram:

- Create Connectors In Place Using the Quick Linker
- Create Connectors Using the Enterprise Architect UML Toolbox
- Create a Group of Elements Using UML Patterns
- Create Domain Specific Connectors From UML Profiles

Tip:

To repeat the last connector you used, click on the appropriate source element and press [F3].

Select Connectors

To select a connector, simply click on it. Drag handles display, indicating that the connector is selected. This gives the connector focus for keyboard commands such as [Delete], and displays connector properties in docked windows such as the Tagged Values window. If there is more than one connector on a diagram, you can cycle through them using the arrow keys.

Drag Connectors

You can drag a connector to position it. Click on the connector and drag the connector to where it is to appear. Note that there are some limitations on how far or to where you can drag a connector.

Note:

You can reposition a connector by selecting and dragging the connectors as required.
Tip:
To reattach the end of a connector to a different source or target element, see Change the Source or Target Element.

Connector Properties and Commands
You can double-click on a connector to change properties, or right-click to display the context menu containing commands to change connector type and direction.

You can also highlight the connectors on a specific element. Select the element and press [L]. All the connectors issuing from or terminating at that element are highlighted.

Create Connectors Without a Diagram
Sometimes it is useful to create relationships between elements without a diagrammatic representation. You can do this using the Project Browser and the Relationship Matrix, as explained in the following topics:
- Add Connectors With the Project Browser
- Add Connectors With the Relationship Matrix

5.4.2.2 Add a Note to a Connector
You can connect notes and constraints to graphical relationships. Notes enable you to provide explanations 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 connectors; for example the (XOR) constraint in the image above indicates that only one of the connectors in the specified set can be true at any one time (exclusivity).

Attach a Note or Constraint to a Connector
To attach a note or constraint to one or more connectors, follow the steps below:
1. Right-click on one of the connectors to attach a note to. The context menu displays.
2. Select the Attach Note or Constraint menu option. The Link Relations dialog displays.
3. Check all the connectors that participate in the set. In the example below, two connectors have been checked to participate in a logical constraint.
4. Click on the OK button to complete the note or constraint creation.
5. You can 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 { and } automatically added to visually indicate the constraint form.

5.4.2.3 Arrange Connectors

Connectors between two elements can be moved around the element borders to create 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 line style; this enables you to add as many line points and bends as you require to create a clean and readable diagram.

Move a Connector

To move a connector, follow the steps below:

1. Click once on the connector to select it.
2. Holding the mouse button down, move the connector in the required direction.
3. To refine the movement, click and hold very near to one end of the connector; this enables a slightly different movement range.
4. To further refine the movement and range, select either a routed, direct or custom line style. Each behaves slightly differently (see Connector Styles).
5.4.2.4 **Change Connector Type**

To change a connector type, follow the steps below:

1. In the **Diagram view**, right-click on the connector to change. The context menu displays.
2. Select the **Connection Detail | Change Type** menu option.
3. In the **Connector Type** field, click on the drop-down arrow and select the required connector type.
4. Click on the **OK** button to apply changes.

5.4.2.5 **Change the Source or Target Element**

After you have created a connector between two elements, you might later want to change either the source or target. Instead of deleting and re-creating the connector, Enterprise Architect enables you to change the source or target. There are two ways of doing this: using the **Set Source and Target** dialog or using the mouse.

**Using the Set Source and Target dialog**

To change the source or target element of a connector using the **Set Source and Target** dialog, follow the steps below:

1. Right-click on the connector to open the context menu.
2. Select the **Advanced | Set Source and Target** menu option. The **Set Source and Target** dialog displays.
3. Click on the drop-down arrows on the **From Element** and **To Element** fields, and select the source and target elements.

4. Click on the **OK** button to apply changes.

## Using the Mouse

To change the source or target element of a connector using the mouse, follow the steps below:

1. Click on the connector and position the cursor over the ‘handle’ at one end.
2. When the cursor changes, click the mouse and drag the handle to the new element.

**Note:**
The connector does not actually move until you release the mouse button over the new source or target element. However:

- A dotted line shows where the connector would be during the move, and
- The solid outline of the nearest element or extension changes to a hatched outline as you move the cursor onto it; this helps you identify where the connector will connect to, if there are many closely-arranged elements, Parts, Ports and other extensions.

## 5.4.2.6 Connector Styles

Connectors come in five different routing styles:

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct</strong></td>
<td>A straight line from element A to element B. You can move the line (back and forward, up and down) to a limited degree.</td>
</tr>
<tr>
<td><strong>Auto Routing</strong></td>
<td>A vertical and horizontal route from A to B with 90-degree bends. You can move the line to improve the route, but the location and number of bends are not configurable.</td>
</tr>
<tr>
<td><strong>Bezier</strong></td>
<td>A smooth curved line from A to B. Bezier style is directly available for Data Flow diagram connectors, Mind Mapping connectors, State Flows, State Transitions, Object Flows, and Control Flows.</td>
</tr>
<tr>
<td><strong>Custom</strong></td>
<td>The most flexible option. You can add one or more line points and bend and push the line into virtually any shape, using the Toggle Line Point at Cursor option.</td>
</tr>
<tr>
<td><strong>Tree (Horizontal, Vertical)</strong></td>
<td>A line from element A to B with two bends, and the end points fixed to selected locations on the elements (Horizontal or Vertical).</td>
</tr>
</tbody>
</table>

### Set the Connector Style

To set the connector style, follow the steps below:

1. Right-click on the connector to change; the context menu displays.
2. Select the **Line Style** option.
3. From the submenu, select the required style - Direct, Auto Routing, Custom or Tree (and Bezier, where appropriate).
Alternatively:

1. Select the connector to change.
2. Press the following keys to change the style:
   - \[\text{Ctrl}]+[\text{Shift}]+[D]\ for Direct
   - \[\text{Ctrl}]+[\text{Shift}]+[A]\ for Auto Routing, or
   - \[\text{Ctrl}]+[\text{Shift}]+[C]\ for Custom
   - \[\text{Ctrl}]+[\text{Shift}]+[Z]\ for Bezier, where appropriate).

**Bend Connectors**

To bend a connector to quickly and easily route connectors in the required layout, follow the steps below:

1. Right-click on the connector; the context menu displays.
2. Set the line style to Custom Line \([\text{Ctrl}]+[\text{Shift}]+[C]\); this enables the **Bend Line at Cursor** option in the context menu.
3. Click on the **Bend Line at Cursor** option to add a line point.

   **Note:**
   Right-clicking a line point displays the **Straighten Line at Cursor** option, which you can use to remove the line point.

4. Using the mouse, drag the line point to the required position.

Alternatively:

1. Hold down \[\text{Ctrl}\] or \[\text{Shift}\] and click on a point on the connector to create a line point.

   **Note:**
   \[\text{Ctrl}\]+click also removes a line point.

2. Using the mouse, drag the line point to the required position.

**Tidy Line Angles**

To tidy line angles (custom connector), follow the steps below:

1. Right-click on the connector; the context menu displays.
2. Click on the **Tidy Line Angles** menu option; this nudges the custom line in horizontal and vertical increments, saving you the time of trying to get a good layout manually.

**Note:**

You can set the **Tidy Line Angles** option to operate by default; click on the **Tools | Options** menu option to display the **Options** dialog, and select the **Diagram Behavior** page.

### Suppress Line Segments

To suppress individual line segments, follow the steps below:

1. Right-click on the connector; the context menu displays.
2. Set the line style to Custom Line ([Ctrl]+[Shift]+[C]), this enables the **Suppress Line Segment** option in the context menu.
3. Click on the **Suppress Line Segment** option to suppress a line between two bend points.

**Note:**

The segment you right-clicked is suppressed.

4. To show the segment again, right-click on the line and click on the **Show All Line Segments** option. One application for this is to represent the continuation of flow when your diagram crosses the page boundary marker in the **Diagram View**. When you suppress the line segment that crosses the boundary, the link name (connector properties) displays at both ends of the hidden segment. When you print the diagram on multiple pages, the link name identifies the connection apparently broken by the page boundary.

### 5.4.2.7 Create Connector in Project Browser

You can create a connector from one element to another directly in the **Project Browser**.

### Connect Elements from the Project Browser

To connect elements from the **Project Browser**, follow the steps below:

1. In the **Project Browser**, either:
   - Right-click on the element to create a connector for, and select the **Add | Create Link** context menu option, or
   - Select the element, press **[Insert]** and select the **Create Link** context menu option. The **Create Link** dialog displays.
2. In the **Direction** field, click on the drop-down arrow and select the direction of the new connector (**Outgoing** means this element is the source).
3. In the **Link Type** field, click on the drop-down arrow and select the type of connector.

4. In the **Choose target(s)** list, click on the name of the target. (If necessary, in the **Select Target Type** field click on the drop-down arrow and select a feature to list only elements having that feature.)

5. Click on the **OK** button to create the connector.

### 5.4.2.8 Delete Connectors

To delete a connector, follow the steps below:

1. Right-click on the connector. The context menu displays.

2. Select the **Delete Connector** option. The **Remove Connector** dialog displays.
3. This dialog provides the options to hide the connector so that it remains functional, or remove the connector completely. Click on the appropriate radio button and click on the OK button. If you select the Hide option, it has the same effect as hiding the connector on the Links tab of the source element Properties dialog, or using the Visibility | Hide Connector context menu option. It also hides the relationship on the Relationships window.

**Note:**
The dialog does not display if:

- You have previously selected the Don't ask again checkbox or
- On the Links page of the Options dialog (Tools | Options | Links) the Prompt on connector deletes checkbox is not selected.

Selecting the Don't ask again checkbox also deselects the Prompt on connector deletes checkbox. Selecting the Prompt on connector deletes checkbox restores the dialog if you have used the Don't ask again checkbox.

If you hide the dialog, the Delete Connector context menu option defaults to the setting you last used on the dialog. Make sure that you have selected the right option to default to.

### 5.4.2.9 Generalization Sets

A generalization set enables you to specify the relationship of a group of subtypes.

To create a generalization set, follow the steps below:

1. Right-click on the connector. The context menu displays.
2. Select the Advanced | Generalization Set | New menu option. The following dialog displays.
3. In the **Name** field, type the name of the Generalization set. e.g. **Gender**.
4. In the **Power Type** field, either type a new power type or click on the drop-down arrow and select an existing one.
5. Check the **IsMember** column for the child subtypes that are part of this Generalization set.

The **OMG UML specification (UML Superstructure Specification, v2.1.1, section 7.3.21, p. 77)** 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.*

### 5.4.2.10 Hide/Show Connectors

Connectors/relations that appear in multiple diagrams can be selectively shown or hidden. This makes it easier to read diagrams where elements might 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. Double-click on the required diagram element in the **Diagram** view. The element **Properties** dialog displays.
2. Select the **Links** tab.
3. Right-click on the connector to hide or show. The context menu displays.
4. Select the **Show Relation** menu option to show the hidden connector on the diagram, or the **Hide Relation** menu option to hide the visible connector.
Tip:
Alternatively, hide a connector by right-clicking on it on the diagram and selecting the Visibility | Hide Connector menu option from the context menu. However, you must use the Links tab of the element Properties dialog to show the relationship again.

Note:
Certain elements, such as Requirements, do not have a Links tab in the Properties dialog. In these cases, open the Relationships window (View | More Windows | Relationships) for the element and right-click on the relationship in the list to display the context menu. This enables you to hide or show that relationship in the diagram. Be aware that, in the Corporate edition with security on, locks on the diagram and elements can make the required option unavailable.

Hide or Show a Connector in Other Diagrams
To hide or show a connector in other diagrams, follow the steps below:

1. Right-click on the connector in the diagram. The context menu displays.
2. Select the Visibility | Hide Connector in Other Diagrams menu option. The Set Connector Visibility dialog displays.

Clear check box to hide connector in diagram:

- [ ] Inventory
- [x] Orders

[ Suppress All ]  [ OK ]  [ Cancel ]

3. If the two connected elements have been included in other diagrams, these diagrams are listed here. In the list, all diagrams for which the checkbox is selected show the connector. Deselect the checkbox for any diagrams in which to hide the connector.

Tip:
To hide the connector in all of the diagrams listed, click on the Suppress All button.

4. Click on the OK button to save the changes.

5.4.2.11 Hide/Show Labels
You can hide or display one or more labels on a connector. To do this, follow the steps below:

1. Right-click on the connector. The context menu displays.
2. Select the Visibility | Set Label Visibility menu option. The Label Visibility dialog displays.
3. Select the checkbox against each label to display, and clear the checkbox against each label to hide.
4. Click on the OK button.

5.4.2.12 Connector In-place Editing Options

You can edit many of the Enterprise Architect connector labels directly on the diagram. Each label can be bound to a single connector field.

Procedure
To put a label into Edit mode, either:
- Select the Edit Label option from the context menu, or
- Select a label and press [F2].

To save the current text to the field, either press [Return] or deactivate the Edit window.
To cancel edit mode without saving any changes, press [Esc].

5.4.2.13 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.

Procedure
To reverse a connector, follow the steps below:
1. Right-click on the incorrect connector to open the context menu.
2. Select the Connection Detail | Reverse Direction menu option.

5.4.2.14 Set Association Specializations

UML enables specialization of properties defined by Associations. Enterprise Architect enables this through the Specialize Associations option in the advanced section of the context menu for an Association.

The following dialog displays, showing all Associations between the two Classes connected by the current Association and their parents.
5.4.2.15 **Show Uses Arrow Head**

By default the *Use* connector in Use Cases has no arrow head. To generate arrow heads on the connectors, follow the steps below.

1. Select the **Tools** | **Options Links** menu option. The *Links* page of the *Options* dialog displays.

2. In the **General** panel, select the **Show Uses arrowheads** checkbox.
3. Click on the **Close** button.

When you save the Use Case diagram, the Use connectors change to display arrowheads.

### 5.4.2.15.1 Relationship Visibility

You can change the visibility of individual connectors or relationships, diagram by diagram.

**Set Relationship Visibility**

To set relationship visibility, follow the steps below:

1. Open the diagram to change.
2. Select the **Diagram** | **Visible Relations** menu option. Alternatively, press **[Ctrl][Shift][I]**.
3. Select the checkbox against each list item to show, and clear the checkbox against each item to hide.
4. Click on the **OK** button to apply the changes.
5.4.2.16 **Tree Style Hierarchy**

In Enterprise Architect you can create a tree style inheritance diagram using a special form of the Generalization connector, as shown below.

![Diagram showing a tree style hierarchy](image)

**Note:**
The Son ->Parent connector has not yet been put in Tree Style - Vertical style.

This style of diagram provides a clearer layout for inheritance hierarchies and is easy to work with.

**Create a Tree Style Connector**

To create a tree style connector, follow the steps below:

1. Create a normal Generalization between two elements.
2. Right-click on the connector to open the context menu.
3. Select the **Line Style | Tree Style - Vertical** or the **Line Style | Tree Style - Horizontal** menu option.
4. Enterprise Architect automatically makes the Generalization layout conform to a specific shape. By adding more Generalization connectors, and checking their **Tree Style** options, you can achieve the appearance of the diagram above. You can slide the root and child Classes left and right to achieve the required result; Enterprise Architect maintains the conformity of the branch connectors.
Set the Default Connector Style

To set this style of connector as default, follow the steps below:

1. Select the Tools | Options | Links menu option. The Links page of the Options dialog displays.

2. Select the Generalization link style Default = Tree checkbox to make this branching style the default style for inheritance connectors.
5.4.3 Connector Properties

To access the connector Properties dialog, double-click on a connector in a diagram. You can change several characteristics of connectors from this dialog.

Many of these characteristics generate text labels on or around the connector. You can change these labels using the Label context menu.

The connector Properties dialog has the following tabs:

- General (see below)
- Constraints
- Source Role
- Target Role

The General tab enables you to configure the name of the connector (Link Name), the direction, the line style, the stereotype (optional) and a comment.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Type in the name of the source element for the connector.</td>
</tr>
<tr>
<td>Target</td>
<td>Type in the name of the target element for the connector.</td>
</tr>
<tr>
<td>Name</td>
<td>(Optional) Type a name for the connector. If entered, the name displays on the diagram.</td>
</tr>
<tr>
<td>Alias</td>
<td>(Optional) Type an alternative name or alias for the connector.</td>
</tr>
<tr>
<td>Direction</td>
<td>Select the appropriate direction details: from source to destination, destination to source, or bi-directional. Some connectors have arrow heads that depend on this setting. Some connectors are logically dependent on this (e.g. inheritance).</td>
</tr>
</tbody>
</table>
### What Is UML Modeling? | Work With Connectors

#### 5.4.3.1 Connector Constraints

A UML connector can also have associated constraints placed on it. Constraints tell us something about the rules and conditions under which a relation operates. For example, it might be a pre-condition that a customer is of a certain type before an Association connector to an Account is allowed.

**Tip:**

Constraints about an Association (connector) can be added to further refine the model. Constraints detail the business and operational rules for the model.

**Set Constraints on a Connector**

To set constraints on a connector, follow the steps below:

1. Double-click on a connector to open the Connector Properties dialog.
2. Select the Constraints tab.
3. Fill in details of the constraint(s) that apply and click on the Save button.

---

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Style</strong></td>
<td>Select the appropriate connection style; choose from: <strong>Direct</strong>, <strong>Auto-Routing</strong>, <strong>Bezier</strong>, <strong>Custom</strong>, <strong>Tree (Vertical)</strong> or <strong>Tree (Horizontal)</strong>.</td>
</tr>
<tr>
<td><strong>Stereotype</strong></td>
<td>(Optional) Type the name of a stereotype for the connector, or click on the drop-down arrow and select one. Alternatively, click on the [ ... ] button and select the stereotype from the Stereotype Selector dialog. If entered, the stereotype is displayed on the diagram and over-rides the connector type in the RTF documentation.</td>
</tr>
<tr>
<td><strong>Virtual Inheritance</strong></td>
<td>Indicate that inheritance is virtual. Available only for Generalization connectors.</td>
</tr>
<tr>
<td><strong>Scope</strong></td>
<td>Select the appropriate value for the scope (used for inheritance). Available only for Generalization connectors where the child Class is C++.</td>
</tr>
<tr>
<td><strong>Notes</strong></td>
<td>(Optional) Type any notes on the connector. The notes are displayed in documentation, if required. You can format the text, using the Rich Notes Text toolbar at the top of the field.</td>
</tr>
</tbody>
</table>

**See Also**

- Role Tagged Values
- Message Scope

---

© 1998-2008 Sparx Systems Pty Ltd
What Is UML Modeling? | Work With Connectors

5.4.3.2 Source Role

This description refers to the role of the Source element in a relationship, but applies equally to the role of the Target element.

A connector can have certain properties assigned to one end, and be associated with the particular role that element plays in the relationship. You can enter details about this role to further develop your model.

Set Source Role Details

To set the source role details, follow the steps below:

2. Select the Source Role tab.
3. Enter the required details and click on the OK button.
### What Is UML Modeling? | Work With Connectors

#### Option Use to

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Type&gt;</code> Role</td>
<td>Type in the name of the role to be played.</td>
</tr>
<tr>
<td>Alias</td>
<td>Type an alias for the role, if required.</td>
</tr>
<tr>
<td>Role Notes</td>
<td>Type in any required notes about the role.</td>
</tr>
<tr>
<td>Derived</td>
<td>Indicate that the role value or values can be computed from other information.</td>
</tr>
<tr>
<td>Owned</td>
<td>Indicate that the role is owned by the opposite Class as opposed to the Association.</td>
</tr>
<tr>
<td>Derived Union</td>
<td>Indicate that the role is derived from the properties that subset it.</td>
</tr>
</tbody>
</table>
| Multiplicity         | Specify the role multiplicity. (You can define the values of this field on the Cardinality tab of the UML Types dialog.) This is the range of instances of the role that can be active in the relationship; for example, one employee can be assigned to tasks; for the target role you define the range of instances (e.g. tasks) the employee could be assigned to. The values have the following formats: * `*`, or `0..*` - zero, one or many instances  
  * `0..n` - zero or up to n instances, but no more than n  
  * `n` - exactly n instances  
  * `n..*` - n, or more than n instances.  
  Note that you can also define source and target element multiplicity in the element Attribute properties. |
| Ordered              | Indicate that the role is a list and the list is ordered. |
| Allow                | Indicate that the role can contain duplicate elements (relevant only if multiplicity is > 1). |

© 1998-2008 Sparx Systems Pty Ltd
### What Is UML Modeling? | Work With Connectors

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicates</td>
<td>Maps to the UML property <em>isUnique</em> (selecting the checkbox maps to the <em>isUnique</em> value of <em>FALSE</em>).</td>
</tr>
<tr>
<td>Containment</td>
<td>Indicate the nature of the containment at the Destination (reference, value...).</td>
</tr>
<tr>
<td>Access</td>
<td>Select the access level for the role.</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Select the type of aggregation that this role uses.</td>
</tr>
<tr>
<td>Target Scope</td>
<td>Select the level at which this role applies (instance or classifier).</td>
</tr>
<tr>
<td>Navigability</td>
<td>Select whether or not this role is navigable (non-navigable ends are shown depending on diagram properties).</td>
</tr>
<tr>
<td>Changeable</td>
<td>Select whether this role is subject to change.</td>
</tr>
<tr>
<td>Constraint(s)</td>
<td>Type in any constraint on the role.</td>
</tr>
<tr>
<td>Qualifier(s)</td>
<td>Type any qualifiers or restrictions on the role. Separate multiple qualifiers with a semi-colon.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>(Optional) Type or select the name of a stereotype that applies to this end of the Association.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Type a role type that can be used when generating collection Classes for multiplicity &gt; 1.</td>
</tr>
</tbody>
</table>

**Note:**

Source role details are displayed at the start end of a connector. If you have drawn the connector the wrong way, you can always use the *Reverse Direction* menu option from the connector context menu.

#### 5.4.3.3 Target Role

A connector can have certain properties assigned to one end, and be associated with the particular role that element can play in the relationship.

You can 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 Connector Properties dialog.
2. Select the Target Role tab.
3. The details and appearance of this tab are identical to the Source Role tab. See [Source Role](#).

**Note:**

Destination role details are displayed at the terminating end of a connector on the diagram.
5.4.4 Role Tagged Values

For Association and Aggregation connector types you can set additional Tagged Values on the source and/or target role.

Set Tagged Values

To set Tagged Values for the connector, follow the steps below:

1. Press [Ctrl]+[Shift]+[6] or select the View | Tagged Values menu option. The Tagged Values window displays.
2. Click on the connector in the diagram. The Tagged Values information for the connector displays in the Tagged Values window.
3. Select Connector Source or Connector Target as required.
4. Either click on the New Tags button or press [Ctrl]+[N]. The Tagged Value dialog displays.
5. In the Tag field type the tag name and value, or click on the drop-down arrow and select a predefined Tagged Value type.
6. Click on the OK button to save the changes.
5.4.5 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 might have to change either the start or end point of a message as the responsibilities of elements change during design. For this reason, Enterprise Architect enables you to change the message scope by setting a new start or end element.

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.
4. In the pop up dialog, in the From Element and To Element fields, click on the drop-down arrows and select the required elements.
5. Click on the OK button to save changes.

The message is re-routed to meet your changed requirements.
5.5 Requirements Management

Introduction
Gathering requirements is typically the first step in developing a solution, be it for developing a software application or for detailing a business process. Requirements are essentially 'what the system must do'. The requirements management built into Enterprise Architect can be used to define Requirement elements, connect Requirements to model elements that implement that requirement, connect Requirements into a hierarchy, report on Requirements and move internal requirements (responsibilities) into external model element Requirements.

Typical Tasks
Typical tasks you could perform when managing requirements with Enterprise Architect include:

- Create Requirements
- External Requirements
- Color Code External Requirements
- Internal Requirements
- Make Internal Requirement External
- Requirement Properties
- Composition
- Implementation
- Requirements Hierarchy
- Dependency Report
5.5.1 Create Requirements

Create Requirements at Diagram Level
To create requirements at the diagram level, follow the steps below:

1. Open the Custom pages on the Enterprise Architect UML Toolbox.
2. Drag the Requirement element onto the current diagram.
3. Enter the Name and other details for the requirement.

Enterprise Architect creates a requirement in the current diagram and in the current package.

Note:
External requirements can be created with or without an identifying E in the top right corner of the element. To toggle display of this letter, select or deselect the Show stereotype icon for requirements checkbox on the Options dialog, Objects page (Tools | Options | Objects).

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 the Insert | New Element menu option. Alternatively, press [Ctrl]+[M].
3. In the New Element dialog select the Requirement type.
4. Enter the Name (or select Auto) and click on the OK button.

Enterprise Architect creates a requirement in the current package.

Tip:
You can also move internal responsibilities of an element to external requirements - see the Make Internal Requirement External topic.
5.5.2 Requirement Elements

Separate Requirement elements can be manipulated at the diagram level. These correspond to the 'system level requirements' and can be connected using Realization type connectors to other model elements that 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 can also form a hierarchy, as in the example below.

Tip:
Using Aggregation, Requirements can be connected to show construction of a complete requirements 'tree'.

Note:
External requirements can be created with or without an identifying E in the top right corner of the element. To toggle display of this letter, select or deselect the Show stereotype icon for requirements checkbox on the Options dialog, Objects page (Tools | Options | Objects).

Implementation is managed using Realization connectors, as in the example below:

Once the connectors are established, the Hierarchy window displays the complete requirement implementation / composition details; see the 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.
5.5.3 Color Code External Requirements

External requirements can be color coded to provide quick visual cues indicating the status of a requirement. To enable color coded external requirements follow the steps below:

1. Select the Tools | Options menu option. The Options dialog displays.
2. From the hierarchical tree select Objects, and select the Show status colors on diagrams checkbox to enable the status of external requirements to be represented by color coding.

The color code requirements use the following default conventions:

- Yellow for Proposed
- Blue for Approved
- Green for validated
- Orange for Mandatory
- Black for Implemented.

You can change these colors, and add or remove status types, using the Status Types dialog.
5.5.4 Internal Requirements

Internal requirements in Enterprise Architect are Class (or any element) Responsibilities. In the example below an internal responsibility to enable the user to login to the system has been defined for the Login Use Case. This is a responsibility of the Use Case - an 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 implements either in full or in part.
5.5.5 Make Internal Requirement External

Elements in Enterprise Architect have internal requirements, or responsibilities (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.

Procedure

If you have defined an internal requirement for an element and want to move it out (where it can perhaps be implemented by multiple elements), follow the steps below:

1. Double-click on the element in a diagram or in the Project Browser to open the element Properties dialog.
2. Click on the Require tab.
3. Locate and highlight the requirement.
4. Click on the Move External button.
5. Select the package to place the new requirement in.
6. Click on the OK button.

Enterprise Architect creates a new Requirement element in the target package and automatically generates a Realization connector from the element to the requirement.
Notice the requirement is now marked external and the dialog fields grayed out. To edit its details, double-click on the requirement.

Also notice that a Requirement element has been created in the target package.
5.5.6 Requirement Properties

To edit the properties of a Requirement, double-click on the element in a diagram or right-click on it in the Project Browser and select the Properties menu option. 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.

Type in a short description of the requirement, then type a more detailed explanation of the Requirement in the Notes field at the bottom of the dialog. Set the Status, Difficulty, Priority, and other parameters as required.

When you have finished entering the requirement properties, click on the OK button.

The system needs to be designed to cope with the distribution process for regular retail sales cut-offs.
5.5.7 Implementation

Requirements are implemented by model elements, such as Use Cases, Classes, Interfaces and Components. You can specify this relationship in Enterprise Architect using the Realization connector. A model element is marked as ‘Realizing’ a requirement. Once this connector exists, Enterprise Architect displays the Requirement in the Require tab of the element Properties dialog, in the Requirements Hierarchy window and in the Dependency and Implementation reports, as well as in the standard RTF output.

A quick method of generating a Realization connector is to drag a Requirement element from the Project Browser into a diagram, over the implementing element. Enterprise Architect interprets this as a request to create the Realization connector and does so automatically. To confirm this, perform the action and then go to the Require tab of the target element. There should now be an external relationship to the requirement that was dragged over the target.

Note:

External requirements can be created with or without an identifying E in the top right corner of the element. To toggle display of this letter, select or deselect the Show stereotype icon for requirements checkbox on the Options dialog, Objects page (Tools | Options | Objects).
5.5.8 Composition

Requirements that are connected by Aggregation relationships form a composition hierarchy. High level Requirements can 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.
5.5.9 **Requirements Hierarchy**

Requirements can be connected 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.

For further information, see the [Hierarchy](#) topic.
5.6 Relationship Matrix

The Relationship Matrix is a spreadsheet display of relationships between model elements within packages. You select a source package and a target package, the relationship type and direction, and Enterprise Architect identifies all the relationships between source and target elements by highlighting a grid square and displaying an arrow indicating the direction of the relationship.

**Note:**

The direction is a reflection of which elements are the source elements and which are the target. It does not indicate the Direction property of the connector, as defined in the connector Properties dialog.

The Relationship Matrix is a convenient method of visualizing relationships quickly and definitively. It also enables you to create, modify and delete relationships between elements with a single mouse click - another quick way to set up complex sets of element relationships with a minimum of effort.

If you click on a square in the matrix, the square and the row and column headers are highlighted, as shown in the example above. The example also illustrates the ‘bent arrow’ icon, indicating that connectors exist in both directions between the source and target elements.

The relationship squares in the example are green. This indicates that the source element is not locked. If the element is locked the highlight is pink, as follows:

For information on accessing the Relationship Matrix, see the [Open the Relationship Matrix](#) topic.

You can also:

- **Select options** for modifying the type of information the Relationship Matrix displays
- **Update, delete and create** relationships through the Relationship Matrix
- **Export the contents** of the Relationship Matrix to a CSV file
What Is UML Modeling?  | Relationship Matrix

- **Print the contents** of the Relationship Matrix
- **Save a profile** of the Relationship Matrix settings to monitor development of the same source and target packages
- **Investigate the Source and Target elements** in the relationship.
5.6.1 Open the Relationship Matrix

To open the Relationship Matrix you can:

- Select the **View | Relationship Matrix** menu option
- Click on the **Relationship Matrix** icon in the **Other Views** toolbar
- Right-click on any package in the **Project Browser**, and select the **Documentation | Open in Relationship Matrix | As Source** or **As Target** menu option.

Once the **Relationship Matrix** opens you can:

- **Set the source and target packages**
- **Select which element type to show**
- **Select connector type and direction to show**

The **Relationship Matrix** refreshes after every change you make to the input parameters.

**Tip:**

The **Relationship Matrix** includes 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.
### 5.6.2 Set Element Type

The Relationship Matrix can show all element types, or you can specify which type to show.

To set the element type, follow the steps below:

1. Click on the **Type** drop-down arrow for the Source or Target package.
2. Find the required connector and click on it. Enterprise Architect refreshes the Relationship Matrix content.

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Action</td>
</tr>
<tr>
<td></td>
<td>ActionPin</td>
</tr>
<tr>
<td></td>
<td>Activity</td>
</tr>
<tr>
<td></td>
<td>ActivityParameter</td>
</tr>
<tr>
<td></td>
<td>ActivityPartition</td>
</tr>
<tr>
<td></td>
<td>ActivityRegion</td>
</tr>
<tr>
<td></td>
<td>Actor</td>
</tr>
<tr>
<td></td>
<td>Artifact</td>
</tr>
<tr>
<td></td>
<td>Association</td>
</tr>
<tr>
<td></td>
<td>BusinessProcess</td>
</tr>
<tr>
<td></td>
<td>CentralBufferNode</td>
</tr>
<tr>
<td></td>
<td>CentralTopic</td>
</tr>
<tr>
<td></td>
<td>Change</td>
</tr>
<tr>
<td></td>
<td>Class</td>
</tr>
<tr>
<td></td>
<td>ClientPage</td>
</tr>
<tr>
<td></td>
<td>ClientScriptObject</td>
</tr>
<tr>
<td></td>
<td>Collaboration</td>
</tr>
<tr>
<td></td>
<td>CollaborationDoc</td>
</tr>
<tr>
<td></td>
<td>Comment</td>
</tr>
<tr>
<td></td>
<td>Component</td>
</tr>
<tr>
<td></td>
<td>Constraint</td>
</tr>
<tr>
<td></td>
<td>DataBox</td>
</tr>
<tr>
<td></td>
<td>DataObject</td>
</tr>
<tr>
<td></td>
<td>DataStore</td>
</tr>
<tr>
<td></td>
<td>Decision</td>
</tr>
<tr>
<td></td>
<td>DeploymentSpec</td>
</tr>
<tr>
<td></td>
<td>Device</td>
</tr>
<tr>
<td></td>
<td>DiagramFrame</td>
</tr>
<tr>
<td></td>
<td>EndEvent</td>
</tr>
</tbody>
</table>
5.6.3 Set Connector Type and Direction

The Relationship Matrix requires that you set the connector type to report on and the connector direction. To do this, follow the steps below:

1. Click on the Link Type drop-down arrow to display a list of connector types.

   ![Link Type dropdown](link_type_dropdown.png)

2. Scroll through the list and click on the appropriate connector type.
3. Click on the Direction drop-down arrow to display a list of directions.
4. Scroll through the list and click on the appropriate direction.

Enterprise Architect refreshes the Relationship Matrix content.

**Notes:**

- If you set Direction to Both, each relationship is indicated by two arrows - a From-To arrow and a To-From arrow. See the screen illustration in the Relationship Matrix topic.

- The direction is a reflection of which elements are the source elements and which are the target. It does not indicate the Direction property of the connector, as defined in the connector Properties dialog.
5.6.4 Set Source and Target Package

You must set both the source and target packages for the Relationship Matrix before relationships can be displayed.

Tip:

You set the source and target packages AFTER setting the connector and element types/details; as Enterprise Architect 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. Click on the [...] (Browse) button at the end of the Source or Target field.

   **Source**: Req Package
   **Target**: New Customer

2. The Browse Project dialog displays.

3. Select the required package and click on the OK button.
5.6.5 Relationship Matrix Options

You can define local settings for how the Relationship Matrix works.

Click on the Options button on the Relationship Matrix to access the context menu, then click on the Options menu option. The Matrix Options dialog displays.

Select from the following options:

- **Include Source Children** - to recursively include child packages and contents under the Source
- **Include Target Children** - to recursively include child packages and contents under the Target
- **Include All Extended Meta Types** - to include elements that are extensions of a specified meta-type. For example, if there are Block elements (extending Class) in the package, selecting this option and specifying the type Class includes Class and Block elements, and any further derivatives of Block in the matrix.
- **Sort Axes** - to ensure package elements display in alphabetical order
- **Show Package Names** - to hide or show package names in the Relationship Matrix; this is useful for shortening the displayed texts, especially in circumstances where packages have long names
- **Use Element Alias If Available** - to display an element's alias instead of the element name, if an alias has been defined
- **Show Level Numbering If Available** - to reproduce level numbering in the Relationship Matrix, if it is turned on in the Project Browser; see the screen illustration in the Relationship Matrix topic.
5.6.6 Modify Relationships in Matrix

You can modify or delete relationships, or create new relationships, directly from the Relationship Matrix.

To Modify or Delete Relationships

Right-click on a highlighted relationship to open the context menu, and select from the following options:

- View relationship - opens the Properties dialog for the selected relationship
- Source element properties... - opens the Properties dialog for the source element
- Target element properties - opens the Properties dialog for the target element
- Delete relationship.

If you have selected Delete relationship, Enterprise Architect prompts you to confirm this action.

Note:

The Delete relationship option does not work if:

- The source element (that is, the owner) is locked
- You have selected Both in the Direction field - you are effectively trying to delete half a relationship.

If you have selected one of the other options, modify any properties as required, and click on the OK button to save the changes.

To Create a New Relationship

1. Select the required relationship type in the Link Type field.
2. Right-click on the empty intersection of the source row and target column to display the context menu.
3. Select the Create new relationship... option to create a new connector between the two elements.

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).
5.6.7 Export to CSV

The contents of the Relationship Matrix can be exported to a CSV file. This provides a convenient mechanism for moving the matrix data to a spreadsheet environment such as Microsoft Excel. (This option is also active in the 'Lite', read-only version of Enterprise Architect.)

To export to CSV, follow the steps below:

1. Click on the Options button on the Relationship Matrix to display the context menu.
2. Select the Matrix | Export to CSV menu option. The Windows Browser dialog displays.
3. Browse to the required file location and type in a .CSV filename to export to.
4. Click on the Save button to export the data.
5.6.8 Print the Matrix

You can print a WYSIWYG representation of the Relationship Matrix using the current printer settings. Click on the Options button on the Relationship Matrix to display the context menu.

- To print the matrix select the Matrix | Print menu option.
- To preview prior to printing, select the Matrix | Print Preview menu option.
5.6.9 Matrix Profiles

To save a certain Relationship Matrix configuration as a named profile for later recall, follow the steps below:

1. Set up the Relationship Matrix as required, with source and target, element types and relationship types.
2. Click on the Options button on the Relationship Matrix to display the context menu, then select the Profiles | Save as New Profile menu option.
3. In the Enter Value field, type a profile name of up to 12 characters. Click on the OK button.

Once you have created a profile, you can select it from the drop-down list in the Profile field at the top of the Relationship Matrix screen.
5.6.10 Review Source and Target Elements

As well as providing information on connectors and relationships, the Relationship Matrix enables you to obtain information on the source and target elements in a relationship.

To help you quickly identify which source or target elements have relationships with a particular element, you can highlight the entire row or column for the element by clicking on the element name in the row or column titles. If the list of elements is long, you can scroll across or down the highlighted row or column and quickly identify where the relationships are.

If you right-click on an element name in the row or column titles, the following context menu displays:

- Display the Properties dialog for the selected element
- Display either the only diagram in which the element is used, with the element highlighted, or a list of the diagrams in which the element is used; you then select and open a diagram from the list
- Locate and highlight the element name in the Project Browser
- Make the selected element the context or focus in any docked screens or windows that are open, such as the Tagged Values window.
5.7 UML Profiles

What are UML Profiles?

UML Profiles provide a means of extending UML, which enables you to build UML models in particular domains. They are based on additional stereotypes and Tagged Values that are applied to elements, attributes, methods, connectors, connector ends and so on. 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 describes a set of extensions to basic UML model elements to enable accurate modeling of XSD Schemas (see Modeling XML Applications with UML, David Carlson, p. 310).

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 topic. You can import these XML files into Enterprise Architect through the Resources window. Once imported, you can drag and drop Profile elements onto the current diagram. Enterprise Architect attaches the stereotype, Tagged Values and default values, notes and even metatitle 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 and values.

The imported Profile also automatically generates a page of elements and relationships in the Enterprise Architect UML Toolbox.

Note:

To control the appearance of elements, you can also set a default element template. For more information, see the Use Element Templates topic.

Profiles in the Resources Window

The Resources window 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 Enterprise Architect 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 Resources 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 (e.g. Class); a stereotyped attribute is automatically 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, messages and dependencies are added by selecting them in the Project Browser, then clicking on the start element in a diagram and dragging to the end element (in the same manner as adding normal connectors). A stereotyped connector is added.
- Stereotypes that apply to association ends can be added by dragging the connector 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/uml_profiles.htm](http://www.sparxsystems.com/uml_profiles.htm). You can download these and import them into Enterprise Architect. Over time Sparx Systems intend to expand the range of Profiles, the content of each Profile and the degree of customization possible in each Profile.

You can also create your own Profiles to describe modeling scenarios specific to your development environment. For more information see [SDK for Enterprise Architect](http://www.sparxsystems.com/enterprise_architect/sdk.htm).
5.7.1 Use Profiles

This topic describes the use of Profiles for UML modeling. It describes tasks including:

- How to import a UML Profile for use in a model
- Use of Tagged Values in Profiles
- How to add a Profile Connector to a diagram
- How to synchronize tags and constraints

A Technology Developer might create a new Profile, which they can save (export) to disk for future UML models. The processes of creating and exporting a new UML Profile are described in SDK for Enterprise Architect.

5.7.1.1 Import a UML Profile

To import a Profile you must have a suitable Profile XML file, such as the Profiles supplied on the Sparx Systems website at www.sparxsystems.com/uml_profiles.htm. If the Profile includes references to any metatags, they should be in the same directory as the Profile XML file.

Import a Profile

To import a Profile, follow the steps below:

1. Open the Resources window (View | Resources).
2. Right-click on the UML Profiles tree node and select Import Profile from the context menu. The Import UML Profile dialog displays.
3. Locate the XML Profile file to import using the [ ... ] (Browse) button.
4. Set the required import option checkboxes for all stereotypes defined in the Profile; you can select:
   - Element Size - to import the element size attributes
   - Color and Appearance - to import the color (background, border and font) and appearance (border thickness) attributes
   - Alternate Image - to import the metafile image
   - Code Templates - to import the code templates if they exist
   - Overwrite Existing Templates - to overwrite any existing code templates defined in the current project.
5. Click on the Import button. The Profile is added to the UML Profiles folder.

If the Profile already exists, Enterprise Architect prompts you to overwrite the existing version and import the
What Is UML Modeling? | UML Profiles

new one (or cancel). Once the import is complete, the Profile is ready to use. You can expand the Profile and
drag items from it onto a diagram.

When you import a Profile, it automatically creates pages of elements and connectors in the Enterprise
Architect UML Toolbox. Therefore, you can also populate diagrams from this page.

5.7.1.2 Tagged Values in Profiles

Stereotypes within a UML Profile can have one or more associated Tagged Values. When you create an
element based on a UML Profile Stereotype by dragging from the Resources window 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 use of Enterprise Architect 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:

```xml
<TaggedValues>
  <Tag name="mixed" description="Determines whether this element can contain mixed element and character content.
  See the W3C XML Schema recommendation"/>
  <Tag name="modelGroup" description="Overides 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 Profile Elements
page of the Enterprise Architect UML Toolbox onto a diagram), the Tagged Values are added automatically.

![XSD Complex Types](catalog)

- **tags**
  - memberNames = false
  - modelGroup = sequence

Tagged Values that have default values are automatically set and displayed in the element tags section, if
applicable. When you select the element, the Tagged Values 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
(e.g. values="element | attribute | both" default="both") display in the Properties window with a drop list of
enableable values when selected (as in the example below). Where no Value list exists, the tag accepts free
text.
5.7.1.3 Add Profile Connector to Diagram

To add a Profile-based connector to the current diagram, click on the connector in the Resources window, then click on the source element in the diagram and drag it to the target.

You can also drag the connector from the Resources window to the source and use the list box below to select the target.

5.7.1.4 Synchronize Tags and Constraints

When you first create an element, attribute, operation or link from a UML Profile item, you can also create Tagged Values and constraints. Over time you might modify the Tagged Values and constraints associated with a particular element, so the items already created might be missing additional Tagged Values or constraints.
Similarly, you might 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 Resources window.
2. Locate the stereotyped profile element.
3. Right-click on it to display the context menu, and select the Synch Tagged Values and Constraints option. The Synch Profiled Elements dialog displays.
4. Click on the OK button to proceed. The list is populated with the items that have been modified and the changes that were made.
5.7.2 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 requirements (e.g. profiles for XML schema and for business process modeling). This topic provides a quick list of the types of things that can be pre-defined and the characteristics of each.

This topic gives you a reference to:

- Supported Types of stereotype with Tagged Values and/or constraints
- Profile Structure
- Supported Attributes
- Example of the XML file that constitutes a Profile

5.7.2.1 Supported Types

A UML profile is made up of one or more stereotypes that might have Tagged Values and constraints. The table below and the Supported Attributes table define what can be stereotyped and what information must be supplied.

List of All Supported Types in AppliesTo/Apply Node

<table>
<thead>
<tr>
<th>AppliesTo / Apply</th>
<th>Type</th>
<th>Tags</th>
<th>Constraint</th>
<th>Metafile</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;actor&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;package&quot;</td>
<td>Package</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;usecase&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;collaboration&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;class&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;table&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;component&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;node&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;object&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;sequence&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;entity&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;screen&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;GUIElement&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;requirement&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;state&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;activity&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;interface&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;event&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;issue&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;change&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;hyperlink&quot;</td>
<td>Element</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>&quot;attribute&quot;</td>
<td>Attribute</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>&quot;operation&quot;</td>
<td>Operation</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>&quot;association&quot;</td>
<td>Connector</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>&quot;associationEnd&quot;</td>
<td>AssociationEnd</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What Is UML Modeling?  | UML Profiles

### 5.7.2.2 Profile Structure

UML Profiles for Enterprise Architect are distributed in XML format. The file has the following format:

#### General Header Details

```xml
<?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. Can also include tagged values, constraints, metafile and descriptive comments-->-
</UMLProfile>
```

#### Stereotype Definitions

The header is followed by one or more Stereotype definitions; for example:

```xml
<!-- «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>
</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 can have a default value, can be empty and can specify enableable values. Tagged Values are edited in the Properties window of an element, method, attribute or connector.

You can also specify the default size, default comment and Metafile for drawing an element; see the fragment below:

```xml
<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, otherwise the load fails.

Also note how to specify a default comment for an element. All white space between lines is ignored. To force a line break, use the \n character. To force tabs, use \t.

```xml
<Comment>
```
What Is UML Modeling? | UML Profiles

Some text here about how this works

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 works

with comments being imported from the XML description in one long row.

5.7.2.3 Attributes Supported in XML Profile

The table below lists the three main types of object you can define for the main XML element nodes in an XML Profile document. These are the:

- Stereotype, which creates a visible entry in the UML Profile folder in the Resources window
- Tagged Values, which are additional properties that an element or connector support
- Constraints that apply to the model element.

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute</th>
<th>Optional</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>stereotype</td>
<td>name</td>
<td>No</td>
<td>Stereotype name.</td>
</tr>
<tr>
<td></td>
<td>notes</td>
<td>Yes</td>
<td>Notes visible in browser.</td>
</tr>
<tr>
<td>metafile</td>
<td>cx</td>
<td>Yes</td>
<td>Deprecated. Initial x coordinate of element.</td>
</tr>
<tr>
<td></td>
<td>cy</td>
<td>Yes</td>
<td>Deprecated. Initial y coordinate of element.</td>
</tr>
<tr>
<td></td>
<td>_sizeX</td>
<td>Yes</td>
<td>Initial width of the element, in pixels at 100% zoom.</td>
</tr>
<tr>
<td></td>
<td>_sizeY</td>
<td>Yes</td>
<td>Initial height of the element, in pixels at 100% zoom.</td>
</tr>
<tr>
<td></td>
<td>_imageFile</td>
<td>Yes</td>
<td>Location of image file (.wmf).</td>
</tr>
<tr>
<td></td>
<td>_image</td>
<td>Yes</td>
<td>Shape script definition.</td>
</tr>
<tr>
<td>tag</td>
<td>name</td>
<td>No</td>
<td>Tag name.</td>
</tr>
<tr>
<td></td>
<td>description</td>
<td>Yes</td>
<td>A description of the tag; appears in the tag tab and for elements in the Properties window setting notes.</td>
</tr>
<tr>
<td></td>
<td>values</td>
<td>Yes</td>
<td>List of possible values; values separated by '</td>
</tr>
<tr>
<td></td>
<td>default</td>
<td>Yes</td>
<td>A default value; e.g. 'true'.</td>
</tr>
<tr>
<td>constraint</td>
<td>name</td>
<td>Yes</td>
<td>Constraint name.</td>
</tr>
<tr>
<td></td>
<td>type</td>
<td>Yes</td>
<td>Constraint type (e.g. 'pre' for precondition, 'post' for postcondition).</td>
</tr>
<tr>
<td></td>
<td>notes</td>
<td>No</td>
<td>Additional explanatory notes.</td>
</tr>
</tbody>
</table>

5.7.2.4 Example Profile

Below is an example UML Profile showing the structure and use of the file:

```xml
<?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"/>
  <Content>
    <!-- List of stereotypes used in this profile-->
    <Stereotypes>
      <!-- A profile is a list of stereotypes, that apply to elements, connectors and features in a UML model. Stereotypes can have set tagged values, constraints, valid targets, default dimensions. The examples below are a good starting point -->
      <Stereotype name="SimpleStereotype" notes="Place notes about stereotype here">
        <metafile="router.emf"/>
      </Stereotype>
    </Stereotypes>
  </Content>
</UMLProfile>
```

© 1998-2008 Sparx Systems Pty Ltd
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 are automatically added to the target element when created-->
  "true | false" note the use of a " | " to separate values -->
  <TaggedValues>
    <Tag name="hasNamespace" description="Indicates element is bound to namespace">
      <true 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="My Notes"/>
        </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">
              <Tag name="indexedList" description="indicates if indexed or not" values="true | false" default="true"/>
              <Tag name="adjustContraint" description="">
                <Constraint name="constraint1" type="pre" opType="pre" notes="My Notes"/>
              </TaggedValues>
              <Constraint name="constraint12" type="pre" opType="post" notes="My Notes"/>
            </Constraints>
          </TaggedValues>
          <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>
            </Stereotype>
            <!-- Attribute -->
            <Stereotype name="attname" notes="">
              <AppliesTo>
                <Apply type="attribute"/>
              </AppliesTo>
            </Stereotype>
          </Stereotype>
        </Stereotype>
      </Stereotype>
    </Stereotype>
  </Stereotype>
</AppliesTo>
5.8 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 standard stereotypes associated with them.

For further definition of stereotypes, see the OMG UML specification (UML Superstructure Specification, v2.1.1, section 18.3.8, pp. 667-672).

A stereotype is generally displayed as in the example below (where «myStereotype2» is the stereotype).

In some cases the stereotype causes the element to be drawn differently, as below:

A metafile can be associated with the applied stereotype, as in the example below:
New, or customized, stereotypes can be created. Stereotypes can also be associated with new shapes, using either metafiles (image files) and colors or Shape Scripts, to apply non-UML shapes to elements and connectors. For further information on customizing stereotypes and applying Shape Scripts, see SDK for Enterprise Architect.
5.8.1 Apply Stereotypes

Enterprise Architect 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. In the Stereotype field, type the stereotype(s) to apply as a comma-separated list.
2. Click on the drop-down arrow and select the required stereotype from the list.
3. Click on the [...] button to use the Stereotype Selector dialog.

Stereotype: merge stereotype

To apply a stereotype to an element using the Properties window, select any of the following steps:

1. In the Stereotype field, type the stereotype(s) to apply as a comma-separated list.
2. Click on the drop-down arrow and select the required stereotype from the list.
3. Select the browse other stereotypes... option in the drop-down list to use the Stereotype Selector dialog.
5.8.2 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. The appearance of the stereotype is influenced by the stereotype visibility settings on the Diagram Properties dialog.

Select Stereotypes to Apply/Remove

1. On the element or connector Properties dialog, click on the [ ... ] button near the Stereotype field. The Stereotype for:<object type> dialog displays.

2. Click on the Profile drop-down arrow and choose the required stereotype source.

3. In the Stereotypes list, enable or disable the required stereotype by selecting or deselecting the checkbox against it.

4. Click on the OK button to apply the selection.

You can also define a new stereotype to apply to the required construct by clicking on the New... button and entering the name of the new stereotype when prompted.
5.8.3 Stereotype Visibility

You control the visibility of applied stereotypes using three options in the diagram Properties dialog. Select:

- The **Show Element Stereotypes** checkbox to show or hide all element stereotypes in the current diagram
- The **Show Feature Stereotypes** checkbox to show or hide all attribute and operation stereotypes in the current diagram
- 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.

```
<server page, target, metaclass, thread, processor>
AutomotiveDomain::Class1
```

**Use Stereotype Icons** enabled: displays icons for those stereotypes with icons defined. Stereotypes without icons defined are still displayed in the comma-separated string.

```
<metaclass, thread, processor>
AutomotiveDomain::Class1
```
### 5.8.4 Standard Stereotypes

Below is a list of standard element stereotypes (as provided in the `EABase.eap` base model), each enclosed by guillemets (« »):

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Base Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>«access»</td>
<td>Permission</td>
</tr>
<tr>
<td>«become»</td>
<td>Flow</td>
</tr>
<tr>
<td>«call»</td>
<td>Usage</td>
</tr>
<tr>
<td>«copy»</td>
<td>Flow</td>
</tr>
<tr>
<td>«create»</td>
<td>Message</td>
</tr>
<tr>
<td>«derive»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«destroy»</td>
<td>Message</td>
</tr>
<tr>
<td>«document»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«executable»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«facade»</td>
<td>Package</td>
</tr>
<tr>
<td>«file»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«framework»</td>
<td>Package</td>
</tr>
<tr>
<td>«friend»</td>
<td>Permission</td>
</tr>
<tr>
<td>«global»</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>«implementation»</td>
<td>Class</td>
</tr>
<tr>
<td>«implementation»</td>
<td>Generalization</td>
</tr>
<tr>
<td>«import»</td>
<td>Permission</td>
</tr>
<tr>
<td>«instantiate»</td>
<td>Usage</td>
</tr>
<tr>
<td>«invariant»</td>
<td>Constraint</td>
</tr>
<tr>
<td>«library»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«local»</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>«metaclass»</td>
<td>Class</td>
</tr>
<tr>
<td>«parameter»</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>«postcondition»</td>
<td>Constraint</td>
</tr>
<tr>
<td>«powertype»</td>
<td>Class</td>
</tr>
<tr>
<td>«precondition»</td>
<td>Constraint</td>
</tr>
<tr>
<td>«process»</td>
<td>Classifier</td>
</tr>
<tr>
<td>«refine»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«requirement»</td>
<td>Comment</td>
</tr>
<tr>
<td>«responsibility»</td>
<td>Comment</td>
</tr>
<tr>
<td>«self»</td>
<td>AssociationEnd</td>
</tr>
<tr>
<td>«send»</td>
<td>Usage</td>
</tr>
<tr>
<td>«stub»</td>
<td>Package</td>
</tr>
<tr>
<td>«table»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«thread»</td>
<td>Classifier</td>
</tr>
<tr>
<td>«trace»</td>
<td>Abstraction</td>
</tr>
<tr>
<td>«type»</td>
<td>Class</td>
</tr>
</tbody>
</table>
### Stereotype Table

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Base Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;utility&quot;</td>
<td>Classifier</td>
</tr>
</tbody>
</table>
5.8.5 Stereotypes with Alternative Images

You can alter the appearance of elements using stereotypes. If the stereotype has an associated metafile, when the stereotype is applied to a Class or other element that supports alternative graphical format Enterprise Architect then draws the alternative image instead of the standard one.

`Metafiles let you specify physical diagrams in recognisable images - use the Reference/Stereotype dialog to associate metafiles with stereotypes.`
5.9 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 an XML Pattern file. This XML file can then be imported as a UML Resource that can be used in any model.

Sparx-Created GoF Patterns

To get you started with Design Patterns in Enterprise Architect, Sparx Systems provide you with a zip file containing 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. Download this zip file of the Gang of Four Patterns for Enterprise Architect from [www.sparxsystems.com/uml_patterns.html](http://www.sparxsystems.com/uml_patterns.html).
5.9.1 Create a Pattern

To create a Pattern you first must model the Pattern as a standard UML diagram within Enterprise Architect. The following diagram was created from an example in the GoF book *Design Patterns - Elements of Reusable Object-Oriented Software* by Gamma et al.

Notes:
- In the Corporate edition of Enterprise Architect, if security is enabled you must have Manage Diagrams permission to save a diagram as a Pattern.
- If your source diagram contains information flows, the Information Items Conveyed and Information Flows Realized data is not copied into the Pattern.

Save a Diagram as a Pattern

To save a diagram as a Pattern, follow the steps below:

1. Select the Diagram | Save UML Pattern menu option. The Save Diagram as UML Pattern dialog displays.
2. In the **Pattern Name** field, type the Pattern name.
3. In the **Filename** field, type a .XML filename into which to save the Pattern.
4. In the **Category** field, type the Category under which the Pattern should be listed in **UML Patterns** (required).
5. In the **Version** field, type the Pattern version number, and in the **Notes** field type any notes on the Pattern.
6. Select the actions for the elements that are contained in the Pattern by selecting the appropriate checkboxes. These actions are performed when the Pattern is used (for more detail refer to the **Use a Pattern** topic). The available actions are:
   - **Create**: Creates the Pattern element directly without modification
   - **Merge**: Merges the Pattern element with an existing element, enabling 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
   - **Type**: Creates the Pattern element types as an existing element.

**Notes**:
- If your Pattern includes an Object element, you would use **Instance** to set the classifier of the Object to one of the Classes in the diagram onto which you are dropping the Pattern.
- If your Pattern includes a Property (Port or Part) you would use **Type** to set the type of the Property to one of the Classes in the diagram onto which you are dropping the Pattern.

7. To change the name of one of the elements, double-click on the element to display the **Edit** dialog.
   From this dialog you can also add comments detailing the element's purpose.
8. Click on the OK button to save the Pattern. Once saved you can load it into Enterprise Architect as a Pattern in the Resources window.
5.9.2 Import a Pattern

Before using a previously created Pattern file in a UML model, you must first import it into the current UML model; it is then available from the Resources window and optionally from the Enterprise Architect UML Toolbox. To import a UML Pattern you have previously saved, follow the steps outlined below:

1. Select the Resources window.
2. Right-click on the UML Patterns node. The context menu displays.
3. Select the Import UML Pattern menu option. The Select UML Pattern Import Filename dialog displays.
4. Locate the XML file to import.
5. Click on the Open button to import the Pattern.

The imported Pattern is placed in the appropriate category as defined in the XML file. If the category does not already exist under UML Patterns, a new one is created. To download examples of the Gang of Four patterns for Enterprise Architect open the GoF Patterns zip file.
5.9.3 Use a Pattern

Using a Pattern enables you to use items defined in the Pattern with the UML model. Using Patterns enables you to rapidly create template solutions for code structures that perform the same type of task in other situations.

To use a Pattern that you have previously imported into the model, follow the steps below:

1. Open the diagram into which to add the UML Pattern.
2. Select the Resources window.
3. Expand the UML Pattern folder and find the Pattern to add.
4. Either:
   - Right-click on the Pattern and select the Add Pattern to Diagram menu option or
   - Drag and drop the Pattern from the Resources window onto the diagram.

(You can also view the Pattern details in read-only mode by selecting the View Pattern Details menu option.)

The Add Pattern dialog displays.

<table>
<thead>
<tr>
<th>Panel</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview</td>
<td>Display a preview of the Pattern; click on the Preview link to open a view of the Pattern and drag the sides into as large a picture as you require.</td>
</tr>
<tr>
<td>Pattern Elements</td>
<td>Access the individual elements contained in the Pattern. From here you can:</td>
</tr>
<tr>
<td></td>
<td>- select the action for the individual element (Create, Merge, Instance or Type, as applicable for each element) by clicking on the drop-down arrow, or</td>
</tr>
<tr>
<td></td>
<td>- modify the default of the Pattern element or - for a merged element - choose the namespace, by clicking on the [...] button on the right of the Default entry.</td>
</tr>
<tr>
<td>Element Notes</td>
<td>Display the comments that describe the element in the Pattern. Highlight an element in the Pattern Elements panel to view the notes.</td>
</tr>
</tbody>
</table>

5. Once the appropriate selections have been made, click on the OK button to import the Pattern into the model, recreating the original diagram with new GUIDs.
Change Pattern Element Default

To change the default of the Pattern element, follow the steps below:

1. From the Add Pattern dialog select the individual element in the Pattern Element panel.
2. Click on the [...] button to display the Edit dialog. The specific method for changing the element name is dependant upon the entry in the Action column of the Pattern Elements panel.
3. If the Action entry is Create, then in the Default field in the Edit dialog delete the existing value and type your own, user-defined value. Click on the OK button. The element default is updated on the Add Pattern dialog.
4. If the Action entry for the element is Merge, in the Edit dialog click on the [...] button to browse to an existing element classifier. The Set Element Classifier dialog displays.
5. Select an existing element classifier from the Classifier list. You can restrict the number of choices by selecting the elements from a specific namespace; to do this, click on the In Namespace drop-down arrow and select a namespace. For more information regarding setting element classifiers see the Using Classifiers topic.
The Model Driven Generation (MDG) Technologies enable you to access and use resources pertaining to a specific technology in Enterprise Architect. You have various options for bringing MDG Technologies into use with Enterprise Architect:

- Sparx Systems already provide some in the Enterprise Architect Install directory, such as ICONIX, BPMN 1.4, Data Flow Diagrams, and Mind Mapping; you can see which technologies are available using the MDG Technologies dialog; these are available across Enterprise Architect.

- Sparx Systems provide other MDG Technologies for download from www.sparxsystems.com/resources/mdg_tech/, which you can add to those in your Enterprise Architect Install directory; these are available across Enterprise Architect.

- You can access and activate MDG Technologies remote from Enterprise Architect, in system folders or web sites; these are available across Enterprise Architect.

- You can import UML Profiles, UML Patterns, code templates and language types from elsewhere to be contained in a single area that you can easily access through the Resources window; these are available only within the model in which you imported them.

- Technology Developers can create new MDG Technologies and deploy them to the project team as appropriate; For more information see SDK for Enterprise Architect.

Having made the MDG Technologies available to Enterprise Architect, you can manage their availability to users and you can work with them.
5.10.1 Import MDG Technologies

To import an MDG Technology you must have a suitable MDG Technology XML file. If the MDG Technology includes references to any metafiles, they should be in the same directory as the MDG Technology XML file.

An imported MDG Technology is available only within the model into which it has been imported, not in every model you have in Enterprise Architect. To make the MDG Technology available across all your models, download it into the Enterprise Architect install directory.

Import an MDG Technology

To import an MDG Technology, follow the steps below:

1. Select the Tools | Import Technology menu option. The Import Technology dialog displays.

2. In the Filename field, type the path and filename of the MDG Technology file to import, or browse for it using the [...] button.

   **Note:**

   When you enter the filename, the MDG Technology name displays in the Technology field and the option checkboxes become available. Any options that remain grayed out indicate that no examples of that type exist in the MDG Technology XML file.

3. All option checkboxes default to selected. Clear those against resources you do not want to import, and leave selected the checkbox against each of the resources to import. Leave selected:
   - Patterns, to import patterns, if they exist
   - Images, to import graphics
   - Profiles, to import profiles, if they exist
   - Element Size, to import the element size attributes
   - Alternate Image, to import the metafile image
   - Tagged Values, to import Tagged Values
   - Color and Appearance, to import the color (background, border and font) and appearance (border thickness) attributes
   - Code Modules, to import the various languages associated with the technology, if they exist
   - Data Types, to import the data types
   - Code Templates, to import the code templates, if they exist
   - Code Options, to import the options that include items such as default file extensions and default file paths.

4. Click on the Import button.
If the MDG Technology already exists, Enterprise Architect displays a prompt to overwrite the existing version and import the new one.

Once the import is complete, the MDG Technology is listed in the MDG Technologies folder of the Resources window and in the MDG Technologies dialog.
5.10.2 Manage MDG Technologies

You use the MDG Technologies dialog to manage the MDG Technologies available and accessible to Enterprise Architect users. To display this dialog, select the Settings | MDG Technologies menu option.

![MDG Technologies Dialog](image)

The MDG Technologies dialog lists the technologies held in the Enterprise Architect Install directory (available in all models), and those imported into the Resources window for the current model.

Enable and Disable MDG Technologies

All MDG Technologies listed can be made available (enabled) or removed from use (disabled). To enable or disable a Technology, click on its Enabled checkbox.

When an MDG Technology is enabled, three things happen:

- The MDG Technology is added to the list of available options in the profile field of the Default Tools toolbar, so that you can apply the interface profiles of the MDG Technology.
- At least one set of Toolbox pages for the MDG Technology is automatically added to the Enterprise Architect UML Toolbox; you can access the added Toolbox pages through the More Tools menu.
- Any MDG Technology-specific diagram templates are added to the New Diagram dialog for selection; when selected, these display the diagram-specific Toolbox pages.

**Note:**

Whilst you have to enable an imported MDG Technology to access its Toolbox groups and interface profile, you do not have to enable it in order to drag its objects from the Resources window onto diagrams.

You can quickly enable or disable all the listed MDG Technologies by clicking on the All or None buttons.

Set as Default

You can make an MDG Technology the default interface to Enterprise Architect. Depending on the MDG Technology selected, this can change the way Enterprise Architect windows are displayed and override the Enterprise Architect UML Toolbox pages with pages specific to that Technology.

To set an MDG Technology as the default interface, click on it in the Technology panel and click on the Set Active button.
What Is UML Modeling? | MDG Technologies

This displays an asterisk against the MDG Technology name in the Technology panel, and selects the MDG Technology in the profile field of the Default Tools toolbar. If the MDG Technology has not been enabled, this also enables it.

MDG Technologies Outside Enterprise Architect

The MDG Technologies dialog lists technologies that have been loaded into the Enterprise Architect install directory or imported into the Resources window. You can also add MDG Technologies in folders and websites remote from Enterprise Architect. To do this, click on the Advanced button. See the Access Remote MDG Technologies topic.

5.10.2.1 Access Remote MDG Technologies

You can access MDG Technologies in folders and websites remote from Enterprise Architect.

If you have not already identified the location of the MDG Technology, you must first do this. You can then select the MDG Technology for use.

Later, if you have no further use for the MDG Technology, you can remove it from the list of identified MDG Technologies.

Note:

If you add or remove remote MDG Technologies, you must restart Enterprise Architect to show them on or remove them from the list on the MDG Technologies dialog.

Identify Remote MDG Technology

To specify the location of the MDG Technology to access, follow the steps below:

1. Select the Settings | MDG Technologies menu option. The MDG Technologies dialog displays.
2. Click on the Advanced button. The MDG Technologies - Advanced dialog displays.
3. Click on the Add button. A short context menu displays, offering the options:
   - Add Path
   - Add URL
4. To specify an MDG Technology in a directory folder, select the Add Path option. The Browse for Folder dialog displays.
   - Browse for the MDG Technology folder, click on it, and click on the OK button. Go to step 6.
5. To specify an MDG Technology on a web site, select the Add URL option. The Input dialog displays.
6. The folder path or URL for the MDG Technology displays in the Path panel.

**Use Remote MDG Technology**
To access a remote MDG Technology listed in the MDG Technologies - Advanced dialog, double-click on the folder path or URL.

**Remove Listed MDG Technology**
To remove an MDG Technology listed in the MDG Technologies - Advanced dialog, click on the folder path or URL and click on the Remove button. The path or URL is deleted.
5.10.3 Work with MDG Technologies

Any MDG Technology listed on the MDG Technologies dialog can be enabled, which makes their interface profiles and Enterprise Architect UML Toolbox pages available for your use.

When you import an MDG Technology, you can also access its resources immediately through the Enterprise Architect Resources window.

MDG Technology Toolbox Pages

When you enable an MDG Technology, any Technology-specific diagram types are added to the New Diagram dialog lists, and the Technology’s UML Toolbox pages are added to those available through the More tools menus in the Enterprise Architect UML Toolbox.

If you set the MDG Technology to Active, its Toolbox pages override any parallel Enterprise Architect UML Toolbox pages. For example, the ICONIX Class pages would override the Enterprise Architect Class pages.

You create Technology-specific diagrams and populate them with elements and connectors in the same way as for standard Enterprise Architect diagrams.

The Resources Window

The Resources window (View | Resources) displays a tree structure containing nodes such as imported MDG Technologies, Documents, Stylesheets, Matrix profiles and UML Profiles.
MDG Technologies can bundle the functionality provided by UML Profiles, UML Patterns, Code Templates and Model Types.

Profiles contained in MDG Technologies are applied to:

- Elements such as Classes and Interfaces, which are dragged directly from the Enterprise Architect UML Toolbox or the Resources window to the current diagram.
- Attributes, which are dragged over a host element (e.g. Class) to be automatically added to the element feature list.
- Operations which, like Attributes, are dragged over a host element to add the operation.
- Connectors such as Association, Generalization, and Dependency, which are added by selecting them in the Toolbox or Resources window, then clicking on the source element in a diagram and dragging to the target element (in the same way as adding normal connectors); the connector is added with the new stereotype and Tagged Value information.
- Association Ends, which are added by dragging the connector end element over the end of an Association in the diagram.

Patterns contained in MDG Technologies are used to:

- Enable reuse in a model.
- Build in robustness.

Code Templates are used to:

- Specify the transformation from UML elements into various parts of a given programming language.

Model Types are used to:

- Define the data types for the model.
5.10.4 Mind Mapping

The following text is derived from the Mind Map entry in the online Wikipedia.

A Mind Map is a diagram used to represent words, ideas, tasks or other items linked to and arranged radially around a central key word or idea. It is used to generate, visualize, structure and classify ideas, and as an aid in study, organization, problem solving, decision making, and writing.

A Mind Map is an image-centered diagram that represents semantic or other connections between portions of information. By presenting these connections in a radial, non-linear graphical manner, it encourages a brainstorming approach to any given organizational task, eliminating the hurdle of initially establishing an intrinsically appropriate or relevant conceptual framework to work within.

The elements are arranged intuitively according to the importance of the concepts and are organized into groupings, branches, or areas. The uniform graphic formulation of the semantic structure of information on the method of gathering knowledge, may aid recall of existing memories.

The use of the term Mind Maps is trademarked in the UK and the USA by The Buzan Organization, Ltd.

For further information on the concepts of Mind Mapping, refer to the Wikipedia item and its linked sources.

Mind Mapping in Enterprise Architect

Enterprise Architect enables you to develop Mind Maps quickly and simply, through use of an MDG Technology integrated with the Enterprise Architect installer. The Mind Mapping facilities are provided in the form of:

- A Mind Mapping diagram type, accessed through the New Diagram dialog
- A Mind Mapping page in the Enterprise Architect UML Toolbox
- Mind Mapping element and relationship entries in the UML Toolbox Shortcut Menu and Quick Linker.

Mind Mapping Toolbox Page

You can access the Mind Mapping page of the Toolbox through the More tools | Mind Mapping menu option. The following icons are available:

- Central Topic
- Main Topic
- Topic
- Sub Topic
- Relationship

- Central Topic is the main theme of the Mind Map; you would normally have one or two of these on the diagram, but can add as many as are necessary
- Main Topic represents the immediate concepts generated by the Central Topic
- Topic represents the larger divisions of a Main Topic
- Sub Topic represents the finer divisions of a Topic or Main Topic; you could also have Subtopics of Subtopics to represent increasingly finer distinctions
- Relationship represents the connection between any two elements; you can have several Relationships per element. Each relationship has three anchor points, so you can curve the lines to develop the flow of concepts more easily.

When dragged onto a Mind Mapping diagram, the elements and relationship have the following appearances:
As the elements can represent any concept, object or relationship, you can use the full range of element properties and features to expand on what the element represents, including adding Note elements. However, to preserve the simplicity and readability of the diagram itself, you cannot display the element compartments on the diagram.

**Disable Mind Mapping**

If you prefer not to use Mind Mapping in Enterprise Architect, you can disable it (and subsequently re-enable it) using the MDG Technologies dialog (Settings | MDG Technologies).
5.10.5 Data Flow Diagrams

The following text is derived from the Data flow diagram entry in the online Wikipedia.

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then "exploded" to show more detail of the system being modeled.

Data flow diagrams were invented by Larry Constantine ... based on Martin and Estrin's "data flow graph" model of computation. [They] are one of the three essential perspectives of Structured Systems Analysis and Design Method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a dataflow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's dataflow diagrams to draw comparisons to implement a more efficient system.

Developing a DFD helps in identifying the transaction data in the data model.

For further information on the concepts of Data Flow Diagrams, refer to the Wikipedia item and its linked sources.

Data Flow Diagrams in Enterprise Architect

Enterprise Architect enables you to develop Data Flow diagrams quickly and simply, through use of an MDG Technology integrated with the Enterprise Architect installer. The Data Flow diagram facilities are provided in the form of:

- A Data Flow diagram type, accessed through the New Diagram dialog
- A Data Flow Diagram page in the Enterprise Architect UML Toolbox
- Data Flow element and relationship entries in the UML Toolbox Shortcut Menu and Quick Linker.

Data Flow Diagram Toolbox Page

You can access the Data Flow Diagram page of the Toolbox through the More tools | Data Flow Diagrams menu option. The following icons are available:

- **Process** is a process or activity in which data is used or generated
- **External** represents an external source, user or depository of the data
- **Data Store** represents an internal physical or electronic repository of data, into and out of which data is stored and retrieved
- **Data Flow** (connector) represents how data flows through the system, in physical or electronic form.

When dragged onto a Data Flow diagram, the elements and relationship have the following appearances:
To preserve the simplicity and readability of the diagram, you cannot display the element compartments on the diagram.

**Context Diagram**

A *Context* diagram is a top-level Data Flow diagram that has just one Process element representing the system being modeled, showing its relationship to external systems.

**Disable Data Flow Diagrams**

If you prefer not to use Data Flow Diagramming in Enterprise Architect, you can disable it (and subsequently re-enable it) using the MDG Technologies dialog (Settings | MDG Technologies).
5.10.6 ICONIX Process

The following text is derived from the ICONIX entry in the online Wikipedia.

The ICONIX Process is a minimalist, streamlined approach to Use Case driven UML modeling that uses a core subset of UML diagrams and techniques to provide thorough coverage of object-oriented analysis and design. Its main activity is robustness analysis, a method for bridging the gap between analysis and design. Robustness analysis reduces the ambiguity in use case descriptions, by ensuring that they are written in the context of an accompanying domain model. This process makes the use cases much easier to design, test and estimate.

The ICONIC Process was developed by D. Rosenberg. For more information on the ICONIX Process, see the ICONIX Software Engineering Inc. website http://iconixsw.com/.

ICONIX Process in Enterprise Architect

Enterprise Architect enables you to develop models under the ICONIX Process quickly and simply, through use of an MDG Technology integrated with the Enterprise Architect installer. The ICONIX facilities are provided in the form of:

- ICONIX pages in the Enterprise Architect UML Toolbox
- ICONIX element and relationship entries in the UML Toolbox Shortcut Menu and Quick Linker.

To further help you develop and manage a project under the ICONIX Process, Enterprise Architect also provides a white paper on the ICONIX Roadmap.

In addition, Enterprise Architect has an alternative visual layout specific to the ICONIX Process.

ICONIX Toolbox Pages

Within the Toolbox, Enterprise Architect provides ICONIX versions of the pages for UML Analysis, Use Case, Class, Interaction (Sequence), Activity and Custom diagrams (which often form the basis for Robustness diagrams). Compared to the standard Enterprise Architect Toolbox pages, these have slightly different element and relationship sets. You can access them by either:

- Selecting the More tools | ICONIX | <Diagram Type> menu option for a specific Toolbox page, or
- Selecting the ICONIX option in the drop-down field of the Default Tools toolbar, which adds all six pages to the Toolbox. The first page and the Common page are expanded, and the others are closed up.
The ICONIX layout re-organizes the Enterprise Architect work area, opening the:

- **Toolbox** on the right hand side of the screen (follow the instructions above to display the ICONIX pages)
- **Project Browser** and **Model Views** windows nested in the top left of the screen
- **Tasks Pane** and **Resources** window autohidden on the top left of the screen
- **Pan & Zoom, Notes** and **Properties** windows nested on the bottom left of the screen, and
- **System** and **Output** windows autohidden on the bottom left of the screen.
To apply this layout, select the View | Visual Layouts menu option.

**Disable ICONIX**

If you prefer not to use the ICONIX Process in Enterprise Architect, you can disable it (and subsequently re-enable it) using the MDG Technologies dialog (Settings | MDG Technologies).

This does not affect the ICONIX layout, which you can switch back to your own layout or the Enterprise Architect default layout using the View | Visual Layouts menu option.
5.10.7 BPMN 1.4

The following text is derived from the Business Process Modeling Notation entry in the online Wikipedia.

The Business Process Modeling Notation (BPMN) is a standardized graphical notation for drawing business processes in a workflow. BPMN was developed by Business Process Management Initiative (BPMI), and is now being maintained by the Object Management Group since the two organizations merged in 2005.

The primary goal of BPMN is to provide a standard notation that is readily understandable by all business stakeholders. These business stakeholders include the business analysts who create and refine the processes, the technical developers responsible for implementing the processes, and the business managers who monitor and manage the processes. Consequently BPMN is intended to serve as common language to bridge the communication gap that frequently occurs between business process design and implementation.

... The adoption of BPMN standard notation will help unify the expression of basic business process concepts (e.g., public and private processes, choreographies) as well as advanced modeling concepts (e.g., exception handling, transaction compensation).

BPMN ... supports only the concepts of modeling that are applicable to business processes ... other types of modeling done by organizations for non-business purposes [are] out of scope for BPMN. For example, ... modeling ... the following is not a part of BPMN:

- Organizational structures
- Functional breakdowns
- Data models

In addition, while BPMN shows the flow of data (messages) and the association of data artifacts to activities, it is not a data flow diagram.

For further information on the concepts of BPMN, refer to the Wikipedia item and its linked sources.

**Note:**

Version 1.4 of MDG Technology for BPMN is integrated with the Enterprise Architect installer for release 7.0 and above. This configuration does not include model validation. You can also install MDG Technology For BPMN version 1.3 separately, if you require the model validation component. However, this version is intended for - and only correctly integrates with - release 6.5 of Enterprise Architect. If you intend to install MDG Technology for BPMN 1.3, do not load it through the BPMN Add-In menu.

**BPMN in Enterprise Architect**

The BPMN notation is specifically targeted at the business modeling community and has a relatively direct mapping to UML through a BPMN Profile. Enterprise Architect enables you to develop BPMN diagrams quickly and simply, through use of the BPMN Profile integrated with the Enterprise Architect installer. The BPMN facilities are provided in the form of:

- A BPMN diagram type, accessed through the New Diagram dialog
- BPMN pages in the Enterprise Architect UML Toolbox
- BPMN element and relationship entries in the UML Toolbox Shortcut Menu and Quick Linker.

**BPMN Toolbox Pages**

You can access the BPMN pages of the Toolbox through the More tools | BPMN menu option. These pages provide the graphical (Core) and non-graphical (Types) BPMN elements for use on business process diagrams.

Specifications of these elements and relationships are defined by Tagged Values (for example, to define the Message, Timer and Default Path (/) symbols in the diagram below).

For further information on BPMN and Tagged Values, see the Change Element Appearance Using Tagged Values topic.
### What Is UML Modeling?

**Business Process**
Extend a composite Activity that defines a business process.

---

**BPMN Core**
- Business Process
- Activity
- Start Event
- Intermediate Event
- End Event
- Gateway
- Pool
- Lane
- Data Object
- Group
- Text Annotation

**BPMN Relationships**
- Sequence Flow
- Message Flow
- Association

**BPMN Types**
- Message
- Participant
- Rule
- Transaction
- Web Service
- Property

---

<table>
<thead>
<tr>
<th>Item</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process</td>
<td>Extend a composite Activity that defines a business process.</td>
</tr>
</tbody>
</table>
### Item | Use to
--- | ---
**Activity** | Define an activity within a business process.
**Start Event** | Define the initiating event in a process.
**Intermediate Event** | Define an intermediate event in a process.
**End Event** | Define the terminating event in a process.
**Gateway** | Define a decision point in a business process. If a condition is true, then processing continues one way; if not, then another.
**Pool** | Extend a *Partition* element to logically organize an Activity.
**Lane** | Extend a *Partition* element to subdivide a Pool.
**Data Object** | Extend an *Artifact* element to define a physical piece of information used or produced by a system.
**Group** | Extend a *Boundary* element to group other elements.
**Text Annotation** | Create a comment.
**Sequence Flow** | Extend a *Control Flow* relationship to define the flow of activity.
**Message Flow** | Extend a *Control Flow* relationship to define the flow of communications in the process.
**Association** | Associate information and artifacts with flow objects.
**Message** | Extend a *Class* element to define a message.
**Participant** | Extend a *Class* element to define a participant in an activity.
**Rule** | Extend a *Class* element to define rule statements.
**Transaction** | Extend a *Class* element to define a transaction in an activity.
**Web Service** | Extend a *Class* element to define a web service.
**Property** | Extend an attribute to drag onto another element.

#### 5.10.7.1 Change Element Appearance

To define the specifications of BPMN elements and relationships, open the Tagged Values window and select the required element or relationship in a diagram. The Tagged Values window shows the appropriate Tagged Values and provides a list of values to assign to each one.

Some Tagged Values directly affect the appearance of the elements they apply to, as described in the following examples:

- **Events** - to change the decoration of a Start Event or Intermediate Event, set the *Trigger* Tagged Value; to change the decoration of an End Event, set the *Result* Tagged Value. For example, to create a BPMN 'off-page' connector, set the *Trigger* or *Result* Tagged Value to *Link* to depict flow onto the diagram, into a branch diagram or off the diagram for the Start, Intermediate and End events respectively.
- **Gateways** - to create the different varieties of Gateway, set the *GatewayType* Tagged Value; other display options are available for XOR gateways - you can set the *XORType* Tagged Value to create Event-based or Data-based XOR gateways, and for Data-based XOR gateways you can set the *MarkerVisible* Tagged Value to *false* to hide the decoration.
- **Activities** - there is a wide variety of appearance options for Activities:
  - The *ActivityType* Tagged Value can be set to *Task* or *Sub-Process*; the latter option displays the 'plus-in-a-box' decoration on the bottom edge of the shape
  - An Ad-hoc Activity is shown by setting the *AdHoc* Tagged Value to *true*; this displays the 'tilde' decoration on the bottom edge of the shape
  - A Compensation Activity is shown by setting the *IsCompensation* Tagged Value to *true*; this displays the 'rewind' icon on the bottom edge of the shape
  - A Multiple Instance Activity is shown by setting the *IsMultipleInstance* Tagged Value to *true*; this displays the 'pause' icon on the bottom edge of the shape
  - A Loop Activity is shown by setting the *LoopType* Tagged Value to either *Standard* or *MultiInstance*, this displays the 'loop' icon on the bottom edge of the shape.
- Transactions - to denote a Transaction with a double-lined border, set the IsATransaction Tagged Value to true.
- Sequence Flows - to put a diagonal slash across the line at the source end, set the ConditionType Tagged Value to Default; to put an unfilled diamond-shaped decoration at the source end, set the Tagged Value to Expression.
5.11 Business Modeling

Modeling the Business Process
Modeling the business process is an essential part of any software development process. It enables 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 fits into the organizational structure and daily activities. It can also provide the justification for building the system by capturing the current manual and automated procedures that are to be rolled up into a new system, and the associated cost benefit.

As an early model of business activity, it enables 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 connectors, 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 enables the analyst to clearly map what is in the scope of the proposed system and what is to be implemented in other ways (e.g. a manual process).

An Example
The example below demonstrates the kind of model that can 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 and shipping company, and completes when an order is delivered to the customer.

See
- Process Modeling Notation
- Inputs, Resources and Information
- Events
- Outputs
- Goals
- A Complete Business Process

See Also
- Business Modeling and Business Interaction Diagrams
- Web Stereotypes
5.11.1 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:

- Can affect more than one organizational unit
- Can have a horizontal organizational impact
- Creates value of some kind for the customer; customers can be internal or external.

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 organization, 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 can be placed inside the process element.

The BPMN Profile

One popular notation and approach to business modeling is the Business Process Modeling Notation (BPMN). This notation is specifically targeted at the business modeling community and has a relatively direct mapping to UML through a BPMN Profile. Sparx Systems provides a built-in UML profile for BPMN modeling in Enterprise Architect.
5.11.2 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 can come from external sources, from customers, from internal organizational units and could 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 connector indicates that the information or object linked to the process is not used up in the processing phase. For example, order templates can 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 connector 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).
### 5.11.3 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 might be consumed and transformed (for example a customer order) or simply act as a catalyst (for example, nightly batch job).
5.11.4 Outputs

A business process typically produces one or more outputs of value to the business, either for internal use or to satisfy external requirements. An output might 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 might feed into another process, either as a requested item or a trigger to initiate new activities.

An Output connector indicates that the business process produces some object (either physical or logical) that is of value to the organization, either as an externally visible item or as an internal product (possibly feeding another process).
5.11.5 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 requirements.

A Goal connector indicates that the attached object to the business process describes the goal of the process. A goal is the business justification for performing the activity.
5.11.6 A Complete Business Process

The diagram below illustrates how the various model elements can be grouped together to produce a coherent picture of a named business process. Included are the inputs, outputs, events, goals and other resources that are of significance.
What is a UML Model?

A Model is a special type of package and belongs to a collection of packages; it has a ParentID of 0. This is the top level entry point to an Enterprise Architect project file. Each model is a root node in the Project Browser and can contain items such as Views and packages. The model contains the diagrams, elements, relationships and associated metadata that define the structure and function of a system or process. These components are organized into a hierarchy of packages, which help to group and manage related components. By iterating through all models, you can access all the elements within the project hierarchy. You can also use the Add New function to create a new model. A model can also be deleted, but remember that everything contained in the model is deleted as well.

In Model Management, you configure and maintain:

- Project files and data repositories
- Project sharing in team environments
- Version Control of model packages
- User Security in updating model packages and elements
- Auditing of model changes
- Baseline and differencing to capture and roll back changes
- The Traceability of model development
- Use of the Automation Interface and Add-Ins to automate and extend Enterprise Architect functionality
- The transfer of data between projects in similar or different databases
- A Discussion Forum for distributing and discussing information concerning a model or project.

Project Files and Data Repositories

An Enterprise Architect project is stored in a data repository. Enterprise Architect enables you to work with EAP files (a Microsoft JET database). In the Enterprise Architect Corporate edition, you can also work with DBMS repositories such as:

- SQL Server
- MySQL
- Oracle 9i, 10g or 11g
- PostgreSQL
- Adaptive Server Anywhere
- MSDE Server
- Progress OpenEdge

Information on how to get started with projects can be found in the Quick Start - Create a Project topic.

Project Sharing in UML Models

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 enables 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.
Version Control For UML Models

Enterprise Architect UML Model version control enables you to:

- Coordinate sharing of packages between users, with either read-only access or update access
- Save and retrieve a history of changes to packages.

To use version control in Enterprise Architect, you require a third-party source-code control application such as Subversion, CVS, or any other version control product that complies with the Microsoft Common Source Code Control standard.

User Security

**Note:**

This feature is available in the Corporate edition only.

UML Model User Security in Enterprise Architect provides a means of limiting access to update functions in a project. Elements can be locked per user or per group, and a password defined for login. Enterprise Architect offers two security policies:

- Standard, where each element is considered unlocked until specifically locked
- Rigorous, where each element is assumed to be locked until specifically unlocked.

Traceability

Traceability identifies the way a given process has been, or is to be, developed in a system. The process can be an internal, model-management process, where you monitor work by asking questions such as 'what work has been done to realize this Requirement or Use Case?', or a business or system process that is being modeled, where you ask questions such as 'what Requirements, Use Cases, Classes, Components, Test Cases and other elements define the implementation and deployment of this process?'

Audit UML Models

Auditing is a project-level feature, available in the Corporate Edition, that enables you to record model changes in Enterprise Architect. By enabling this option, model administrators can view a range of information regarding changes, such as:

- Who changed an element
- How many elements they changed
- When they changed the data
- What the previous values were, and
- What type of elements they changed.

Baselines and Differences

The Enterprise Architect Corporate edition provides a facility to Baseline or snapshot a model branch in XMI format at a particular point in time, and store it within the model in compressed format. More than one baseline can be stored against a single Enterprise Architect package. Using Baselines, you can compare packages at the current and earlier stages of development, using the Compare (Diff) utility. The Compare utility is available in the Professional and Corporate editions of Enterprise Architect. It enables you to compare the current model with either an exported or a version-controlled Enterprise Architect XMI file on disk, as well as with a Baseline.

Project Data Transfer

**Note:**

This feature is available in the Corporate edition only.

Enterprise Architect enables you to transfer project data between project data repositories, row by row, table by table.

The Automation Interface

The Enterprise Architect Automation Interface provides a way of accessing the internals of Enterprise
Architect models to, for example, perform repetitive tasks or produce custom reports. All development environments capable of generating ActiveX Com clients, such as Microsoft C# or Java, should be able to connect to the Automation Interface.

**Add-Ins**

Add-Ins are ActiveX COM objects that expose public Dispatch methods. The Enterprise Architect Add-In model builds on the features provided by the Automation Interface to enable you to extend the Enterprise Architect user interface and add functionality.

**Project Discussion Forum**

Enterprise Architect provides a Project Discussion Forum which can be used to discuss the development and progress of a project or model. You can switch the forum to other projects, so you can monitor and compare developments in several projects at once.

**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, Enterprise Architect also enables projects to be shared using replication. Replication is a simple process that enables data interchange between EAP based repositories and is suitable for use in situations where many different users work independently. Modelers merge their changes into a Design Master only as required. It is recommended that a backup is carried out prior to replication.

Replication requires the use of EAP based repositories, and cannot be performed on repositories stored on a DBMS server.

**See Also**

- Upgrading Models
- Project Data Integrity
- Setting Up a Database Repository
- Model Maintenance
- Manage Views
- XMI Import and Export
- Team Development
- Spell Checking
- Reference Data
6.1 Enterprise Architect Project Files

An Enterprise Architect project is stored in a data repository. In Enterprise Architect Desktop and Professional editions, you work with a single file having a .EAP extension. In Enterprise Architect Corporate edition you can use a suitable DBMS database for project files.

Project Files

.EAP Files

In Enterprise Architect Desktop and Professional editions, a single file with a .EAP extension is used to store projects. A .EAP file is a Microsoft JET database, so you can also open it using MS Access or any other reporting tool that can work with JET databases.

DBMS Repositories

In Enterprise Architect Corporate edition, you can use a suitable DBMS database for project files. DBMS project files have the same logical structure as .EAP files, but must be connected to using ADO/ODBC. See Connect to a Data Repository, below.

Whenever you launch Enterprise Architect, the first thing displayed is the Start Page. From here, you can create a new project, open a project and (Corporate edition only) connect to a data repository.

Create a New Project File

On creating a new project, the Model Wizard enables you to select various model packages.

You can also add models to a project from the Project Browser by:

- Right-clicking on an existing model and selecting the New Model or Add model using Wizard context menu options
- Right-clicking on a package and selecting the Add | Add model using Wizard context menu option
- Clicking on an existing model, pressing [Insert] and selecting the New Model or Add model using Wizard context menu options
- Clicking on a package, pressing [Insert] and selecting the Add model using Wizard context menu option.

Open an Existing Project

There are various ways to open a project in Enterprise Architect. New users are advised to explore the EAExample file supplied with Enterprise Architect.

Connect to a Data Repository

Note:

This feature is available in the Corporate edition only.

Enterprise Architect enables you to connect to a data repository. Enterprise Architect currently supports the following data repositories:

- MS Access (in all editions - .EAP files are stored in Microsoft JET databases)
- SQL Server
- MySQL
- Oracle 9i, 10g or 11g
- PostgreSQL
- MSDE
- Adaptive Server Anywhere
- **Progress OpenEdge**

To create a new data repository, you must first create a new database with the DBMS management software, then run supplied scripts to create the logical structure. You should then use Enterprise Architect data transfer functions to move a project from a .EAP or DBMS model into the new project.
6.1.1 What is a Project?

An Enterprise Architect project is a mechanism for storing and managing the components of one or more UML models.

A project can be a .EAP file in an MS Access database or (in the Enterprise Architect Corporate edition) a structure of files in a database management system such as MySQL or Oracle.

A project can contain a single model, or a number of models, each of which defines a particular aspect of the system or process. You create the model from a template specifically structured to support the aspect that the model represents, such as business process, requirements, or deployment.

A model contains the diagrams, elements, relationships and associated metadata that define the structure and function of a system or process. These components are organized into a hierarchy of packages, which help to group and manage related components.

The top-level packages in a model are Views, which represent partitions of the model that you define yourself. You can start with standard Views such as Class or Component, or create whatever partitions are appropriate to your model.

So a typical project could have a structure something like the following:

```
- Project
  - Development Model
  - Requirements Model
    - Requirements View
    - Use Case View
      - Req Package
      - Use Case Package
        - Use Case Diagram
        - New Customer
        - Existing Customer
      - Use Case 1
      - Use Case 2
```

The project Project contains two models:

- Development Model and
- Requirements Model.

Each model contains Views. Requirements Model contains:

- Requirements View and
- Use Case View.

Each View contains packages. Use Case View contains:

- Req Package and
- Use Case Package.

Each package contains one or more diagrams, one or more packages, and several elements. Use Case Package contains:

- Use Case Diagram
- New Customer package
- Existing Customer package
- Use Case 1 element
- Use case 2 element.

Each subordinate package also contains diagrams, elements and (if necessary) further packages. The
elements are related by connectors created in the diagrams, and each element and connector has properties, attributes, operations and extensions defined in the respective Properties dialog.
6.1.2 Open a Project

Enterprise Architect supports several different methods of opening an Enterprise Architect project file.

From the Main Menu
Select the File | Open Project menu option. From the Open Project dialog, select the path for the file to open and click on the Open button.

From the Start Page
· Click on Open a Project File.
· The Open Project dialog displays.
· Use the file browser ( [... ] ) to navigate to the project to open, which has a .EAP file extension (*.EAP). Select the project and click on the Open button.

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 to open is in the Recent list, simply click once on the name of the project to open it.

Note:
If you already have a project open, Enterprise Architect prompts you to save changes before loading.

Enterprise Architect 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 Enterprise Architect installation directory. The default installation directories, depending on which version you have installed, are:
· Registered version: C:\Program Files\Sparx Systems\EA\EAExample.eap
· Trial version: C:\Program Files\Sparx Systems\EA\EATrial\EAExample.eap
· Lite version: C:\Program Files\Sparx Systems\EA\EALite\EAExample.eap

Connect to a Data Repository
This feature is available in the Corporate edition only.
If you are using the Enterprise Architect Corporate edition, you also have the option to connect to SQL Server, MySQL, Oracle 9i, 10g or 11g, PostgreSQL, ASA, MSDE Server, and Progress OpenEdge data repositories.
6.1.3 Create a New Project

Enterprise Architect projects can be created via the File | New Project menu option, which displays the Save New Enterprise Architect Project dialog. Select a directory and enter a file name for your project, then click on the Save button. Once the project has been saved, the Model Wizard displays. A selection of models are available. Select the models to include and click on the OK button.

Alternatively, new projects can be created from the Start Page; select the Create a New Project option.

The Model Wizard

The Model Wizard is used to add a selection of models to the project.

The EABase Project File

The default project file (EABase.EAP) is supplied when you install Enterprise Architect. By default, the example project file is stored in your Enterprise Architect 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

Design a Custom Template

You can customize any Enterprise Architect project and use it as the base for the new project. This enables 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 default project is no different from 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.

Configure Project

Having created your project, you can set a range of project parameters to define defaults, tailor the project to particular coding languages, and make development and use of the project consistent. See the following topics:

- The Settings Menu
- Defaults and User Settings
6.1.3.1 Model Wizard

The Model Wizard is available on creating a new project. Once you have created a project, you can also access the Model Wizard from the Project Browser. Right-click on a root node and select the Add Model using Wizard menu option.

Select model(s) to add to your project

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select From Technology</td>
<td>Select the model category from which to select the model type to create.</td>
</tr>
<tr>
<td>(If you have advanced Add-Ins and MDG Technologies, this panel lists them). Select the appropriate technology to list the associated templates in the Name panel. To list the standard Enterprise Architect model templates, select &lt;default&gt;.</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Select all of the models.</td>
</tr>
<tr>
<td>None</td>
<td>Clear all models selected.</td>
</tr>
<tr>
<td>OK</td>
<td>Create the project tree for your project.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Create a blank project tree.</td>
</tr>
<tr>
<td>Help</td>
<td>Display this help topic.</td>
</tr>
</tbody>
</table>

If you are a Technology Developer, you can also create and distribute custom templates as part of your own MDG Technology. The name of your technology displays in the Technology panel and, when you select the technology, the model template names display in the Name panel. If you have defined a filter in your Model Technology File, you can select that from the Select from: drop-down list (see SDK for Enterprise Architect).
6.1.4 Set Up a Database Repository

Introduction
The Desktop and Professional versions of Enterprise Architect use an MS JET database as the model repository.

If you purchase the Corporate edition, you can create and use a SQL Server 2000 or 2005, MySQL, PostgreSQL, Adaptive Server Anywhere 8 or 9, Progress OpenEdge, MSDE or Oracle 9i, 10g or 11g data repository. You upsize the Enterprise Architect models (either existing or template) to use your selected DBMS.

The process of upsizing a model is straightforward and comprises 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 Enterprise Architect and use the Project Data Transfer function (select the Tools | Data Management | Project Transfer menu option) to move a model from a .EAP file to the DBMS repository.

This topic details how to do this for MySQL, SQL Server, PostgreSQL, Adaptive Server Anywhere, Progress OpenEdge, MSDE and Oracle 9i, 10g or 11g.

Note:
You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

Setting up a database repository is a two- or three-stage process: firstly, you set up an ODBC driver for your database; secondly, you create the repository tables using scripts downloaded from the Sparx Systems website; and finally, you connect to the repository. Full instructions on all three stages are provided below.

Set 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 stage can be skipped. To find out how to set up an ODBC driver, go to:
- Set Up a MySQL ODBC Driver
- Set Up a PostgreSQL ODBC Driver
- Set Up an Adaptive Server Anywhere ODBC Driver
- Set Up a Progress OpenEdge ODBC Driver

Create a Repository
To find out how to download the scripts and create the data repository tables, go to:
- Create a MySQL Data Repository
- Create a SQL Server Data Repository
- Create an Oracle Data Repository
- Create a PostgreSQL Data Repository
- Create an Adaptive Server Anywhere Data Repository
- Create an MSDE Server Data Repository
- Create a Progress OpenEdge Data Repository

Connect 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 Data Repository
- Connect to a PostgreSQL Data Repository
- Connect to an Adaptive Server Anywhere Data Repository
- Connect to an MSDE Server Data Repository
- Connect to a Progress OpenEdge Data Repository
6.1.4.1 Upsize to Sybase ASA

Before you set up Enterprise Architect for use with Sybase Adaptive Server Anywhere (ASA), it is recommended that you run the project integrity check tool (select the Tools | Data Management | Project Integrity Check menu option) on the base project to upsize to ASA. This ensures the data is ‘clean’ before uploading.

**Note:**

You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

**Warning:**

Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

You upsize your database for ASA in three stages, as follows:

#### Stage One: Install ASA Components

1. Install Adaptive Server Anywhere - SQL Anywhere Studio 8 or higher. This also installs the ASA ODBC driver.
2. Create a new database for the Enterprise Architect repository using Sybase Central.
3. Create a suitable ODBC Data Source to point to your new database.

**Note:**

See [Set up an Adaptive Server Anywhere ODBC Driver](http://www.sparxsystems.com/registered/).

#### Stage Two: Configure the Database

From Sybase Central:

1. Right-click on the newly created database.
3. Run the script to create all required data structures.

**Note:**

See [Create a New Adaptive Server Anywhere Repository](http://www.sparxsystems.com/registered/).

You now have an empty database, and can transfer an existing model into the server.

#### Stage Three: Transfer the Data

1. Open Enterprise Architect (click on the Cancel button on the Open Project screen to open with no project loaded).
2. Select the Tools | Data Management | Project Transfer menu option. The Project Transfer dialog displays.
3. In the **Transfer Type** panel, select **EAP to DBMS**.

4. In the **Source Project** field, type the name of the .EAP file to upsize to ASA.

5. At the right of the **Target Project** field, click on the {...} (Browse) button. The Datalink Properties dialog displays.

6. Select **Microsoft OLE DB Provider for ODBC Drivers** from the list, then click on the **Next** button.

7. In the **Use Data source name** field, click on the drop-down arrow and select the ODBC Data Source you configured to point to your new database.

   **Note:**
   See [Connect to an Adaptive Server Anywhere Data Repository](#) for more information.

8. Click on the **OK** button.

9. If required, select the **Logfile** checkbox and enter a path for the data transfer log file.

10. Click on the **Transfer** button to begin the data transfer process. When the process is complete, you have up sized your model to Adaptive Server Anywhere and can now open it from Enterprise Architect.

### 6.1.4.2 Upsize to Progress OpenEdge

Before you set up Enterprise Architect for use with OpenEdge, it is recommended that you run the project integrity check tool (select the **Tools | Data Management | Project Integrity Check** menu option) on the base project to upsize to OpenEdge. This ensures the data is clean before uploading.

**Note:**
You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

**Warning:**
Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

You upsize your database for OpenEdge in three stages, as follows:
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/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 data structures.

   Note:
   See Create a New OpenEdge Repository.

Stage Three: Transfer the Data
1. Open Enterprise Architect (click on the Cancel button on the Open Project screen to open with no project loaded).
2. Select the Tools | Data Management | Project Transfer menu option. The Project Transfer dialog displays:

   - In the Transfer Type panel, select EAP to DBMS.
   - In the Source Project field, type the name of the .EAP file to upsize to OpenEdge.
   - At the right of the Target Project field, click on the [...] (Browse) button. The Datalink Properties dialog displays.
   - Select Microsoft OLE DB Provider for ODBC Drivers from the list, then click on the Next button.
   - In the Use Data source name field, click on the drop-down arrow and select the ODBC Data Source you configured to point to your new database.
Click on the **OK** button.

9. If required, select the **Logfile** checkbox and enter a path and filename for the data transfer log file.

10. Click on the **Transfer** button to begin the data transfer process.

When the process is complete, you have upsized your model to OpenEdge and can now open it from Enterprise Architect.

### 6.1.4.3 Upsize to MSDE

Before you set up Enterprise Architect for use with SQL Server Desktop Engine (MSDE), it is recommended that you run the project integrity check tool (select the **Tools | Data Management | Project Integrity Check** menu option) on the base project to upsize to MSDE. This ensures the data is 'clean' before uploading.

**Note:**
You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

**Warning:**
Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

Follow the steps in the doc to upsize your model to MSDE.

### 6.1.4.4 Upsize to PostgreSQL

Before you set up Enterprise Architect for use with PostgreSQL, it is recommend that you run the project integrity check tool (select the **Tools | Data Management | Project Integrity Check** menu option) on the base project to upsize to PostgreSQL. This ensures your data is clean before uploading.

**Note:**
You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

**Warning:**
Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

You upsize your database for PostgreSQL in three stages, as follows:

#### 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 the doc for more information.

#### 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 Systems website.
2. Run the script to create all required data structures.
Note:
See Create a New PostgreSQL Repository.

You now have an empty database and can transfer an existing model into the server.

Stage Three: Transfer the Data

1. Open Enterprise Architect (click on the Cancel button on the Open Project screen to open with no project loaded).
2. Select the Tools | Data Management | Project Transfer menu option. The Project Transfer dialog displays.

3. In the Transfer Type panel, select .EAP to DBMS.
4. In the Source Project field, type or select .EAP file to upsize to PostgreSQL.
5. At the right of the Target Project field, click on the ... (Browse) button. The Datalink Properties dialog displays.
6. Select Microsoft OLE DB Provider for ODBC Drivers from the list, then click on the Next button.
7. In the Use data source name field, click on the drop-down arrow and 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. Click on the OK button.
9. If required, select the Logfile checkbox and type a path and filename for the data transfer log file.
10. Click on the Transfer button to begin the data transfer process.

When the process is complete, you have upsized your model to PostgreSQL and can now open it from Enterprise Architect.
6.1.4.5 Upsize to Oracle 9i, 10g or 11g

Before you set up Enterprise Architect for use with Oracle, it is recommended that you run the project integrity check tool (select the **Tools | Data Management | Project Integrity Check** menu option) on the base project to upsize to Oracle. This ensures your data is clean before uploading.

**Note:**

You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

**Warning:**

Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

You upsize your database for Oracle in three stages, as follows:

#### 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 Systems website.
2. Make sure the new database is selected as the current database.
3. Run the script to create all required data structures.
   
   **Note:**
   
   See [Create a New Oracle Repository](#).

You now have an empty database and can transfer an existing model into the server.

#### Stage Three: Transfer the Data

**Note:**

When transferring a project you must have permission to execute the `CREATE SEQUENCE` command.

1. Open Enterprise Architect (click on the **Cancel** button on the **Open Project** screen to open with no project loaded).
2. Select the **Tools | Data Management | Project Transfer** menu option. The **Project Transfer** dialog displays:
3. In the Transfer Type panel, select `.EAP to DBMS`.

4. In the Source Project field, type the name of the `.EAP` file to upsize to Oracle.

5. At the right of the Target Project field, click on the `[ ... ]` (Browse) button. The Datalink Properties dialog displays.

6. Select Oracle Provider for OLE DB from the list, then click on the Next button.

7. On the Connection page of the Data Link Properties dialog, enter the Oracle service name in the DataSource field, and the user name and password as required.

   **Note:**
   
   See Connect to an Oracle Data Repository for more information.

8. Click on the OK button.

9. If required, on the Project Transfer dialog, select the Logfile checkbox and type a path and file name for the data transfer log file.

10. Click on the Transfer button to begin the data transfer process. Once the process is complete, you have upsize your model to Oracle and can now open it from Enterprise Architect.

### 6.1.4.6 Upsize to SQL Server

Before you set up Enterprise Architect for use with SQL Server, it is recommended that you run the project integrity check tool (select the Tools | Data Management | Project Integrity Check menu option) on the base project to upsize to SQL Server. This ensure the project data is “clean” before uploading.

**Note:**

You cannot move a model from a source `.EAP` file of a version earlier than 3.5.0.

**Warning:**

Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

You upsize your database for SQL Server in three stages, as follows:
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 Systems Website.
2. Make sure the new database is the currently active database.
3. Run the script to create all required data structures.

Note:
See Create a New SQL Server Repository.

You now have an empty database, and can transfer an existing model into the server.

Stage Three: Transfer the Data

Note:
When transferring a project you must have permission to execute the SET.IDENTITY_INSERT [table] {ON | OFF} command.

1. Open Enterprise Architect (click on the Cancel button on the Open Project screen to open with no project loaded).
2. Select the Tools | Data Management | Project Transfer menu option. The Project Transfer dialog displays.

3. In the Transfer Type panel, select .EAP to DBMS.
4. In the Source Project field, type the name of the .EAP file to upsize to SQL Server.
5. At the right of the Target Project field, click on the [...] (Browse) button. The Datalink Properties dialog displays.
6. Select Microsoft OLE DB Provider for SQL Server from the list, then click on the Next button.
7. On the Data Source Details page of the Connection dialog, type in the server name, database name and security details as required.

   **Note:**
   See Connect to a SQL Server Data Repository for more information.

8. Click on the OK button.

9. If required, select the Logfile checkbox and type in a path and filename for the data transfer log file.

10. Click on the Transfer button to begin the data transfer process. When the process is complete, you have upsized your model to SQL Server and can now open it from Enterprise Architect.

### 6.1.4.7 Upsize to MySQL

Before you set up Enterprise Architect for use with MySQL, it is recommended that you run the project integrity check tool (the Tools | Data Management | Project Integrity Check menu option) on the base project to upsise to MySQL. This ensures data is ‘clean’ before uploading.

**Note:**
You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

**Warning:**
Before proceeding, ensure MDAC 2.6 or higher is installed on your system.

### Upsizing Your Database

You upsise your database for MySQL in four stages, as follows:

#### 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:**
   There are two critical non-default settings required; see Set up a MySQL ODBC Driver and ensure you select the checkboxes in step 7.

#### Stage Two: Select Table Type

1. If you are using InnoDB tables, set up the MySQL .ini file as required and run the MySQL - InnoDB BaseModel script.
2. If you are using MyISAM tables, set up the MySQL .ini file as required and run the MySQL - MyISAM BaseModel script.

   **Notes:**
   - See the discussion on MySQL limitations.
   - The scripts are available to registered users on the Corporate edition Resources page of the Sparx Systems website.

#### Stage Three: Create the Database

1. Create an empty database.

   **Note:**
   See Create a New MySQL Repository.

You now have an empty database, and can transfer an existing model into the server as described below.
Stage Four: Transfer the Data

1. Open Enterprise Architect (click on the Cancel button on the Open Project screen to open with no project loaded).
2. Select the Tools | Data Management | Project Transfer menu option. The Project Transfer dialog displays:

   ![Project Transfer Dialog]

3. In the Transfer Type panel, select .EAP to DBMS.
4. In the Source Project field, type the name of the .EAP file to upsize to MySQL.
5. At the right of the Target Project field, click on the [...] (Browse) button. The Datalink Properties dialog displays.
6. Select Microsoft OLE DB Provider for ODBC Drivers from the list, then click on the Next button.
7. In the Use Data source name field, click on the drop-down arrow and 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. Click on the OK button.
9. If required, select the Logfile checkbox and type a path and filename for the data transfer log file.
10. Click on the Transfer button to begin the data transfer process.

   When the process is complete, you have upsized your model to MySQL and can now open it from Enterprise Architect.

### 6.1.4.8 Set Up an ODBC Driver

This topic details how to set up the following ODBC drivers:

- **MySQL ODBC Driver**
- **PostgreSQL ODBC Driver**
- **Adaptive Server Anywhere ODBC Driver**
- **Progress OpenEdge ODBC Driver**
6.1.4.8.1 MySQL ODBC Driver

Before you can connect to a MySQL data repository, you must first set up a MySQL ODBC driver for which, in turn, you must first have installed Microsoft MDAC components, a MySQL DBMS system and a MySQL ODBC driver.

Note:
The MySQL ODBC driver version 3.51.14 creates problems in incorporating tests in elements. Use a different version, such as 3.51.12.

To set up your MySQL ODBC Driver, follow the steps below:

1. Select the Windows™ Control Panel | Administrative Tools | Data Sources (ODBC) option. The ODBC Data Sources Administrator window displays.

2. Click on the Add button. The Create New Data Source dialog displays, enabling you to add a new DSN.
3. Select **MySQL ODBC 3.51 Driver** from the list.

4. Click on the **Finish** button. The **Connector/ODBC 3.51.12 - Add Data Source Name** dialog displays.

5. Enter the following configuration details:
   - A data source name for the connection
   - A description (optional)
   - The host address of the DBMS server
   - User name and password
   - The database name on the selected server.

   See the example below:

6. Click on the **Advanced** tab and **Flags 1** tab to set the advanced options.
7. Select the **Don't Optimize Column Width** and **Return Matching Rows** checkboxes, then click on the **OK** button.
8. Click on the **Test** button to confirm that the details are correct.
9. If the test succeeds, click on the **OK** button 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.

### 6.1.4.8.2 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 must have Microsoft MDAC components, PostgreSQL DBMS system and PostgreSQL ODBC driver (version 7.03.01.00 minimum) installed.

To set up your PostgreSQL ODBC driver, follow the steps below:

1. Select the Windows™ **Control Panel** | **Administrative Tools** | **Data Sources (ODBC)** option. The **ODBC Data Sources Administrator** window displays.
2. Click on the **Add** button. The **Create New Data Source** dialog displays, enabling you to add a new DSN.

3. Select **PostgreSQL** from the list.
4. Click on 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.
6. Click on the **Datasource** button and set the options on Page 1 and Page 2 as shown on the examples below:
Note:
On Page 2, for PostgreSQL version 8+ select the Disallow Premature checkbox and, in the Protocol panel, select the 7.4+ radio button.

7. Click on the OK button to complete the configuration.
Your PostgreSQL connection is now available to use in Enterprise Architect.

6.1.4.8.3 ASA ODBC Driver

Before you can connect to an Adaptive Server Anywhere (ASA) data repository, you must first set up an ASA ODBC driver. To do this, you must have Microsoft MDAC components, the ASA DBMS system and the ASA ODBC driver (installed with the ASA DBMS) installed.

To set up your ASA ODBC Driver, follow the steps below:

1. Select the Windows™ Control Panel | Administrative Tools | Data Sources (ODBC) option. The ODBC Data Sources Administrator window displays.
2. Click on the **Add** button. The **Create New Data Source** dialog displays, enabling you to add a new DSN.

3. Select **Adaptive Server Anywhere 8.0** from the list.
4. Click on the **Finish** button.
5. Enter the following configuration details:
   - A name for the connection on the **ODBC tab**.
- The username and password on the Login tab (dba, sql are the defaults when ASA is installed).

- The server name and the path to the database, on the Database tab.
6. You can now return to the ODBC tab and test the connection.
7. Click on the **OK** button to complete the configuration.

Your Adaptive Server Anywhere connection is now available to use in Enterprise Architect.
6.1.4.8.4 Progress OpenEdge ODBC Driver

Before you can connect to an OpenEdge data repository, you must first set up an OpenEdge ODBC driver. To do this, you must have Microsoft MDAC components, OpenEdge DBMS system and DataDirect ODBC driver for OpenEdge (version 4.20 minimum) installed.

To set up the ODBC Driver, follow the steps below:

1. Select the Windows™ Control Panel | Administrative Tools | Data Sources (ODBC) option. The ODBC Data Sources Administrator window displays.

2. Click on the Add button. The Create New Data Source dialog displays, enabling you to add a new DSN.

3. Select the DataDirect/OpenEdge SQL driver from the list.

4. Click on the Finish button. The DSN Configuration dialog displays.

5. Enter the following configuration details:
- The **Data Source Name**
- The **Description** (optional)
- The **Host Name** and **Port Number** of the DBMS server
- The **Database Name** on the selected server
- The **User ID**.

See the example below:

6. Click on the **Test Connect** button to confirm that the details are correct.
7. If the test succeeds, click on the **OK** button to complete the configuration.
8. If the test does not succeed, review your settings.

Your OpenEdge connection is now available to use in Enterprise Architect.

### 6.1.4.9 Create a Repository

This topic details how to create the following data repositories:

- MySQL
- SQL Server
- Oracle 9i, 10g or 11g
- PostgreSQL
- Adaptive Server Anywhere (ASA)
- MSDE Server

#### 6.1.4.9.1 MySQL Repository

**Note:**

This feature is available in the Corporate edition only.

Before creating a MySQL data repository in Enterprise Architect, you must set up the MySQL and MySQL ODBC drivers. For further information on setting these up, see Setup a MySQL ODBC Driver.

To create a new MySQL repository, you must first create a database into which to import the table definitions for Enterprise Architect. Sparx Systems provide SQL scripts to create the required tables; how you create the database and execute that script are up to you.

Create the Data Repository

Once you have created the database and executed the script, you should have an empty Enterprise Architect 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 might want to consider a suitable frontend 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 and restores.

To get started with Navicat, follow the steps below:

1. Create a new database.

![Create New Database](image)

2. Open and execute the MySQL script.

![MySQL Script](image)
3. Below is an example showing the tables created in the MySQL repository after running the script in Navicat.

6.1.4.9.2 SQL Server Repository

**Note:**

This feature is available in the Corporate edition only.

Before creating a SQL Server data repository, you must have SQL Server and MDAC 2.6 or higher installed, and access permission to create a new database. Please note that setting up SQL Server and the issues involved are beyond the scope of this user guide. 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/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/resources/corporate/.

Create a SQL Server Repository

SQL Server repositories are created without any data, so you must perform a project data transfer in Enterprise Architect to copy a suitable starter project. If you are starting from scratch, EABase.EAP is a good starting point. If you are using an existing .EAP model, you can upsize it.

To use SQL Enterprise Manager to create a SQL Server repository, follow the steps below:

1. In SQL Enterprise Manager, locate the server on which to create your new Enterprise Architect model; in the example below this is (LOCAL).
2. Right-click and choose New Database from the context menu.
3. Enter a suitable name for the database. Set any file options as required.

**Note:**
Ensure that the database collation is case-insensitive.

4. Click on the database to select it, then select the Tools | SQL Query Analyzer menu option.

5. In Query Analyzer, use the File Find dialog to locate the supplied Enterprise Architect SQL Server Model script file. Click on the Open button.
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. Click on the Run button; SQL Server executes the script, which creates the base model for an Enterprise Architect project.

Tip:
Use the Project Data Transfer function to upload a basic model into the repository.

6.1.4.9.3 Oracle Server Repository

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

Before creating an Oracle data repository, you must have the appropriate version of Oracle (9i, 10g or 11g) and MDAC 2.6 or higher installed, and access access 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 documentation for guidance.

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/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/resources/corporate/.

Create the Data Repository

Oracle repositories are created without any data, so you must perform a project data transfer in Enterprise Architect to copy a suitable starter project. If you are starting from scratch, EABase.EAP is a good starting point. If you want to use an existing .EAP model, you can upsize it; follow the steps below:

1. Create a new database on the Oracle 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 creates the base model tables and indexes for an Enterprise Architect Project.
Note:
Use the Project Data Transfer function to upload a basic model into the repository.

6.1.4.9.4 PostgreSQL Repository

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

Before creating a PostgreSQL data repository in Enterprise Architect, you must set up PostgreSQL and PostgreSQL ODBC drivers. For further information on setting these up, see Set up a PostgreSQL ODBC Driver.

To create a new PostgreSQL repository, you must first create a database into which to import the table definitions for Enterprise Architect. 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/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/resources/corporate/

Create the Data Repository

After you create the database and execute the script, the result should be an empty Enterprise Architect 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 might want 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 and restores.

To get started with EMS PostgreSQL Manager, follow the steps below:

1. Create a new database.

2. Open and execute the PostgreSQL sql script.
3. Below is an example showing the tables created in the PostgreSQL repository after running the script in EMS PostgreSQL Manager.

```
CREATE SEQUENCE id_seq;
CREATE SEQUENCE property_id_seq;
CREATE SEQUENCE category_id_seq;
CREATE SEQUENCE connector_id_seq;
CREATE SEQUENCE data_type_id_seq;
CREATE SEQUENCE diagram_id_seq;
CREATE SEQUENCE instance_id_seq;
CREATE SEQUENCE glossary_id_seq;
CREATE SEQUENCE image_id_seq;
CREATE SEQUENCE issue_id_seq;
CREATE SEQUENCE object_id_seq;
CREATE SEQUENCE req_id_seq;
CREATE SEQUENCE operation_id_seq;
```

6.1.4.9.5 Adaptive Server Anywhere Repository

**Note:**

This feature is available in the Corporate edition only.

Before creating an ASA data repository in Enterprise Architect, you must set up ASA and ASA ODBC drivers. For further information on setting these up, see Setup an Adaptive Server Anywhere ODBC Driver.

To create a new ASA repository, you must first create a database into which to import the table definitions for Enterprise Architect. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you.

- Trial users can obtain the scripts from the Corporate edition Resources page of the Sparx Systems website at http://www.sparxsystems.com/resources/corporate/.

Create the Data Repository

After you create the database and execute the script, the result should be an empty Enterprise Architect 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 might want 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 and restores.

To get started with Sybase Central, follow the steps below:

1. Create a new database.
2. Open and execute the ASA SQL script.
Below is an example showing the tables created in the ASA repository after running the script in EMS ASA Manager.
6.1.4.9.6 **MSDE Server Repository**

**Note:**

This feature is available in the Corporate edition only.

Before creating a SQL Server MSDE data repository, you must have MSDE Server and MDAC 2.6 or higher installed. Please note that setting up MSDE Server and the issues involved are beyond the scope of this user guide. Consult your program documentation for guidance.

Sparx Systems provide SQL scripts to create the required tables; how you create the database and execute that script are up to you.


Use the SQL Server 2000 and 2005 script for MSDE, and follow the steps to [Create a New SQL Server Data Repository](#).

6.1.4.9.7 **Progress OpenEdge Repository**

**Notes:**

- This feature is available in the Corporate edition only.
- The OpenEdge database must be either version 10.0B03 or version 10.1B01, or later.

Before creating a Progress OpenEdge data repository, you must have OpenEdge and MDAC 2.6 or higher installed, and access 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 guidance.
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 Registered Corporate edition Resources page of the Sparx Systems website at http://www.sparxsystems.com/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/resources/corporate/

**Create the Data Repository**

OpenEdge repositories are created without any data, so you must perform a project data transfer in Enterprise Architect to copy a suitable starter project. If you are starting from scratch, EABase.EAP is a good starting point. If you want to use an existing .EAP model, you can upsize 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
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 Project Data Transfer function to upload a basic model into the repository.

### 6.1.4.10 Connect to a Data Repository

This topic describes how to connect to the following data repositories:

- MySQL Data Repository
- SQL Server Data Repository
- Oracle Data Repository
- PostgreSQL Data Repository
- Adaptive Server Anywhere Data Repository
- MSDE Server Data Repository
- Progress OpenEdge

#### 6.1.4.10.1 MySQL Data Repository

**Note:**

This feature is available in the Corporate edition only.

In order to use a MySQL data repository, you must connect to it in Enterprise Architect first. Before connecting to the repository, you must set up a MySQL ODBC driver. Be aware, there are some limitations with using MySQL with Enterprise Architect.

To connect to a MySQL data repository in Enterprise Architect, follow the steps below:

1. In the Open Project dialog, select the Connect to Server checkbox.
2. Click on the [ ... ] (Browse) button, as you normally would to browse for a project. As you have selected the Connect to Server checkbox, the Data Link Properties dialog displays instead of the Browse Directories dialog.
3. Select **Microsoft OLE DB Provider for ODBC Drivers** from the list.
4. Click on the **Next** button. The **Connection** tab displays.
5. Click on the **Use data source name** radio button and on the drop-down arrow in its field. Select the ODBC driver you have set up to connect to your MySQL repository from the list. In the setup example the driver title is **MySQL-EABASE**.

6. If required, type in a **User name** and **Password**.

7. If required, type in an initial catalog.

8. Click on the **Test Connection** button to confirm that the details are correct.

9. If the test succeeds, click on the **OK** button.

10. If the test does not succeed, revise your settings.

11. After you have clicked on the **OK** button, the **Connection Name** dialog displays.

12. Give the connection a suitable name so that you can recognize it in the **Recent Projects** panel on the **Open Project** dialog.

13. If required, select the **Encrypt Connection String** checkbox. This encrypts and hides the connection details of the database from the users that the connection string is given to.
14. Click on the **OK** button to complete the configuration.

### 6.1.10.2 SQL Server Data Repository

**Note:**

This feature is available in the Corporate edition only.

Before you can use a SQL Server data repository, you must connect to it in Enterprise Architect. To connect to your SQL Server data repository in Enterprise Architect, follow the steps below:

1. In the **Open Project dialog**, select the **Connect to Server** checkbox.
2. Click on the [ ... ] (Browse) button, as you normally would to browse for a project. As you have selected the **Connect to Server** checkbox, the Data Link Properties dialog displays instead of the Select Enterprise Architect Project to Open dialog.

   ![Data Link Properties](image)

3. Select **Microsoft OLE DB Provider for SQL Server** from the list.
4. Click on the **Next >>** button. The **Connection** tab displays.
5. Type in the server details, including **Server Name**, **User Name** and **Password**.

6. Click on the **Select the database on the server** option and on the drop-down arrow. From the list, select the model to connect to.

7. Click on the **Test Connection** button to confirm that the details are correct.

8. If the test succeeds, click on the **OK** button. If the test does not succeed, revise your settings.

9. When you click on the **OK** button, the **Connection Name** dialog displays.

10. In the **Name** field, type a suitable name for the connection so that you can recognize it in the Recent Projects panel on the Open Project dialog.

11. If required, select the **Encrypt Connection String** checkbox. This encrypts and hides the connection details of the database from the users that the connection string is given to.

12. Click on the **OK** button to complete the configuration.
6.1.4.10.3 Oracle Data Repository

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

In order to use an Oracle data repository, you must connect to it in Enterprise Architect first. To connect to your Oracle 9i, 10g or 11g data repository in Enterprise Architect, follow the steps below:

1. In the Open Project dialog, select the Connect to Server checkbox.
2. Click on the [...] (Browse) button, as you normally would to browse for a project. As you have selected the Connect to Server checkbox, the Data Link Properties dialog displays instead of the Browse Directories dialog.
3. Select Oracle Provider for OLE DB from the list.
   Note: Do not select Microsoft OLE DB Provider for Oracle; Enterprise Architect might not work as expected.
4. Click on the Next button. The Connection tab displays.
5. Enter the **Data Source** name (the service name of the Oracle database), the database **User Name** and the **Password**. The **Location** field is not required.

6. Click on the **Test Connection** button to confirm that the details are correct.

7. If your test succeeded, click on the **OK** button.

8. If your test did not succeed, revise your settings.

9. After you have clicked on the **OK** button, the **Connection Name** dialog displays.

10. Give the connection a suitable name so that you can recognize it in the **Recent Projects** panel on the **Open Project** dialog.

11. If required, select the **Encrypt Connection String** checkbox. This encrypts and hides the connection details of the database from the users that the connection string is given to.

12. Click on the **OK** button to complete the configuration.
6.1.4.10.4 PostgreSQL Data Repository

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

In order to use a PostgreSQL data repository, you must connect to it in Enterprise Architect first. Before connecting to the repository, you must have set up a PostgreSQL ODBC driver. To connect to a PostgreSQL data repository in Enterprise Architect, follow the steps below:

1. In the Open Project dialog, select the Connect to Server checkbox, or on the Start Page, click on the Connect to Server Repository link.
2. Click on the [ ... ] (Browse) button, as you normally would to browse for a project. As you have selected the Connect to Server checkbox, the Data Link Properties dialog displays instead of the Browse Directories dialog.

3. Select Microsoft OLE DB Provider for ODBC Drivers from the list.
4. Click on the Next button. The Connection tab displays.
5. Click on the **Use data source name** drop-down arrow and, from the list, select the ODBC driver you have set up to connect to your PostgreSQL repository.

6. Click on the **Test Connection** button to confirm that the details are correct.

7. If your test succeeded, click on the **OK** button.

8. If your test did not succeed, revise your settings.

9. After you have clicked on the **OK** button, the **Connection Name** dialog displays.

10. Give the connection a suitable name so that you can recognize it in the **Recent Projects** panel on the **Open Project** dialog.

11. If required, select the **Encrypt Connection String** checkbox. This encrypts and hides the connection details of the database from the users that the connection string is given to.

12. Click on the **OK** button to complete the configuration.
6.1.4.10.5 ASA Data Repository

**Note:**
This feature is available in the Corporate edition only.

In order to use an ASA data repository, you must connect to it in Enterprise Architect first. Before connecting to the repository, you must have set up an ASA ODBC driver. To connect to an ASA data repository in Enterprise Architect, follow the steps below:

1. In the Open Project dialog, select the Connect to Server checkbox or, on the Start Page, click on the Connect to Server Repository... link.
2. Click on the [...] (Browse) button, as you normally would to browse for a project. As you have selected the Connect to Server checkbox, the Data Link Properties dialog displays instead of the browse directories dialog.

![Data Link Properties dialog](image)

3. Select Microsoft OLE DB Provider for ODBC Drivers from the list.
4. Click on the Next button. The Connection tab displays.
5. In the **Use data source name** field, click on the drop-down arrow and select the ODBC driver you set up to connect to your ASA repository.

6. Click on the **Test Connection** button to confirm that the details are correct.

7. If your test succeeded, click on the **OK** button.

8. If your test did not succeed, revise your settings.

9. After you have clicked on the **OK** button, the **Connection Name** dialog displays.

10. Give the connection a suitable name so you can recognize it in the **Recent Projects** panel on the **Open Project dialog**.

11. If required, select the **Encrypt Connection String** checkbox. This encrypts and hides the connection details of the database from the users that the connection string is given to.

12. Click on the **OK** button to complete the configuration.
6.1.4.10.6 MSDE Server Data Repository

Follow the steps in Connect to a SQL Server Repository.

6.1.4.10.7 Progress OpenEdge Repository

**Note:**
This feature is available in the Corporate edition only.

In order to use an OpenEdge data repository, you must connect to it in Enterprise Architect first; follow the steps below:

1. In the Open Project dialog, select the Connect to Server checkbox.
2. Click on the ... (Browse) button, as you normally would to browse for a project. As you have selected Connect to Server, the Data Link Properties dialog displays instead of the Browse Directories dialog.
3. Select Microsoft OLE DB Provider for ODBC Drivers from the list.
4. Click on the Next button. The Connection tab displays.
5. In the **Use data source name** field, click on the drop-down arrow and select the ODBC driver you have set up to connect to your OpenEdge repository. In the setup example, the driver title is `openedge_users`.

6. Enter the **User name** and **Password**.

7. Enter the initial catalog.

8. Click on the **All** tab.

9. In the **Property Value** field, edit the Extended Properties value to: `DefaultSchema=PUB`.

10. Click on the **Test Connection** button to confirm that the details are correct.

11. If the test succeeds, click on the **OK** button. If the test does not succeed, revise your settings.

12. After you have clicked on the **OK** button, the Logon to Progress dialog displays.
13. Check the details, and click on the **OK** button. The **Connection Name** dialog displays.

14. Give the connection a suitable name so you can recognize it in the **Recent Projects** panel on the **Open Project dialog**.

15. If required, select the **Encrypt Connection String** checkbox. This encrypts and hides the connection details of the database from the users that the connection string is given to.

16. Click on the **OK** button to complete the configuration.
6.1.5 MySQL Limitations and SQL Scripts

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 must 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/registered/reg_ea_corp_ed.html.

2. Due to limitations of the SQL query engine, some advanced search capabilities in the Model 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 Enterprise Architect.

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/registered/reg_ea_corp_ed.html.
6.1.6 Copy a Base Project

To copy an existing Base Project, from the Start Page select the Copy a Base Project... option. The Create New Enterprise Architect Project dialog displays.

To create a new Enterprise Architect project, you must select a project template to form the base model for the new project. When you install Enterprise Architect a default model is installed called EABase.eap.

- To select the file path for saving your project, click on the Browse button after the New Project field. If this is to be a shared project, store the file on a shared network resource such as a Network Server or Workgroup Server.
- To replace all GUIDs in the source model with fresh GUIDs, select the Reset New Projects GUIDs checkbox.

Note:
If the new project is based on one that is already under version control, it is recommended that the Reset New Projects GUIDs checkbox be deselected. This prevents duplication of packages when the Get Latest facility is used.

- To select the base model for your project, click on the Browse button after the Model Project field. The field defaults to EABase.eap; however you can select any existing model file (see the Design a Custom Template topic).

When you have entered the filenames, click on the Create Project button to create your project. Click on the Cancel button to close 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.
6.2 Upgrade 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 alerts you to the upgrade required and takes you through the upgrade process. This brings your project to the current level to support all the latest Enterprise Architect features.
6.2.1 The Upgrade Wizard

When you try to load a project that has to be updated to the latest format, the Upgrade Wizard opens and guides you through the required steps. You cannot load a previous Enterprise Architect version model without upgrading.

**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.

**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 Enterprise Architect.

The Wizard:
- Advises you of the necessity to upgrade
- Advises you to back up the current project; backing up before any changes are made is essential
- Checks that the current project can be upgraded; all projects can be upgraded, except replicas that are not Design Masters
- Performs the upgrade
- Opens the newly converted project.
6.2.2 Upgrade Replicas

Models that have replication features added might have to be handled differently 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 Enterprise Architect must remove the replication features, then upgrade the project in the normal manner. Use the Upgrade Wizard to guide you through the steps.

The Upgrade Wizard should handle the upgrade process for you, and detect which upgrade method is the best.
If you have a failed XMI import, network crash or other unforeseen event that could disrupt the integrity of information in the model, it is recommended to run the Project Integrity Check function. This examines all database records and ensures there are no ‘orphaned’ records or inaccurate or unset identifiers. You can run the integrity checker first in report mode to discover if anything should be corrected, and then run it again in repair mode.

When Enterprise Architect checks the model, it attempts to recover lost packages and elements, and generates a new package at the model root level called _recovered_. Check through any elements that are found and, if required, 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 select either to just report on the state of your model, or to try and repair any inconsistencies. The recovery process tries to restore elements where possible, but in some cases simply deletes the lost records.

**See Also**

- Run SQL Patches
6.3.1 Check Project Data Integrity

**Note:**
In the Corporate edition of Enterprise Architect, if security is enabled you must have **Check Project Data Integrity** permission to perform a data integrity check.

To check the data integrity of your project, follow the steps below:

1. Select the **Tools | Data Management | Project Integrity Check** menu option. The Project Integrity Check dialog displays.

2. Select the checks to run; the basic checks available are:
   - Package Structure
   - Object Structure
   - Object Features
   - GUIDs
   - Cross References
   - Connectors
   - UML 2.0 Migration

3. Select either:
   - the **Report Only** option to just view a report on the state of your model, or
   - the **Recover/Clean** option to attempt to recover and clean your project.

**Warning:**
If you intend to select the **Recover/Clean** option, you should back up your project file first.

Click on the **Go** button to run the check.
UML 2.0 Migration

The UML 2.0 Migration check enables you to migrate the project from UML 1.3 semantics to UML 2.0 semantics. 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 that does not enable any of the other checks to be selected. To perform the UML 2.0 migration follow the steps below:

1. Select the Tools | Data Management | Project Integrity Check menu option. The Project Integrity Check dialog displays.
2. Select the UML 2.0 Migration checkbox and click on the Go button. The following message box displays:

   ![Warning]
   
   This check migrates superceded modeling semantics to their UML 2.x counterparts and is run exclusively to the other checks in this process. Refer to the Help for additional information.

   [OK] [Cancel]

3. To proceed, click on the OK button, or to cancel the migration click on the Cancel button.
4. If you are proceeding, click on the Go button to perform the migration.
6.3.2 Run SQL Patches

Occasionally, Sparx Systems might release a patch to correct a model fault. To load such patches and run them, select the **Tools | Run Patch** menu option. The patch generally checks how many records are to be updated, and reports on what is to be done.
6.4 Project Data Transfer

The Corporate edition of Enterprise Architect supports SQL Server, MySQL, and Oracle 9i, 10g and 11g data repositories. At some point, it might become necessary to move a complete model from one repository to another, row by row, table by table.

The project data transfer function enables 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 a .EAP file
- Move a repository from SQL Server to MySQL or from one server to another
- Move all records from a .EAP file with replication to a model with none (Remove Replication)
- Copy all records from a .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 Project Data Transfer topic for instructions.

Note:
You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

Warning:
All records in the target repository are overwritten.
6.4.1 Perform a Project Data Transfer

**Warning:**
During a project data transfer, all records in the target project are overwritten. Before performing the transfer, take a backup of the target project to ensure that you can recover any important information it contains.

**Notes:**
- In the Corporate edition of Enterprise Architect, if security is enabled you must have **Transfer Data** permission to transfer project data between repositories.
- You cannot move a model from a source .EAP file of a version earlier than 3.5.0.

To perform a project data transfer, follow the steps below:

1. Select the **Tools** | **Data Management** | **Project Transfer** menu option. The **Project Transfer** dialog displays.

   ![Project Transfer Dialog](image)

   **Transfer Type**
   - EAP to EAP
   - DBMS to EAP
   - .EAP to DBMS
   - DBMS to DBMS

   **Source and Target Projects**
   - Source Project:
   - Target Project:

   **Logfile**
   - Logfile

   Caution: The Target Project will be erased prior to transfer. Please ensure you have backed up the target if necessary.

   ![Progress](image)

2. Click on the option for the required transfer type. You can choose from:
   - EAP to EAP
   - DBMS to EAP
   - .EAP to DBMS
   - DBMS to DBMS

3. In the **Source Project** and **Target Project** fields, type or select the name or connection string for the Source and Target projects.

4. Click on the **Transfer** button.

   It is good practice to do a **Project Compare** after this process to verify that all records are written.
6.4.2 Why Compare Projects?

It is sometimes useful to compare the size and row counts of two projects; for example, after a database crash, after import from XMI or after performing a deletion of model elements.

You can compare .EAP files to other .EAP files or to DBMS based repositories, or compare two DBMS repositories. Enterprise Architect examines the number of rows in each database and produces 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 projects this way is a convenient 'sanity check' after restoring a backup or doing a project data transfer. If discrepancies are found, you must investigate further manually.

See the Compare Projects topic for instructions.
6.4.3 Compare Projects

To compare projects, follow the steps below:

1. Select the Tools | Data Management | Project Compare menu option. The Project Compare dialog displays:

   ![Compare Type](image)

   - **Source Project**
   - **Target Project**

   - **Compare Type** options:
     - EAP to EAP
     - DBMS to EAP
     - EAP to DBMS
     - DBMS to DBMS

2. Select the option for the required comparison type. You can choose from:
   - EAP to EAP
   - DBMS to EAP
   - EAP to DBMS
   - DBMS to DBMS

3. In the Source Project and Target Project fields, type the name or connection string for the Source and Target projects.

4. Click on the Compare Projects button. The results of the comparison display in the panel at the bottom of the dialog.

5. If required, click on the Print List button to print the results.
6.4.4 Copy Packages Between Projects

Using the XML import/export capabilities of Enterprise Architect, you can copy and move packages between Enterprise Architect models. This gives you a high level of flexibility in building a model from re-usable parts and from elements produced in widely-dispersed geographic regions.

Procedure

To copy a package from one Enterprise Architect model to another, follow the steps below:

1. Open the Enterprise Architect model to copy from.
2. In the Project Browser, right-click on the package to copy. The context menu displays.
3. Select the **Import/Export | Export package to XMI file** menu option. The *Export Package to XMI* dialog displays.
4. Select the appropriate options and filename (see the *Export to XMI* topic for further information).
5. Click on the *Export* button to begin the export process.
6. When the export is complete, open the recipient Enterprise Architect model. In the Project Browser, navigate to the location to import the package into.
7. Right-click to display the context menu, and select the **Import/Export | Import package from XMI file** menu option. The *Import Package from XMI* dialog displays.
8. Select the appropriate options and filename (see the Import Package from XMI topic for further information).

9. Click on the **Import** button. The package is copied from the source project to the destination project.

**Note:**

If the package you are importing already exists in the target model (that is, it has been imported previously), you must either import over the existing package or select the **Strip GUIDs** option, in which case Enterprise Architect creates a replica of the original package. You can also use this technique to copy an entire package within the same model.
6.5 Model Maintenance

This topic highlights some administrative functions you might have to carry out to maintain your model. Note that this maintenance applied to models created as .EAP files. The processes are not required for models stored in a DBMS:

- Rename a Project
- Compact a Project
- Repair a Project
6.5.1 Rename a Project

**Important:**
The only way to rename an Enterprise Architect project is at the Windows file system level.

To rename an Enterprise Architect project .EAP file, 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.
6.5.2 Compact a Project

After some time, a project .EAP file might benefit from compacting to conserve space.

**Notes:**
- Compacting shuffles the contents of the model around, eliminating unused space and generally reducing the size of your model file.
- In the Corporate edition of Enterprise Architect, if security is enabled you must have **Administer Database** permission to compact a project.

To compact a project, follow the steps below:
1. Ensure that no users have the target project open.
2. Select the **Tools | Manage .EAP File | Compact .EAP File** menu option.
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.
6.5.3 Repair a Project

If a project has not been closed properly, such as during system or network outages, on rare occasions the .EAP file does not open correctly. In this case a message displays informing you the project is of an unknown database format or is not a database file.

**Warning:**

Never attempt to repair a project over a network connection; copy it to a local drive first.

**Notes:**

- Poor network connections can also cause this symptom.
- In the Corporate edition of Enterprise Architect, if security is enabled you must have **Administer Database** permission to repair a project.

---

**Repair a Project That Has Not Closed Correctly**

To repair a project that was not closed properly, follow the steps below:

**Note:**

All users must be logged off the project you are attempting to repair.

1. Copy the project file to a local drive on your PC.
2. In Enterprise Architect, select the **Tools | Options** menu option and on the **General** page deselect the **Use Jet 4.0 - requires restart** checkbox.
3. Close and restart Enterprise Architect and open a place holder project to enable access to the **Repair .EAP File** facility.

**Note:**

This is NOT the project you intend to repair, it is a copy of it.

4. Select the **Tools | Manage .EAP File | Repair .EAP File** menu option.
5. Follow the on-screen instructions.

**Ensure Integrity of the Repaired Project**

An additional step you can use to ensure the integrity of your project is to use the **Remove Replication** feature.

1. Open Enterprise Architect, but when you are prompted to open a project, click on the **Cancel** button.
2. Select the **Tools | Manage .EAP File | Remove Replication** menu option.
3. Follow the prompts. When you are prompted for the **Replica Project Browser** window for your problem project, you might be given a warning about the project not being the **Design Master**; accept this warning. Click on the **Next** button.
4. Browse for the clean project (e.g. **EABase.eap**). Click on the **Next** button.
5. Enter the path and name of the new project to be created, then click on the **Next** button.
6. Click on the **Run** button to run the removal process.
7. 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.
6.5.4 Manage Views

The top level packages in a model (below the model root node) are created as Views. Views are used simply to subdivide the model into partitions such as Business Process, Logical View or Dynamic View. They are a good way to extend the model depending on specific requirements and modeling techniques.

There are 6 main types of View, each with their own package icon:

- **Use Case** - e.g. Use Case diagram, Analysis diagram
- **Dynamic View** - e.g. Activity diagram, Communication diagram, Sequence diagram, State diagram
- **Class View** - e.g. Class Model, Code Engineering, Data Model
- **Component View** - e.g. Component diagram
- **Deployment View** - e.g. Deployment diagram
- **Simple View**

You can use the first five categories or devise your own based on the Simple View. You can create Views, rename them, move them into a different order, or delete them. Do this by right-clicking the mouse on the selected View to open the context menu, and choose the appropriate option.

6.5.4.1 Add Views

The example below shows an additional view called Java Code View, which has been appended to the end of the Views list.

Create a View

To create a View, follow the steps below:

1. Right-click on the model root node in the Project Browser. The context menu displays.
2. Select the New View menu option. The Create New View dialog displays.
3. In the **Name** field, type the name of the View.
4. In the **Set View Icon Style** panel, click on the radio button for the required View icon.
5. If the model root node is under version control, the **Add to Version Control** checkbox displays, defaulted to selected. If you do not want the new View to also be under version control, deselect the checkbox.
6. Click on the **OK** button.

### 6.5.4.2 Rename Views

If required, you can rename a view.

**Procedure**

To rename a view, follow the steps below:

1. Right-click on the View in the **Project Browser**. The context menu displays.
2. Select the **Properties** menu option. The **Package Properties** dialog displays.
3. In the **Name** field, type the new name and click on the **OK** button.

### 6.5.4.3 Delete Views

If necessary, you can delete a view.

**Warning:**

If you delete a view, all its contents are deleted at the same time. It **CANNOT** be restored.

**Procedure**

To delete a view, follow the steps below:

1. In the **Project Browser**, right-click on the view to delete. The context menu displays.
2. Select the **Delete <viewname>** option. The following warning displays:

   ![Warning dialog]

   **Warning:** Any child elements will be removed as well

   ![Yes/No buttons]

3. To delete the view and its contents, click on the **Yes** button. To cancel the deletion, click on the **No** button.
6.6 Model Validation

You use Model Validation to check UML models against known UML rules as well as any constraints defined within the model, using the Object Constraint Language (OCL). You can run Model Validation against a single UML element, a diagram or an entire package.

Validating a UML:

- **Element** validates the element and its children, features (attributes and operations) and relationships (connectors)
- **Diagram** validates the diagram itself (for correctness) as well as any elements and connectors within the diagram
- **Package** validates the package and all subpackages, elements, connectors and diagrams within it.

To use Model Validation, follow the steps below:

1. Select the package, diagram or element either from the Project Browser or within an open diagram.
2. Select the **Project | Model Validation | Validate Selected** menu option, or press [Ctrl]+[Alt]+[V].

Enterprise Architect performs the validation, and displays the results in the Output window. To display the Output window, select the **View | Output** menu option.

While performing the validation, Enterprise Architect also displays a progress window containing the **Cancel Validation** button, which enables you to cancel the validation process at any time.

Alternatively, select the **Project | Model Validation | Cancel Validation** menu option.

### Example Model Violations.

The following UML diagram contains several basic violations.
If you run Model Validation on this diagram, Enterprise Architect displays the following violations in the Output window:

- 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 Object1)
- Contains a UML ExceptionHandler (ExceptionHandler1) that is missing its child input ObjectNode.

**Note:**
If you double-click on an error in the Output window, you select the diagram element that the error message refers to.
6.6.1 Configure Model Validation

Use the Model Validation Configuration dialog to enable and disable the rules that are run with the model validator. You can define additional rules in this dialog from any additional Add-Ins that might be installed besides Enterprise Architect.

To display the Model Validation Configuration dialog, select the Project | Model Validation | Configure menu option.

Click on the checkbox against each Validation Rule to apply in performing a model validation.

Tip:
To disable UML syntax ("The requested connection is not UML compliant"), select the Tools | Options menu option, click on Diagram in the hierarchy, and in the General panel deselect the Strict UML Syntax checkbox.

When you perform a validation, each violation listed on the Output window has a violation ID of the format MVRxxnnnn, where:
- MVR stands for Model Validation Rule
- xx is a hexadecimal number corresponding to the position of the validation rule in the Model Validation Configuration dialog, thus indicating which rule is applied and violated
- nnnn is the number of the violation message.

Therefore, messages with the ID MVR01nnnn indicate that the Element: Well-Formedness checkbox is selected and a violation of that rule has been detected. Messages with the ID MVR0Annnn indicate that the Feature: OCL Conformance checkbox (10th in order on the dialog, or A in hexadecimal) is selected and a violation of that rule has been detected.
6.6.2 Rules Reference

Model Validation works against a set of validation rules, arranged in the following groups:

- **(Element, Relationship, Feature, Diagram): Well-Formedness**
  Checks whether or not an element, relationship, feature or diagram is well-formed. This group of rules includes checks such as whether the item is a valid UML item and whether a diagram contains valid elements within it.

- **Element: Composition**
  Checks 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**
  Checks whether or not the item has the correct UML properties defined, and whether the properties contain incorrect or conflicting values; for more information on these properties see the Custom Properties topic.

- **(Element, Relationship, Feature): OCL Conformance**
  Validates an item against any defined constraints in OCL.

### 6.6.2.1 Well-Formedness

This group of rules checks whether or not an element, relationship, feature or diagram is well-formed. The rules includes checks such as whether the item is a valid UML item and whether a diagram contains valid elements within it. Reported violations include:

<table>
<thead>
<tr>
<th>Violation ID</th>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVR0100001</td>
<td>«Element» is not a valid UML Element</td>
<td>The element is not a recognized UML 2.1.1 element.</td>
</tr>
<tr>
<td>MVR0500001</td>
<td>«Relationship» is not a valid UML Relationship</td>
<td>The relationship is not a recognized UML 2.1.1 relationship.</td>
</tr>
<tr>
<td>MVR0500002</td>
<td>«Relationship» is not legal for «Start Element» --&gt; «End Element»</td>
<td>The relationship between the given start and end elements is not valid for those elements.</td>
</tr>
<tr>
<td>MVR0500003</td>
<td>«Parent Element»:isLeaf=true and cannot be generalized by «Child Element»</td>
<td>The Generalization relationship cannot exist between parent and child elements because the parent element is defined as a leaf element.</td>
</tr>
<tr>
<td>MVR0500004</td>
<td>«Child Element»:isRoot=true and cannot generalize «Parent Element»</td>
<td>The Generalization relationship cannot exist between parent and child elements because the child element is defined as a root element.</td>
</tr>
<tr>
<td>MVR0500005</td>
<td>«Element» cannot generalize self</td>
<td>The element cannot be self-generalized.</td>
</tr>
<tr>
<td>MVR0B0001</td>
<td>Statechart violation: «extended information»</td>
<td>The State diagram contains a UML violation; see the extended information for more information about the detected violation.</td>
</tr>
</tbody>
</table>

### 6.6.2.2 Element Composition

This group of rules checks 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.

<table>
<thead>
<tr>
<th>Error ID</th>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVR0200001</td>
<td>«Element» is missing required child element «Child Element».</td>
<td>The element is missing a child element of type Child Element.</td>
</tr>
<tr>
<td>MVR0200002</td>
<td>Invalid UML package child.</td>
<td>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).</td>
</tr>
<tr>
<td>MVR0200003</td>
<td>Invalid child «Child Element name» («Child Element Type»).</td>
<td>The child element is invalid on the tested parent element.</td>
</tr>
</tbody>
</table>
6.6.2.3 Property Validity

This group checks whether or not an element, relationship or feature has the correct UML properties defined for it and whether they contain incorrect or conflicting values. For more information about these properties see the Custom Properties topic.

<table>
<thead>
<tr>
<th>Error ID</th>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVR030001</td>
<td>«Element»: «Property» property is undefined</td>
<td>The element property contains no value.</td>
</tr>
<tr>
<td>MVR030002</td>
<td>«Element»: «Property» property has invalid value: &quot;«Value»&quot;</td>
<td>The element property contains an invalid value.</td>
</tr>
<tr>
<td>MVR030003</td>
<td>«Element»: isLeaf=true and cannot be abstract</td>
<td>The element's isLeaf and isAbstract properties are both set to true, which is invalid.</td>
</tr>
<tr>
<td>MVR060001</td>
<td>«Relationship»: «Property» property is undefined</td>
<td>The relationship property contains no value.</td>
</tr>
<tr>
<td>MVR060002</td>
<td>«Relationship»: «Property» property has invalid value: &quot;«Value»&quot;</td>
<td>The relationship property contains an invalid value.</td>
</tr>
<tr>
<td>MVR090001</td>
<td>Attribute/AssociationEnd mismatch, «Attribute»: «Mismatch description»,...</td>
<td>The given attribute has an associationEnd of the same name but they differ in the listed details.</td>
</tr>
</tbody>
</table>

6.6.2.4 OCL Conformance

This group validates an element, relationship or attribute against any defined constraints in the Object Constraint Language (OCL). OCL is used to describe expressions on UML models, and to express side-effect free constraints. You can add OCL constraints to any element, relationship or attribute in Enterprise Architect.

<table>
<thead>
<tr>
<th>Error ID</th>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVR040001</td>
<td>OCL violation: «violated OCL»</td>
<td>The element violates the OCL constraint specified.</td>
</tr>
<tr>
<td>MVR070001</td>
<td>OCL violation: «violated OCL»</td>
<td>The relationship violates the OCL constraint specified.</td>
</tr>
<tr>
<td>MVR0A0001</td>
<td>OCL violation: «violated OCL»</td>
<td>The attribute violates the OCL constraint specified.</td>
</tr>
</tbody>
</table>

Important:

To have a valid OCL constraint, the syntax must be correctly formed. If the expression is not correct, Enterprise Architect displays a message stating that the OCL constraint is not valid.

Define OCL Constraints for an Element

You can add OCL constraints to an element using the Properties dialog (Element | Properties). Select the Constraints tab, click on the Type drop-down arrow and select OCL.
To perform an OCL Validation, display the Model Validation Configuration dialog and select the **Element: OCL Conformance** checkbox. Any OCL violations are recorded in the Model Validation Output window.

### Define OCL Constraints for a Relationship

You can add OCL constraints to a relationship using the Properties dialog (right-click, `<type> Properties`). Select the Constraints tab, click on the **Type** drop-down arrow and select **OCL**.
To perform an OCL Validation, display the Model Validation Configuration dialog and select the Relationship: (OCL) Conformance checkbox. Any OCL violations are recorded in the Model Validation Output window.

**Define OCL Constraints for an Attribute**

You can add OCL constraints to a feature using the Properties dialog (Element | Attributes). Select the Constraints tab, click on the Type drop-down arrow and select OCL.
To perform an OCL Validation, display the Model Validation Configuration dialog and select the Feature: (OCL) Conformance checkbox. Any OCL violations are recorded in the Model Validation Output window.
6.7 Model Sharing and Team Deployment

Introducing Team Development

Enterprise Architect 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 can be undertaken using two different schemas for deployment, using either:

- EAP based repositories or
- DBMS server based repositories.

Replication requires the use of 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 that enables 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 only as required. It is recommended that a backup is carried out prior to replication.

XMI Import Export

XMI Import/Export can be used to model discrete packages that can be exported and shared between developers. XMI enables the export of packages into XML files which can then be imported into any model.

Package control can be used to set up packages for version control and to enable batch export of packages using XMI. Version Control enables a repository to be maintained by a third-party source code control application that 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.

Further Information


Note:

DBMS Repository support and User Security are only available with the Corporate edition of Enterprise Architect.
6.7.1 Share Enterprise Architect Projects

**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 Enterprise Architect 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.

You can 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**) 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).
6.7.2 Share Projects on Network Drive

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 accepts a number of concurrent connections without issue, although there can be occasional 'lock-outs' when one user tries 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 can occur. It is best to enable 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 might require repair to compensate for the sudden inconsistency. A repair facility is provided (select the Tools | Manage .EAP File | Repair .EAP File menu option) to carry out this task. This only applies to the file-based version of Enterprise Architect; the DBMS-based version does not suffer this problem.
6.7.3 Distributed Development

Enterprise Architect supports distributed development using two different techniques, as described below.

Replication

Use the Replication features to enable 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 topic.

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 necessary 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 (e.g. customer visible, internal release only).
4. You can roll-back parts of a model as required.
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/Export menu options (below) to access this feature; they are available through the Project | Import/Export submenu. Also see the XMI topic for further information on XMI-based import and export.

Note:

XMI based import/export is UML1.3 / XMI1.1 compliant. You can also write XML based tools to manipulate and extract information from XML files to enhance the development process.
6.7.4 Replication

In addition to sharing Enterprise Architect projects in real time over a network, you can also share projects using Replication, options for which are available through the Tools | Manage .EAP File menu.

Replication enables 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 sections of this topic carefully.

Enterprise Architect Merge Rules

Enterprise Architect follows these rules in merging:

- Additions are cumulative; i.e. two replicas each creating three new Classes result in six new Classes after merging.
- Deletions prevail over modifications; if one replica changes a Class name and other deletes the Class, performing a merge results in both files losing the Class.

Conflicting modifications appear in the Resolve Replication Conflicts dialog (Tools | Manage EAP File | Resolve Replication Conflicts menu option). See Resolve Conflicts for details on how to deal with conflicting modifications.

Use Replication

To use replication, follow the steps below:

1. Convert the base project to a design master using the Tools | Manage .EAP File | Make Design Master menu option.
2. Create replicas from the design master using the Tools | Manage .EAP File | Create New Replica menu option.
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.

Upgrades and Replicas

When you upgrade your version of Enterprise Architect, you must not open a replica until you have opened the design master and then synchronized the replicas with the master. You cannot directly upgrade a replica.

Avoid Change Collisions

If two or more people make changes to the same element, e.g. a Class, Enterprise Architect arbitrarily overwrites one person's change with another'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 want to disable the replication features, use the Tools | Manage .EAP File | Remove Replication menu option. Make sure you back up all your files first!
6.7.4.1 **Design Masters**

A *design master* is the first converted Enterprise Architect project that supports replication. From the design master you create replicas that can be modified independently of the master project and re-merged later.

**Create a Design Master**

To create a design master, follow the steps below:

1. Take a back-up of the required Enterprise Architect project.
2. Select the project in the Project Browser.
3. Select the **Tools | Manage .EAP File | Make Design Master** menu option and follow the on-screen instructions.

**Tip:**

Replication is a powerful means of using Enterprise Architect in a work group or multi-user scenario.

6.7.4.2 **Create Replicas**

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Manage Replicas** permission to create a replica.

To create a replica, follow the steps below:

1. First create a *design master* then select the **Tools | Manage .EAP File | Create New Replica** menu option and follow the on-screen instructions.
2. This process creates a replica of the current project which can then be modified independently, and afterwards re-merged with the main project.

6.7.4.3 **Synchronize Replicas**

To copy changes from one member of the replica set to another, use the **Synchronize Replicas** menu option. Note that information is copied both ways, including deletes, updates and inserts.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have **Manage Replicas** permission to synchronize a replica.

**Synchronize a Replica**

To synchronize a replica and a design master, follow the steps below:

1. Open the design master project file.
2. Select the **Tools | Manage .EAP File | Synchronize Replicas** menu option.
3. Locate and select the required replica to merge the open project and the replica.

**Note:**

When you synchronize, both projects end up containing identical information.

**Change Collisions**

Note that if two or more people work on the same element (or package or diagram) 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 can also use the **Tools | Manage .EAP File | Resolve Replication Conflicts** menu option.
6.7.4.4 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 remove all replication features.

Remove Replication

To remove replication, follow the steps below:

1. If a repository is not open, the menu option for removing replication is not enabled. A temporary repository (not the one having replication removed) must be open at the time. Ensure you have a repository open at the time of creation.
2. Select the **Tools | Manage .EAP File | Remove Replication** menu option, to open the Remove Replication Wizard.
3. Enter the full path and file name of the project to have replication removed. Click on the **Next** button.
4. Enter the full path and file name of the base Enterprise Architect model (with no replication) to act as template. Click on the **Next** button.
5. Enter the full path and required file name for the output file. Click on the **Next** button.
6. Select whether to have a log file created, and enter a file name for the log file.
7. Click on the **Run** button to begin removing replication. Enterprise Architect creates a new project containing all the model information.

Your model has now had replication removed, and should be considerably smaller.

6.7.4.5 Upgrade Replicas

With new releases of Enterprise Architect there could be changes to the underlying project structure, such as more tables or changed queries. If you are using replicas to share and work with Enterprise Architect projects, it is very important that you open the **design master** before opening any of the replicas with an updated version of Enterprise Architect.

**Warning:**

Upgrading Replicas takes special care!

Changes to the database design in a replica set can 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 at best does nothing, and at worst causes 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.

6.7.4.6 Resolve Conflicts

When two or more people have changed the same element between synchronization points, Enterprise Architect has trouble resolving which change to accept and which to discard. The choice is made based on some rules within the JET replication manager, but the discarded changes are also stored so you can 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.
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 can very easily be modified through the normal Enterprise Architect interface; for example, by moving a diagram element.

The only case in which this dialog should be used is where a substantial piece of information has been overridden by another user, and you want to retrieve it. Follow the steps below:

1. In the Table with Conflicts list, click on the entry that is likely to contain the lost information.
2. Click on each entry in the Conflicting Records list.
3. When the lost information appears in the Conflict Details list, click on the Overwrite with Conflict button.
6.8 XMI Import and Export

What is XMI?

XML Metadata Interchange (XMI) is an open standard file format that enables 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, Enterprise Architect 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 or exporting to XMI 1.0, some loss of data occurs due to the limitations of XMI 1.0. XMI 1.1 has support for UML 1.3, whereas XMI 2.1 has support for UML 2.0 and UML 2.1.

With XMI, model details can be exchanged between different UML tools and other tools that are capable of using XML. 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 can be exported from and imported into Enterprise Architect models. This greatly improves the flexibility and robustness of Enterprise Architect models by enabling Analysts and Modelers to externalize model elements in XMI for version control, distributed development, post-processing and transferring packages between models. When performing Enterprise Architect-to-Enterprise Architect transfers, ensure that the XMI version selected is 1.1 or 2.1.

XMI Tasks

Tasks you might perform in importing and exporting XMI include:

- **Setting XML Options**: XMI import, export and package control all rely on saving and loading XML files; you can set a number of options to streamline this process
- **Exporting a package** to XMI in XMI 2.1 (and earlier)
- **Importing from XMI** with support for XMI 2.1 (and earlier)
- **Setting up controlled packages**
- **Manually controlling a package** by linking it to an XMI file
- **Batch exporting** controlled packages
- **Batch importing** controlled packages
- **Factoring in the limitations of XMI**
- **Applying a UML Data Type Definition** (DTD)

For further information on XMI, including specifications, see the OMG [XML/XMI Technology](#) topic.

Notes:

- XMI 2.1 exported by Enterprise Architect 7.0 (or later) might not be correctly imported into earlier versions of Enterprise Architect.
- When you select to apply a DTD during an XMI 1.1 export, the UML_EA.DTD file is written to the output directory into which the XML files are written (unless the UML_EA.DTD file is already present in the directory). No error is generated if the UML_EA.DTD file is not present in this directory during the XMI export.

However, an error does occur if you are importing an XMI 1.1 file that has been exported with the UML_EA.DTD file, and the UML_EA.DTD file is not present in the same directory as the XMI file.
<table>
<thead>
<tr>
<th>Important:</th>
</tr>
</thead>
<tbody>
<tr>
<td>When you import an XML file over an existing package, ALL information in the current package is deleted first. Before you import the XML file, please make sure you do not have important changes that you do not want to lose.</td>
</tr>
</tbody>
</table>
6.8.1 Export to XMI

You can export a package to an XMI (XML based) file. This enables you to move Enterprise Architect Model elements between models, for distributed development, manual version control and other benefits. It also enables limited export of Enterprise Architect model elements to Rational Rose and other tools that implement the UML 2.1 XMI 2.1 standard, the UML1.4 XMI 1.2 standard, or the UML 1.3 XMI 1.1 / XMI 1.0 standard.

For more information regarding the limitations of XMI exporting read the Limitations of XMI topic.

Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Export XMI permission to export to XMI.

Export a Package to XMI

To export a package to XMI, follow the steps below:

1. In the Project Browser window, select the package to export.
2. Either:
   - Right-click and select the Import/Export | Export Package to XMI menu option, or
   - Select the Project | Import/Export | Export Package to XMI menu option.
   The Export Package to XMI dialog displays.

3. In the Filename field, type the directory path and filename into which to output the XMI file.
4. In the Stylesheet field, click on the drop-down arrow and select a stylesheet to post-process XMI content before saving to the file.
5. Select the Export Diagrams checkbox to export diagrams in the file.
6. Select the Export Alternate Images checkbox to export the alternative images used in the diagrams.
7. Select the Format XMI Output checkbox to format output into readable XML (this takes a few more seconds at the end of the run).
8. Select the Write Log file checkbox to write a log of export activity (recommended); the log file is saved to the directory into which you export the XMI file.
9. If using XMI 1.1, select the Use DTD checkbox to use the UML1.3 DTD (recommended). Setting this option validates the correctness of the model and checks that no syntactical errors have occurred.
more information regarding the use of DTDs, see the UML DTD topic.

10. Leave the **Enable full EA Roundtrip** checkbox selected to keep data specific to Enterprise Architect in the XMI file.

11. In the **XMI Type** field, click on the drop-down arrow and select the appropriate XMI format:
   - **XMI 1.1**, to generate output in XMI 1.1 format
   - **XMI 2.1**, to generate output in XMI 2.1 format.

10. Select the **Unisys/Rose Format** checkbox to export in Rose UML 1.3, XMI 1.1 format.
11. Select the **Exclude EA Tagged Values** checkbox to exclude Enterprise Architect-specific information from the export to other tools.
12. Click on the **Export** button.

**Notes:**

- **XMI 2.1** exported by Enterprise Architect 7.0 (or later) might not be correctly imported into earlier versions of Enterprise Architect.

- When you select to apply a Data Type Definition (DTD) during an XMI 1.1 export, the UML_EA.DTD file is written to the output directory into which the XML files are written (unless the UML_EA.DTD file is already present in the directory). No error is generated if the UML_EA.DTD file is not present in this directory during the XMI export.

**Important:**

When exporting and importing with XMI 1.0 with Enterprise Architect, some loss of data occurs due to the limitations of XMI 1.0.
6.8.2 Import from XMI

You can import a package from an XMI (XML based) file. This enables you to move Enterprise Architect Model elements between models, for distributed development, manual version control and other benefits.

**Important:**

- When you import an XML file over an existing package, ALL information in the current package is deleted first. Before you import the XML file, please make sure you do not have important changes that you do not want to lose.
- If you are importing an XMI 1.1 file that was previously exported with a UML_EA.DTD file, the UML_EA.DTD file must be present in the directory into which the XMI file is being written. An error occurs if the UML_EA.DTD file is absent.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have Import XMI permission to import packages from XMI.

You can import the following formats:

- 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)

Enterprise Architect can also import the *.emx and *.uml2 files generated by tools such as Rational Software Architect (RSA) and Rational Software Modeler (RSM).

**Import From XMI**

To import a package from XMI, follow the steps below:

1. In the Project Browser window, select the package into which to import the file.
2. Either:
   - Right-click and select the Import/Export | Import Package from XMI menu option, or
   - Select the Project | Import/Export | Import Package from XMI menu option.

The Import Package from XMI dialog displays.
Note:

To import .emx or .uml2 files, click on the Import EMX / UML2 Files button. Go to Import EMX/UML2 Files.

3. In the Filename field, type the directory path and filename from which to import the XMI file.
4. Select the Import diagrams checkbox to import diagrams.
5. Select the Strip GUIDs checkbox to remove Universal Identifier information from the file on import. This enables the import of a package twice into the same model; the second import requires new GUIDs to avoid element collisions.
6. Select the Write log file checkbox to write a log of import activity (recommended); the log file is saved in the directory from which the file is being imported.
7. Click on the Import button.
6.8.3 Import EMX/UML2 Files

Rational Software Architect (RSA) enables you to add many UML models under a single root. These models can have cross references between them. However, RSA cannot save the entire root as one file; it saves each UML model as a separate EMX file. This means that an EMX file with cross-references is not self-contained as it references elements in another EMX file.

In releases earlier than release 7.0, Enterprise Architect treats each EMX file as a separate model and hence does not allow for cross-references between them. From release 7.0, Enterprise Architect enables these cross-references. You therefore have the option of importing a single EMX/UML2 file or a group of EMX/UML2 files. This option enables you to select a group of related files and import them together, thereby retaining the cross-references between the different files.

To import single or multiple *.emx/*.uml2 files into Enterprise Architect, follow the steps below:

1. In the Project Browser window, select the package into which to import the file.
2. Either:
   - Right-click and select the Import/Export | Import Package from XMI menu option, or
   - Select the Project | Import/Export | Import Package from XMI menu option.
   The Import Package from XMI dialog displays.

3. Click on the Import EMX / UML2 Files button. The Import Package from XMI dialog redisplays, formatted for .EMX/.UML2 file imports.
4. Click on the [...] (Browse) button next to the **Directory** field. The **Select Import EMX / UML2 File(s)** dialog displays, which enables you to select multiple files.

5. Select the file or files (use [Ctrl]+click or [Shift]+click to select several files) and click on the **Open** button. The **Import Package from XMI** dialog redisplays; the **Selected File(s)** panel lists the selected files.

6. Select the **Write Log File** checkbox to write a log of import activity (recommended); the log file is saved in the directory from which the file is being imported, with the name `import.log`.

7. Click on the **Import** button. Enterprise Architect indicates the progress of the import in the **Import Progress** field.
6.8.4 Limitations of XMI

Whilst 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 can tolerate using the standard syntax. A lot of information from an Enterprise Architect Model must be converted to Tagged Values, which import into other modeling systems as additional information or are ignored completely.

Enterprise Architect can both generate and read XMI 1.0 and 1.1 using UML 1.3 format, XMI 1.2 using UML 1.4 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 - Enterprise Architect 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 might occur.
- The way packages are arranged in different models can impact 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, and State Operations import but do not appear in diagrams. In addition, Rational Rose only supports import of a full project, not a single package.
- For best results, it is recommended that you keep the model elements to export to Rose simple and conforming as closely as possible to the UML 1.3 specification.
6.8.5 The UML DTD

When you import or export Enterprise Architect packages to XMI, the import or export process can be validated using a Data Type Definition (DTD). The XML parser uses this document to validate the correctness of the model and to check that no syntactical errors have occurred. It is always best to use a DTD when moving packages between Enterprise Architect models as it ensures correctness of the XMI 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 of this with some additional element extensions for non-standard UML types, such as testing details.

Whenever you read an XML file, the XML parser looks in the current directory for the DTD - if specified - using the DOCTYPE element in the XML file. If the parser cannot find the DTD, it records an error and aborts processing. You must ensure the UML_EA.DTD file is in the current XML output path (generated by default).
### 6.8.6 Controlled Packages

Controlled packages are a powerful means of ‘externalizing’ parts of an Enterprise Architect 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 Enterprise Architect repository.
- Support version control by writing model elements in XML text files suitable for version control using standard version control software. Using XMI this way enables you to manually connect to third-party version control software outside the Enterprise Architect environment. Enterprise Architect internally supports the configuration of version control through SCC and CVS.
- Support import and export of model elements between different models; for example, a Class library can 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 XML-compliant output that can 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 Project Browser with a small colored rectangle to the left of the package icon, as shown below for the CIM package:

A controlled package is a package 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.

**Note:**

When you select to apply a Data Type Definition (DTD) during an XMI 1.1 export, the UML_EA.DTD file is written to the output directory into which the XML files are written (unless the UML_EA.DTD file is already present in the directory). No error is generated if the UML_EA.DTD file is not present in this directory during the XMI export.

**Important**

If you are importing an XMI 1.1 file that was previously exported with a UML_EA.DTD file, the UML_EA.DTD file must be present in the directory into which the XML file is being written. An error occurs if the UML_EA.DTD file is absent.

### 6.8.6.1 Controlled Package Menu

The **Package Control** sub-menu is available from the Project Browser **Package** context menu. This menu is for a package that itself is not under version control (but that might contain child packages that are under version control). For a package that is directly under version control, see the **Package Version Control Menu** topic.

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure (various settings for the package) [Ctrl]+[Alt]+[P]</td>
<td>Display the <strong>Package Control Options</strong> dialog, which enables you to specify whether this package (and its child packages) is controlled and which file it is controlled through.</td>
</tr>
<tr>
<td>Manage Baselines [Ctrl]+[Alt]+[B]</td>
<td>Create a <strong>Baseline</strong> of the current package, or compare the current package with a previous Baseline.</td>
</tr>
<tr>
<td>Check In Branch</td>
<td>For the selected branch of the model, (i.e. the selected package and all of its child packages) display the <strong>Select Packages to Check In</strong> dialog, listing all version controlled packages within that branch that are checked out to you. You can then select packages in the displayed list, to be submitted for check-in.</td>
</tr>
<tr>
<td>Save package to file [Ctrl]+[Alt]+[S]</td>
<td><strong>Save a controlled package</strong> to an XMI file.</td>
</tr>
<tr>
<td>Load package from file [Ctrl]+[Alt]</td>
<td><strong>Load</strong> a previously-saved XMI file.</td>
</tr>
</tbody>
</table>
### Menu Option & Function Keys
- **+L**
  - **Use to**
    - View package XMI [Ctrl]+[Alt]+[X]: Display the package XMI after the package has been exported to XML.
- **Compare with XMI file**
  - **Use to**
    - (Package not under version control.) Compare the current package with a previously-saved XMI file of the package.
- **Compare with Controlled Version**
  - **Use to**
    - (Version controlled package.) Compare the current package with the head revision of the version-controlled package file.
- **Add Branch to Version Control**
  - **Use to**
    - Apply version control to all packages within a selected model branch in a single operation.
- **Export as Model Branch**
  - **Use to**
    - Export a newly created model branch from your own private copy of a model.
- **Import a Model Branch**
  - **Use to**
    - Retrieve a model branch and import it into either the source model or another model.
- **Get package (for version control)**
  - **Use to**
    - Enables you to gain access from packages in the version-controlled repository that is currently available in your model.
- **Get All Latest (for version control)**
  - **Use to**
    - Retrieve the latest version of the package from the repository. Available only for packages that are checked in.
    - The alternative option Get Latest - if displayed - is not intended for sharing .EAP files and should only be used when users have their own individual databases.
- **Version Control Settings**
  - **Use to**
    - Display the Version Control Settings dialog.
- **Update Package Status**
  - **Use to**
    - Provide a bulk update on the status of a package. This includes status options such as Proposed, Validate and Mandatory.

### 6.8.6.2 Configure Packages
Before you can use a controlled package, you must configure it with options such as the filename to save to/load from, the type of export and the version number. Once a package is configured and marked as controlled, it is displayed in the Project Browser with a small colored rectangle next to the package icon, indicating it is a controlled package. In the example below, the RQ01: Formal Requirements package is a controlled package.

![Image of Project Browser with RQ01: Formal Requirements package highlighted]

#### Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Configure Packages permission to configure controlled packages and package properties.

### Configure a Controlled Package
To configure a controlled package, follow the steps below:

1. In the Project Browser, right-click on the package to control or configure. The context menu displays.
2. Click on the Package Control | Configure menu option. The Package Control Options dialog displays.
3. Set the required options, as follows:
   - Select the **Control Package** checkbox to indicate that this is a controlled package
   - Click on the **Version Control** drop-down arrow and select the version control repositories; this connects the package to a specific version control configuration
   - In the **XMI Filename** field, type or browse for the path and XMI file for importing and exporting XMI files. The field accepts Local Path Substitution strings; for example, use an XMI local path definition where:
     
     ```
     myLocalPath="C:\Documents and Settings\John\Desktop\EA Models"
     Then %myLocalPath%\CIM.xml is equivalent to C:\Documents and Settings\John\Desktop\EA Models\CIM.xml
     ```
   - In the **UML/XMI Type** field, click on the drop-down arrow and select the type of XMI generated; options include Enterprise Architect XMI/UML 1.3 - currently only Enterprise Architect UML 1.3 is supported for complete import/export round tripping of packages
   - In the **Version ID** field, type the version ID number
   - In the **Owner** field, type or select the name of the package owner
   - If required, click on the **Use DTD** checkbox to use a Data Type Definition (DTD)
   - If required, click on the **Log Import/Export** checkbox to log import and export activity to a log file
   - If required, click on the **Batch Import** checkbox to mark the package as a Batch Import package
   - If required, click on the **Batch Export** checkbox to mark the package as a Batch Export package.

4. Click on the **OK** button to set the Package Control options.

**Note:**

For batch import, the file date of the XMI file is stored. You can bypass the batch import if the file date of the last import matches that of the current file (i.e. there is no change).

### 6.8.6.3 Remove Package from Control

If required, you can remove the control from a package. Before removing the package control, you must **check in** the package if it is being used for version control.

To remove control from a package:

1. In the **Project Browser** window, right-click on the controlled package. The context menu displays.
2. Select the **Package Control | Configure** menu option, or press **[Ctrl]+[Alt]+[P]**. The **Package Control Options** dialog displays.
3. Deselect the **Control Package** checkbox. If the package is under version control, this sets the **Version Control** field to **(None)**.
4. Click on the OK button to remove package control. Package control for the selected package has now been removed.

**Note:**

When disconnecting a package from version control, the association between the package and the exported XML file is removed from your model. However, the XML file itself is not removed from version control, nor is it deleted from your local version control working copy folder. This is because it is possible for another model to be using the version controlled package and still referencing the associated version controlled XML file.

### 6.8.6.4 Save a Package

You can save a controlled package to an XMI file. Once you have correctly configured the package, follow the steps below:

1. In the Project Browser window, right-click on the package to save. The context menu displays.
2. Select the Package Control | Save Package to File menu option.
3. The export process executes 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 must check out the XMI file first to enable Enterprise Architect to overwrite the existing version.

### 6.8.6.5 Load a Package

Using the Controlled Packages facility, you can save and load packages to a named file. If a package has been marked for control it is displayed in the Project Browser with a red rectangle to the left of the package icon. If you have previously saved a controlled package, you can reload it using the Load package from file menu option.

To load a controlled package, follow the steps below:

1. In the Project Browser window, right-click on the package to load. The context menu displays.
2. Select the Package Control | Load package from file menu option.
3. If you have configured the package control details, Enterprise Architect prompts you to confirm the import.

**Warning:**

Importing deletes the current package entirely from the model, and the action cannot be undone. You must be careful not to lose any current changes or information.
4. Click on the Yes button to confirm the import. The current package is deleted and the saved package is imported.

6.8.6.6 Batch XMI Export

You can export a group of controlled packages in one step, using the Batch XMI Export facility.

To export a group of controlled packages, follow the steps below:

1. Select the Project | Import/Export | Batch XMI Export menu option. The Batch XMI Export dialog displays.

   ![Batch XMI Export Dialog]

   - Select Packages to Include in Export
     - Business Domain Model: Project Models
     - Requirements Model: Project Models
     - System Model: Project Models

   - Select All
   - Run Export
   - Save Settings
   - Close
   - Help

2. Select the checkbox against each package to include in this export run. Select the Select All checkbox to select all packages in the list.

3. To save this configuration as the default, click on the Save Settings button.

4. Click on the Run Export button.
   
   Enterprise Architect cycles through each checked package and exports it using the options specified in the Controlled Package dialog. As long as a valid filename exists, Enterprise Architect exports the package to XML and proceeds to the next package.

6.8.6.7 Batch XMI Import

You can import a group of controlled packages in one step, using the Batch XMI Import facility.

To import a group of controlled packages, follow the steps below:

1. Select the Project | Import/Export | Batch XMI Import menu option. The Batch XMI Import dialog displays.
2. Select the checkbox against each package to include in the import. Select the **Select All** checkbox to select all packages in the list.

**Tip:** To avoid re-importing the same module multiple times, select the **Check file date before import** checkbox. Enterprise Architect then does not import a file if the last import file date matches that of the one currently on disk.

3. To save this configuration as the default, click on the **Save Settings** button.
4. Click on the **Run Import** button. Enterprise Architect cycles through the packages and imports each selected package.

As Enterprise Architect processes each package, it updates the **Status** column against each package name on the **Batch XMI Import** dialog.

- If the import is successful, Enterprise Architect updates the package status to **Imported**.
- If the import is unsuccessful, Enterprise Architect updates the package status to **Not Imported**.

Common reasons for an import to fail include:

- The package not being correctly configured
- The last import file date matches the import date of the file currently on disk.

### 6.8.6.8 Manual Version Control with XMI

You can use XMI to support version control by writing model elements in XML text files suitable for use with standard version control software. Using XMI in this manner enables you to manually connect to third-party version control software outside the Enterprise Architect environment. Enterprise Architect internally supports the configuration of version control through SCC and CVS configurations.

To use XMI for version control, you must first:

1. Select suitable packages in the **Project Browser** window, to 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 the steps below:

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 the steps below:

1. Save the current version first (**important**, because the package is completely deleted during the import process) and manually update the version control system if necessary.
2. Get the required package version from the version control system.
3. Select the package to reload.
4. Select the **Package Control | Load package from file** menu option to import the previous version.
   
   Enterprise Architect deletes the controlled package and restores the previous version.
6.9 CSV Import and Export

You can import and export information about Enterprise Architect elements in CSV format. You must define CSV specifications to do this.
6.9.1 CSV Specifications

To import and export element data from Enterprise Architect using CSV files, you must first set up one or more file specifications. A file specification lists the fields in the order they are imported or exported, the filename (optional) and the delimiter between columns. Once you have defined one or more specifications, one can be selected in the CSV Import/Export dialog as the current specification to apply during an import or export action.

CSV only imports and exports elements (within packages) and their properties; items such as Class attributes cannot be imported or exported through this mechanism. XMI provides a solution to this limitation, as does use of the Automation Interface.

To define a specification, select the Project | Import/Export | CSV Import/Export Specifications menu option. The CSV Import/Export Specification dialog displays.

The CSV Import/Export File Specification dialog provides the following functionality:

- Specification Name: Select the unique name for this specification.
- Notes: Additional notes can be added.
- Delimiter: Specify the character delimiter to use between record fields.
- Default Filename: Enter the default filename if desired.
- Default Direction: Select the direction of the file specification (Import or Export).
- Default Types: Specify the default types of elements to import or export.
- Available Fields: Select the available element fields to include in the specification.

The CSV Import/Export File Specification dialog provides the following functionality:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification Name</td>
<td>Select the unique name for this specification.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Specify the character delimiter to use between record fields.</td>
</tr>
</tbody>
</table>
### CSV Import and Export

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>If a field contains an instance of the delimiter, the field is exported wrapped in &quot;&quot; (quotation marks) and all instances of &quot;&quot; in the field are doubled (i.e. &quot;&quot; becomes &quot;&quot;&quot;&quot;).</td>
</tr>
<tr>
<td>Notes</td>
<td>Record a brief description of the specification.</td>
</tr>
<tr>
<td>Default Filename</td>
<td>Select the default filename.</td>
</tr>
<tr>
<td>Default Direction</td>
<td>Set the default action - <strong>Import</strong> or <strong>Export</strong>. A specification can be used in either direction, but this enables you to set the default type.</td>
</tr>
<tr>
<td>Default Types</td>
<td>Limit the element types being exported, by entering a comma-separated list: e.g. <code>class,requirement,component,node,object</code>. <strong>Note:</strong> If you specify element types, ONLY elements of those types are exported or imported. Therefore, in order to enable the <strong>Preserve Hierarchy</strong> option to operate (if selected) you must include <code>Package</code> as an element type. Otherwise there are no packages in which to preserve the hierarchy. If you do not specify any default element types, all elements including <code>Packages</code> are exported or imported and the hierarchy can be preserved.</td>
</tr>
<tr>
<td>Preserve Hierarchy</td>
<td>Include fields generated by Enterprise Architect to embed/reconstruct the package hierarchy. Please see the <a href="#">Using Preserve Hierarchy</a> section for more details.</td>
</tr>
<tr>
<td>Available Fields</td>
<td>Select from a list of possible record fields, not yet allocated.</td>
</tr>
<tr>
<td>File Specification</td>
<td>List the record fields (in order) already assigned.</td>
</tr>
<tr>
<td>Add Field</td>
<td>Move all selected fields in the top list to the bottom list.</td>
</tr>
<tr>
<td>Remove Field</td>
<td>Move all selected fields in the bottom list back to the available list.</td>
</tr>
<tr>
<td>New</td>
<td>Create a new specification.</td>
</tr>
<tr>
<td>Save</td>
<td>Save changes to the currently selected specification.</td>
</tr>
<tr>
<td>Save As</td>
<td>Save the current specification with a new name.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the current specification.</td>
</tr>
<tr>
<td>Close</td>
<td>Close this dialog.</td>
</tr>
</tbody>
</table>

**Note:** In **Available Fields** and **File Specification**, the record fields **Created Date** and **Modified Date** are not set when imported from CSV.

### Using Preserve Hierarchy

When selected, the **Preserve Hierarchy** option inserts two fields into the CSV specification that are:

- automatically populated by Enterprise Architect on export and
- used to reconstruct the exported package's hierarchy upon import.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_KEY</td>
<td>A unique identifier for the current element. <strong>Note:</strong> This key is unique per export; subsequent exports produce different keys for the same set of elements.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSV_PARENT_KEY</td>
<td>The corresponding CSV_KEY of the current element's parent. If the field is left blank or references a non-existent CSV_KEY, the element is added to the top level of the package.</td>
</tr>
</tbody>
</table>

**Note:**

It is highly recommended that you do not change these fields by hand if they have been automatically generated by Enterprise Architect's CSV exporter.
6.9.2 CSV Export

It is possible to export information about Enterprise Architect 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 the steps below:

1. In the Project Browser, right-click on the package containing the elements to export.
2. Select the Import/Export | CSV Import/Export menu option. The CSV Import/Export dialog displays.

3. Set the required options, as outlined below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Confirm the name of the current selected package.</td>
</tr>
<tr>
<td>Specification</td>
<td>Specify the name of the export specification to use.</td>
</tr>
<tr>
<td>Edit/New</td>
<td>Edit the export specification or create a new one.</td>
</tr>
<tr>
<td>File</td>
<td>Specify the filename to export to.</td>
</tr>
<tr>
<td>Types</td>
<td>List the element types to export: leave blank for all, or enter a comma-separated list of types.</td>
</tr>
</tbody>
</table>

**Note:**

If you specify element types, ONLY elements of those types are exported. Therefore, in order to enable the Preserve Hierarchy option in the specification to operate (if selected) you must include Package as an element type. Otherwise no packages are exported in which to preserve the hierarchy.

If you do not specify any element types, all elements including Packages are exported and the hierarchy can be preserved.
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Select the <strong>Export</strong> radio button to export to file.</td>
</tr>
<tr>
<td>Print Results</td>
<td>Print out the result list.</td>
</tr>
<tr>
<td>View File</td>
<td>View the CSV file with the default Windows application for CSV files.</td>
</tr>
<tr>
<td>Run</td>
<td>Perform the export.</td>
</tr>
<tr>
<td>Close</td>
<td>Exit this dialog.</td>
</tr>
</tbody>
</table>
6.9.3  CSV Import

It is possible to import information about Enterprise Architect elements in CSV format. Once you have defined a CSV import specification, you can read in major element attributes from a CSV text file.

**Note:**

You import the CSV file into a selected package; if this package or any element within the package has a lock on it, you cannot import the CSV file into it. The **Import** option on the dialog is grayed out.

When importing, Enterprise Architect checks the specification to see if there is a **GUID** field included. If there is, Enterprise Architect attempts to locate the element identified by the GUID and, if successful, updates the current element rather than creating a new one. If no **GUID** field is defined, or Enterprise Architect 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 Enterprise Architect element types.

**Import Data in CSV Format**

To import data in CSV format, follow the steps below:

1. In the **Project Browser**, right-click on the package to import into.
2. Select the **Import/Export | CSV Import/Export** menu option. The **CSV Import/Export** dialog displays.

3. Set the required options; as outlined below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Confirm the name of the current selected package.</td>
</tr>
<tr>
<td>Specification</td>
<td>Specify the name of the import specification to use.</td>
</tr>
<tr>
<td>Edit/New</td>
<td>Edit the import specification or create a new one.</td>
</tr>
<tr>
<td>File</td>
<td>Specify the filename to import from.</td>
</tr>
<tr>
<td>Types</td>
<td>Not used for import.</td>
</tr>
</tbody>
</table>

[Diagram of CSV Import/Export dialog]

**Results**

---

Enterprise Architect User Guide
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Select the <strong>Import</strong> radio button to import from file. (Grayed-out if the package or a child item in the package is locked.)</td>
</tr>
<tr>
<td>Print Results</td>
<td>Print out the result list.</td>
</tr>
<tr>
<td>View File</td>
<td>View the CSV file with the default Windows application for CSV files.</td>
</tr>
<tr>
<td>Run</td>
<td>Perform the import.</td>
</tr>
<tr>
<td>Close</td>
<td>Exit this dialog.</td>
</tr>
</tbody>
</table>
6.10 MOF

Enterprise Architect offers support for exporting packages to XMI under the *Meta-Object Facility (MOF)* 1.3 and 1.4 standards. MOF models are created by assigning the stereotype *metamodel* to the package. MOF models can be exported to MOF 1.3 or MOF 1.4 XMI file specification.

**Background Knowledge**

MOF is an Object Management Group (OMG) standard that originated in the UML, when the OMG required a Meta-Modeling architecture to define the UML. MOF is designed as a four-layered architecture, as illustrated in the following diagram.
Because of the similarities between the MOF-model and UML structure models, MOF meta-models are usually modeled as UML Class diagrams. You can also use the Metamodel page of the Enterprise Architect UML Toolbox to create MOF model elements and connectors. A supporting standard of MOF is XML, which defines an XML-based exchange format.

MOF is a closed, 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. Because an element on the M2 layer is an object (instance of an M3 model element) as well as a Class (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
### 6.10.1 Getting Started

MOF diagrams are Class diagrams that are contained in packages with a metamodel stereotype. To create a MOF diagram, follow the steps below.

1. Create a package to contain your MOF elements.
2. If the Properties dialog does not immediately display, right-click on the package element and select the Properties context menu option.

3. In the Stereotype field type the value `metamodel`.
4. Click on the OK button.
5. Right-click on the package in the Project Browser and, from the context menu, select Add | Add Diagram. Create a Class diagram (the default diagram).
6. Give your MOF Class diagram an appropriate name.
7. In the Enterprise Architect UML Toolbox, select the More tools | UML | Metamodel menu option and add the required `metamodel` elements to the diagram.

The following is an example of a MOF diagram. A MOF diagram can typically contain Package, Class, Enumeration and Primitive elements, and Generalization, Association, Compose and Aggregate relationships.
6.10.2 Export MOF to XMI

Once you have created your MOF diagram you can export the diagram to XMI, specifying the MOF 1.3 or MOF 1.4 standard.

1. Right-click on the package in the Project Browser. The context menu displays.
2. Select the Import/Export | Export Package to XMI file menu option. The Export Package to XMI dialog displays:

   ![Export Package to XMI dialog](image)

   - In the **Filename** field, type a name for the XMI file.
   - De-select the **Enable full EA Roundtrip** checkbox.
   - In the **XMI Type** field, click on the drop-down arrow and select **MOF 1.3** or **MOF 1.4**.
   - Click on the **Export** button and wait until the Progress bar reads **100%**.
   - Once your file has been created, you can view it by clicking on the View XMI button.

   MOF diagrams exported to XMI can be imported using the regular import XMI features of Enterprise Architect. See Import from XMI.

   ![View XMI button](image)
6.11 Version Control

Enterprise Architect supports version control of packages and their component sub-packages to a central version control repository. You can place any individual packages, View nodes or model root nodes under version control.

Features
Version Control provides two key facilities:

- Coordinating sharing of packages between users
- Saving a history of changes to Enterprise Architect packages, including the ability to retrieve previous versions.

System Requirements
Version controlled packages are packages that have been configured for use with version control software. To use version control in Enterprise Architect, a third-party source-code control application is required that controls access to and stores revisions of the controlled packages. Enterprise Architect supports the following version control applications:

- Subversion, which is available from http://subversion.tigris.org/
- CVS, which is available from http://www.tortoisecvs.org/
- Microsoft Team Foundation Server
- Any other version control product that complies with the Microsoft Common Source Code Control standard, version 1.1 or higher.

Set-Up
Before using Enterprise Architect's version control facility, your version control software must be installed on each machine where it is intended to be used.

Typically there are:

- A server component that manages a version control repository
- Client components on the workstations that Enterprise Architect uses to communicate with the server.

A version control client must be installed on every machine where you run Enterprise Architect and want to access your version control system. Once the version control software has been installed and configured, you must define a Version Control Configuration within Enterprise Architect, to use your installed version control product.

Note:
Sparx Systems strongly urge you not to manipulate version controlled package files outside of Enterprise Architect. It is possible to leave the package files in a state that Enterprise Architect cannot recognize.

Usage
There are four basic ways in which the version control facility might be used:

<table>
<thead>
<tr>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Shared model</td>
<td>Users share an Enterprise Architect model, stored in a central .EAP file or DBMS repository. This configuration enables you to view changes to other users' packages without explicitly having to check them out, but by simply refreshing your view of the model.</td>
</tr>
<tr>
<td></td>
<td>• Version control regulates access to packages, and maintains package revision</td>
</tr>
</tbody>
</table>
Use | Description
--- | ---
Multiple Private models | An Enterprise Architect 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.
  - 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** menu option.
Shared packages | Individual users create separate Enterprise Architect models but share one or more packages.
  - Users share packages through version control.
Standard packages | A company might have a standard set of packages which are broadly shared (on a read-only basis).
  - Individual users retrieve packages with the **Get Package** menu option.

**Version Control Indicators**

Packages under version control are identified in the **Project Browser** by icons that indicate the current status of the package.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Indicates that...</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This package is <strong>controlled</strong> and is represented by an XMI file on disk. Version control either is not being used or is not available. You can edit the package.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This package is version controlled and <strong>checked out</strong> to you, therefore you can edit the package.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This package is version controlled and not checked out to you, therefore you cannot edit the package (unless you check the package out).</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>This package is version controlled, but you checked it out whilst not <strong>connected to the version control</strong> server. You can edit the package but there could be version conflicts when you check the package in again.</td>
</tr>
</tbody>
</table>

For example, below, **CVSOO** and **CVSPackage** are configured for version control. **CVSOO** is checked out to you, and **CVSPackage** is not.

- ![Icon](image) **CVSOO**
- ![Icon](image) **CVSPackage**

**See Also**

- **Version Control Basics**
- **Apply Version Control To Models**
- **Version Control and Team Deployment**
- **Version Control Menu**
- **Version Control Setup**
- **Use Version Control**
- **Offline Version Control**
6.11.1 Version Control Basics

The Lock-Modify-Unlock Solution
Many version control systems use a lock-modify-unlock model to address the problem of different authors in a shared source overwriting each other's work. In this model, the version control repository allows only one person to change a file at a time, and access is managed using locks. Harry must lock a file before he can begin making changes to it. If Harry has locked a file, Sally cannot also lock it, and therefore cannot make any changes to that file. All she can do is read the file, and wait for Harry to finish his changes and release the lock. After Harry unlocks the file, Sally can take her turn in locking and editing the file.

The Copy-Modify-Merge Solution
Subversion, CVS and a number of other version control systems use a copy-modify-merge model as an alternative to locking. In this model, each user's client contacts the project repository and creates a personal working copy—a local reflection of the repository's files and directories. Users then work simultaneously and independently, modifying their private copies. In due course, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately a person is responsible for making it happen correctly.

When Locking is Necessary
While the lock-modify-unlock model is generally considered a hindrance to collaboration, there are still times when locking is necessary.

The copy-modify-merge model is based on the assumption that files are contextually merge-able: that is, the files in the repository are line-based text files (such as program source code). But for files with binary formats, such as artwork or sound, it is often impossible to merge conflicting changes. In these situations, it really is necessary for users to take strict turns in changing the file. Without serialized access, somebody ends up wasting time on changes that are ultimately discarded.
6.11.2 Apply Version Control To Models

All Enterprise Architect models are stored in databases - even the .EAP file is a database. In simple, version control terms, the model is a single entity of binary data. It is not practical to apply version control to the database as a whole. Being binary data, it would require the use of the lock-modify-unlock model of version control, which would mean that only a single user at a time could work on any given (version controlled) model.

To overcome this limitation, Enterprise Architect exports discreet units of the model - the packages - as XMI package files, and it is these XMI files, not the .EAP file, that are placed under version control. The XMI file format used by Enterprise Architect dictates that they too be treated as binary files (therefore it is not possible to merge the XMI files either); however, by splitting the model into much smaller parts, this approach enables many users to work on separate parts of the model simultaneously.

When a user checks-out a package, Enterprise Architect sends a command to the version control system to check-out the equivalent XMI file. The version control system then puts the latest revision of the file into the user's working copy directory, overwriting any previous revision of the file in that directory. Enterprise Architect then imports the package file into the model, updating the contents of the existing package in the model.

When checking-in, Enterprise Architect exports the package as an XMI file, overwriting the existing local working copy of the file. The new file is then checked-in to the version control system.

Nested Version Controlled Packages

Nested version controlled packages result in much smaller XMI files being exported for parent packages, as the parent packages’ XMI files do not contain any content for the version controlled child packages.

Version Control of nested packages together with a model structure having small individual packages also provides greater scope for multiple users to work concurrently, as individual users are locking much smaller parts of the model.

Notes:

- Do not place your .EAP files under version control, as this would create problems for you.
- Most version control systems mark their controlled files as read only, unless they are specifically checked-out to you.
- The .EAP file is an MS Jet database, and Enterprise Architect must be able to open this file for read/write access when you load your model. (Enterprise Architect displays an error message and fails to load the model if it is read-only.)
6.11.3 Version Control & Team Deployment

Team deployment and the use of version control is discussed in two Sparx Systems white papers, available on the Sparx Systems web site:


A brief summary of the process is provided below:

1. Install your version control product.
2. Create a version control repository.
3. Create a version control project to be used with your Enterprise Architect project, and check-out a working copy of the project into a local folder. (You must do this for every team member that is accessing the version controlled packages, whether you are using a single shared model or each team member stores his own private copy of the model.)
4. Within Enterprise Architect, define a version control configuration to provide access to the working copy files. Again, each user must do this on their own workstation, as the details are stored within the Windows registry.
5. Configure packages within the Enterprise Architect model for version control. That is, apply version control to individual packages.
6. Check-out and check-in packages as required.

Note:
The name of the version control configuration must be the same across all machines. That is, all version control access to a given Enterprise Architect package must be through version control configurations with the same name, across all models and all users. (It is possible to use multiple version control configurations within the same model, so different packages can still use different version control configurations within the same model, as long as any given package is always accessed via the same version control configuration.)

The easiest way to perform step 4, (throughout the team), is to have one user set up version control on the model and then share that model with the rest of the team.

- In Shared Model deployment, all users connect to a single instance of the model database, so the model is shared automatically.
- In Private Model deployment, it is easiest to distribute copies of the original model (after version control has been set up) to all other members of the team.

Whenever you open a model (Private or Shared) that uses a version control configuration that is not yet defined on your workstation, Enterprise Architect prompts you to complete the definition for that configuration. This typically means specifying the local working copy directory and possibly choosing the version control project associated with this Enterprise Architect project.

Once this has been done, the version controlled packages that already exist in the model are ready for use.

Version Control Branching

Currently, Enterprise Architect does not support Version Control Branching. Work-arounds to achieve similar results might be possible for certain version-control products; contact Sparx Support for advice:

- Registered users - http://www.sparxsystems.com/registered/reg_support.html
- Trial users - support@sparxsystems.com.
### 6.11.4 Version Control Menu

You access the **Version Control** menu through the **Project | Version Control** menu option. It provides the following options:

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Current Package [Ctrl]+[Alt]+[P]</td>
<td>Display the <strong>Package Control Options</strong> dialog, which enables you to specify whether this package (and its children) is version controlled, and which version control configuration applies.</td>
</tr>
<tr>
<td>Version Control Settings</td>
<td>Display the <strong>Version Control Settings</strong> dialog if it is unavailable to you.</td>
</tr>
<tr>
<td>Work Offline</td>
<td>Work <strong>independently of the version control server</strong> if it is unavailable to you.</td>
</tr>
</tbody>
</table>
6.11.5 Version Control Setup

Before using Enterprise Architect's version control facility, your version control product must be installed on each machine where it is intended to be used. Version Control products supported by Enterprise Architect include MS Team Foundation Server, Subversion, CVS or any other version control product that provides an MS SCC-compliant interface.

- Subversion is available from http://subversion.tigris.org/
- CVS is available from http://www.wincvs.org/

**Note:**

If you are using the Corporate version of Enterprise Architect with security enabled, you must also set up permissions to configure and use version control. See the List of Available Permissions topic for further information.

Typically there should be:

- A server component that manages a version control repository
- Client components on the workstations that Enterprise Architect uses to communicate with the server.

A version control client must be installed on every machine where you run Enterprise Architect and want to access your version control system. Once the version control software has been installed and configured, to use your installed version control product you must define a Version Control Configuration within Enterprise Architect.

Version control can be assigned to individual packages, view nodes or root nodes in Enterprise Architect. 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. You can use the Version Control Settings dialog to set up a connection to your version control application.

To set the Version Control Configuration, select the Project | Version Control | Version Control Settings menu option, and see the Version Control Settings Dialog topic.

**See Also**

- Version Control with SCC
- Version Control with CVS
- Version Control with Subversion
- Version Control with TFS

6.11.5.1 Version Control Settings Dialog

The Version Control Settings dialog enables 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. Enterprise Architect 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 Enterprise Architect model. It is also possible to use the same version control configuration across different models, to facilitate sharing 'standard' packages between those models, through the version control system.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have Configure Version Control permission to set up version control options for the current model.

**Setting Up Version Control**

When you display the Version Control Settings dialog for the first time in any given model, it appears as shown below:
To begin defining a new version control configuration, follow the steps below:

1. Click on the **New** button.
2. In the **Unique ID** field, type a suitable name.
3. Against the **Type** field, click on the radio button for the version control product to connect to. At this point, the middle section of the dialog changes to display a collection of fields relating to the type of Version Control Configuration you are defining. Go to the relevant topic below:
   - Version Control with SCC
   - Version Control with CVS
   - Version Control with Subversion
   - Version Control with TFS

To import a previously defined configuration for use in the current model, follow the steps below:

1. Click on the **New** button.
2. In the Unique ID field, click on the drop-down arrow and select one of the previously defined version control configurations.

3. Click on the Save button to save the selected version control configuration in this model.

See Also
- Version Control Nested Packages

6.11.5.1.1 Version Control Nested Packages

In releases of Enterprise Architect later than version 4.5, when you save 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, Enterprise Architect does not modify or delete nested packages; only the top level package is modified.

As a consequence of this behavior, if you check out or get a version controlled package with nested packages not already in your model, you see stubs in the model for the nested packages only. If you select the Get All Latest option from the version control menu, Enterprise Architect populates these new stubs from the version control system.

Using the above technique you can populate a large and complex model from only the root packages, using Get All Latest to recursively iterate through the attached and nested packages.

This is a powerful and efficient means of managing your project and simplifies handling very large models, even in a distributed environment.

It is recommended you do not mix versions of Enterprise Architect later than version 4.5 with earlier versions when sharing a version controlled model. If this is necessary it is best to go to the Version Control Settings dialog and deselect the Save nested version controlled packages to stubs only checkbox, setting Enterprise Architect to the pre-version 4.5 behavior (for the current model only).

<table>
<thead>
<tr>
<th>Model Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>This model is private (for Shared models, it is best to disable this check box)</td>
</tr>
<tr>
<td>☑ Save nested version controlled packages to stubs only (recommended)</td>
</tr>
</tbody>
</table>

6.11.5.2 Version Control with SCC

To set up an SCC version control configuration, you must:

- Set up the source code control provider with SCC, and
- Connect the Enterprise Architect model to version control with SCC.

See also, the topic on version control with SCC when Upgrading at Enterprise Architect 4.5.

Set Up the Source Code Control Provider with SCC

To set up the third-party source code control provider, see 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 Enterprise Architect Model to Version Control with SCC

To connect an Enterprise Architect model to version control, follow the steps below:

1. Open or create the Enterprise Architect model to place under version control.
2. Select the Project | Version Control | Version Control Settings menu option. The Version Control Settings dialog displays.
3. Click on the **New** button.

4. In the **Unique ID** field, type a suitable name. Click on the **SCC** radio button.

5. To the right of the **Local Project** path field, click on the **Select Path...** button. The **Browse for Folder** dialog displays.
6. Locate and click on the local folder in which to keep local working copies of the XML files to be stored in the Version Control repository.

7. Click on the OK button. The Select SCC Provider dialog displays.

8. Click on an SCC provider, and click on the OK button to return to the Version Control Settings dialog.

9. Click on the Save button to save the configuration you have defined.

   The SCC provider is likely to prompt you for various details including the name of the project to connect to, and perhaps the user name to use when you log in.

10. The new configuration is added to the list in the Defined Configurations panel.

**Note:**
All users of the shared database must specify the same SCC provider.
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.

11. When you have finished defining your version control configurations, click on the Close button.

For further information on the fields on the Version Control Settings dialog, see the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>This model is private</td>
<td>Specify whether all users connect to a single shared copy of the model (e.g. a DBMS) or each user connects to their own private copy of the model. When unselected (for shared models), the option disables the File History - Retrieve functionality when the selected package is checked out by another user. This prevents modifications that might have been made by the other user from being discarded through importing a prior revision from version control.</td>
</tr>
<tr>
<td>Save nested version controlled packages to stubs only</td>
<td>Set nested version controlled packages to stubs or fully expanded trees. Defaults to selected. For a full explanation of this option, see Version Control Nested Packages.</td>
</tr>
<tr>
<td>Unique ID</td>
<td>Specify a configuration name that readily distinguishes this configuration from other configurations. The unique ID is displayed as a selection in the list of Version Control configurations a package can connect to. You can also click on the drop-down arrow and select a previous version control configuration, providing the configuration is not in the current model.</td>
</tr>
<tr>
<td>Local Project Path</td>
<td>Specify the folder in which 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, and this should not be a shared network folder. Particularly bear this in mind if you are creating a .EAP file that is to be shared (e.g. a SQL database).</td>
</tr>
<tr>
<td>Current User</td>
<td>Read only. Shows your user name as the user currently logged into the SCC provider.</td>
</tr>
<tr>
<td>SCC Provider</td>
<td>Read only. Shows the name of the provider specified in the database.</td>
</tr>
<tr>
<td>SCC Project</td>
<td>Read only. Shows the project selected during the initial setup of the connection to the SCC provider.</td>
</tr>
</tbody>
</table>

Note:

Sparx Systems strongly urge you not to manipulate version controlled package files outside of Enterprise Architect. It is possible to leave the package files in a state that Enterprise Architect cannot recognize.

6.11.5.2.1 Upgrade at Enterprise Architect 4.5

When a version-controlled project created under a release of Enterprise Architect earlier than 4.5 is opened in Enterprise Architect release 4.5 or later, you must identify the SCC connection with a new unique ID. You can 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, you can assign a configuration quickly and efficiently using configurations that have been created previously for other version controlled repositories. This enables you to configure the many packages to use an existing version control repository; this can apply to packages created for more than just one model enabling a great deal of flexibility.

To upgrade an existing SCC version control project created before release 4.5, in Enterprise Architect release 4.5 or later, follow the steps below.

1. Open the project that has an SCC Version Control Configuration created in Enterprise Architect earlier than version 4.5.
2. The Select or Create Unique ID for Version Control dialog prompts you to create an ID for an existing configuration or to choose a previously created one from the Unique ID drop-down list.
3. The existing SCC configuration is the initial value, represented by SCC-XXXXX; this number is not
especially meaningful, therefore it is recommended that the configuration be given a meaningful name.

4. You can associate the version controlled package with a previously-defined configuration by selecting an existing configuration from the **Unique ID** drop-down list (if one exists).

5. After you have assigned the unique ID, click on the **OK** button to load the model.

### 6.11.5.3 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 Enterprise Architect, you must install version control software on your local machine. Also, you must create a working directory (using your version control software) before you can configure Enterprise Architect. You can have as many working directories as you like on your local machine.

You must also connect to a repository, which can be either a **remote repository** or **local** to your machine. If your repository is local, it must be created with your version control software.

Each working folder you create contains information on connection to a repository. This connection information includes the path to the local or remote repository, the user name and password in order to make a connection.

#### 6.11.5.3.1 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, follow the steps below:

1. Ask your system administrator to install CVS and create a remote repository with a module that you can use to control your Enterprise Architect package files. Your administrator must 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 to hold your CVS working copy directory; for example:
   ```bash
   C:\> cd myCVSWorkSpace
   ``

3. Connect to the remote CVS repository. An example connection command is:
   ```bash
   C:\myCVSWorkSpace> cvs -d:pserver:myUserID@ServerName:/repositoryFolder login
   ```
4. Create a local CVS workspace, derived from the remote repository. An example command is:
   ```
   C:\myCVSWorkSpace> cvs -d:pserver:myUserID@ServerName:/cvs checkout moduleName
   ```
   **Note:**
   The above command creates 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. Enterprise Architect uses this file to obtain your CVS user ID.

5. Verify that your CVS installation is working correctly.
6. Change directory to the one you specified as the working copy, in the `cvs checkout` command above, i.e. `C:\myCVSWorkSpace\moduleName`
7. 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
   ```
8. 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`
9. The editors command should produce output resembling the following:
   ```
   Test1.txt myUserID Tue Aug 9 10:08:43 2009 GMT myComputer C:\myCVSWorkSpace\moduleName
   ```
10. Take note of the `userID` that follows the filename. Enterprise Architect must find and use this user ID when you create your version control configuration. (See the example dialog below.)
11. Launch Enterprise Architect and open or create the model containing the packages to place under version control.
12. Select the **Project | Version Control | Version Control Settings** menu option. The **Version Control Settings** dialog displays.
13. Click on the **New** button, enter a suitable name in the **Unique ID** field, then click on the **CVS** radio button in the **Type** field.
14. To specify the Working Copy path value, click on the Select Path button. Select the local folder in which to keep local working copies of the XML files to be stored in the Version Control repository.
15. Click on the OK button to return to the Version Control Settings dialog.

16. The Current User field should display the user name used to log into the remote CVS repository. If this does not happen, it indicates that Enterprise Architect cannot extract the user name from the file .\..\WorkingCopyPath\CVS\Root and the configuration does not work correctly.

17. If necessary, set the CVS Exe Path by clicking on the Select Path... button and browsing to the filepath for the file cvs.exe, the CVS executable.

18. Click on the Save button to save the configuration you have defined. 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.

Options | Use to
--- | ---
This model is Private | Specify whether all users connect to a single shared copy of the model (e.g. a DBMS) or each user connects to their own private copy of the model.

When unselected (for shared models), the option disables the File History - Retrieve functionality when the selected package is checked out by another user. This prevents modifications that might have been made by the other user from being discarded through importing a prior revision from version control.

Save nested version controlled packages to stubs only | Set nested version controlled packages to stubs or fully expanded trees. Defaults to selected.

For a full explanation of this option, see the Version Control Nested Packages topic.

Unique ID | Specify a configuration name that readily distinguishes it from other configurations. The unique ID displays 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 | Specify 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 Enterprise Architect, 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 a .EAP file which is to be shared (e.g. a SQL database).
### Options and Use to

<table>
<thead>
<tr>
<th>Options</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current User</td>
<td>Specify the CVS user name associated with all CVS commands that are issued. This name is used by Enterprise Architect, to determine who has a package ‘checked-out’.</td>
</tr>
<tr>
<td>CVS EXE Path</td>
<td>Specify the full path of the CVS client’s executable file.</td>
</tr>
</tbody>
</table>

**Note:**

Sparx Systems strongly urge you not to manipulate version controlled package files outside of Enterprise Architect. It is possible to leave the package files in a state that Enterprise Architect cannot recognize.

### 6.11.5.3.2 CVS with Local Repositories

Before you can set up Enterprise Architect, 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 follow the steps below:

1. Launch Enterprise Architect and open or create the Enterprise Architect model for which packages are to be placed under version control.
2. Select the **Project | Version Control | Version Control Settings** menu option. The **Version Control Settings** dialog displays.
3. Click on the **New** button.
4. In the **Unique ID** field, type a suitable name for the configuration.
5. Against the **Type** field, click on the **CVS** radio button.
6. Click on the **Select Path...** button to the right of the **Working Copy path** field and browse for and select the local folder in which to keep local working copies of the XML files to be stored in the Version Control repository.

7. If necessary, click on the **Select Path...** button to the right of the **CVS Exe Path** field and browse to the file path for the file `cvs.exe`, the CVS executable.
8. Click on the **Save** button to save the configuration you have defined.
9. The new configuration is added to the list in the **Defined Configurations** panel.

**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.

For further information on the fields in the **Version Control Settings** dialog, see the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This model is Private</strong></td>
<td>Specify whether all users connect to a single shared copy of the model (e.g. a DBMS) or each user connects to their own private copy of the model. When unselected (for shared models), the option disables the <strong>File History - Retrieve</strong> functionality when the selected package is checked out by another user. This prevents modifications that might have been made by the other user from being discarded through importing a prior revision from version control.</td>
</tr>
<tr>
<td><strong>Save nested version controlled packages to stubs only</strong></td>
<td>Set nested version controlled packages to stubs or fully expanded trees. Defaults to selected. For a full explanation of this option, see <strong>Version Control Nested Packages</strong>.</td>
</tr>
<tr>
<td><strong>Unique ID</strong></td>
<td>Specify a name that readily distinguishes the configuration from other configurations. The Unique ID is displayed as a selection in the list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from the drop-down menu providing the configuration is not in use in the current model.</td>
</tr>
<tr>
<td><strong>Working Copy Plan</strong></td>
<td>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 Enterprise Architect, 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 a .EAP file which is to be shared (e.g. a SQL database).</td>
</tr>
<tr>
<td><strong>Current User</strong></td>
<td>The CVS user name associated with all CVS commands that are issued. This name is used by Enterprise Architect, to determine who has a package <code>checked-out</code>.</td>
</tr>
<tr>
<td><strong>CVS EXE Path</strong></td>
<td>The full path name of the CVS client's executable file.</td>
</tr>
</tbody>
</table>
6.11.5.4 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 Enterprise Architect version 6.0 or greater.

Tasks in setting up version control with Subversion include:

- Set up Subversion
- Create a new Repository Sub-Tree
- Create a Local Working Copy
- Subversion Under WINE-Crossover
- Version Control Configuration
- TortoiseSVN

6.11.5.4.1 Set up Subversion

Obtain and Install Subversion

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. Enterprise Architect does not work with Subversion releases earlier than Subversion 1.2.

Before Enterprise Architect can be used with Subversion, the appropriate software must be installed by a Subversion administrator. Ask your system administrator to obtain and install the Subversion server and client applications.

Enterprise Architect must use the Subversion command line client to communicate with the Subversion server; it cannot use other clients such as TortoiseSVN.

Important:

Before you attempt to use Subversion through Enterprise Architect, you must first verify that you can use the Subversion command line client to access and operate on files within the working copy folder that Enterprise Architect will use. Your environment must be set up such that you can perform these operations without ever being prompted for user ID or password. For further information, please see the topic *Caching Client Credentials* in the official Subversion documentation.

The official Subversion documentation can be found at: http://svnbook.red-bean.com/en/1.4/index.html, while executable files for Subversion can be obtained from: http://subversion.tigris.org/project_packages.html#binary-packages.

You 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 must 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 chapter.

Your administrator should set up user IDs and passwords for every person who is to 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 must create local working copies from the repository by checking-out a repository sub-tree.

Steps for setting up a repository and creating a local working copy can be found at: http://svnbook.red-bean.com/en/1.4/svn.basic.in-action.html#svn.advanced/reposurls.
Note:

Sparx Systems recommend that each new Enterprise Architect 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 using 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.

<table>
<thead>
<tr>
<th>Schema</th>
<th>Access Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>file:///</td>
<td>Direct repository access (on local disk).</td>
</tr>
<tr>
<td>https://</td>
<td>Same as http://, but with SSL encryption.</td>
</tr>
<tr>
<td>svn://</td>
<td>Access via custom protocol to an svnserve server.</td>
</tr>
<tr>
<td>svn+ssh://</td>
<td>Same as svn://, but through an SSH tunnel.</td>
</tr>
</tbody>
</table>

For more information on how Subversion parses URLs, see http://svnbook.red-bean.com/en/1.4/svn.basic.in-action.html#svn.advanced.reposurls.

See Also

- Configure Version Control with Subversion

6.11.5.4.2 Create a new Repository Sub-tree

If a repository sub-tree has already been created for your Enterprise Architect model, skip this topic and see the Create a Local Working Copy topic. If your Enterprise Architect model has not previously been added to version control, create a sub-tree for it in your SVN repository by following the steps below:

1. Create a temporary directory structure to import into the SVN repository, which initializes the repository sub-tree for this Enterprise Architect 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"

   Note:

   After the import is finished, the original tree is not converted into a working copy. To start working, you must still svn checkout a fresh working copy of the tree.

3. Delete the directory tempDir and all its contents.

6.11.5.4.3 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. Follow the steps below:

1. Choose a suitable directory on your system, in which 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 to hold your Working Copy directory and check-out the model's sub-tree from the repository, with the following command:
   
   ```
   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, you should verify everything is working correctly before you attempt to use it from within Enterprise Architect. You must be able to commit files to the repository, without being prompted for ID or passwords.

   Enterprise Architect 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 commit <fileName> -m"A meaningful comment."
   ```

   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 favorite editor:

   ```
   svn commit <fileName> -m"A meaningful comment."
   ```

6.11.5.4.4 Subversion Under WINE-Crossover

When running Enterprise Architect under WINE/CrossOver, you can use either a Windows-based Subversion client or a Linux-based Subversion client.

If you intend to use the HTTP or HTTPS protocols you must use the Unix-like client, as the Windows client cannot access the libraries necessary to create the required network connections. It is also easier to set up your working copy folder using the native Unix-like Subversion client and then continue using that same client from within Enterprise Architect.

However, to make use of the Unix-like client under CrossOver, you must also download and install a bridging utility called `SVN_gate`, which is available from the CodeWeavers’ (CrossOver) web site:


When using the Windows Subversion client, you simply install the Win32 Subversion client under CrossOver. Once you have set up your working copy directory, you are ready to use Subversion with Enterprise Architect. However, setting up your working copy is more difficult when using Win32 Subversion under CrossOver.

Because you cannot see any output from Subversion commands run under CrossOver, the best way to run the commands is to create a Windows batch file containing the command to run, and to run that batch file as a Windows command under CrossOver. (See the `example batch file` below.)

If you are running directly under WINE, launch the Windows batch file from a Unix shell script such as follows;

   ```
   /home/user/cxoffice/bin/wine --bottle "ea" --untrusted
   --workdir '/home/user/.cxoffice/ea"/drive_c"
   -- "/home/user/.cxoffice/ea/drive_c/batfile.bat"
   ```

Enterprise Architect uses the Subversion command line client to communicate with your Subversion server. In order for Enterprise Architect to work successfully with Subversion, your Subversion working copy environment must be set up such that you can issue commands to Subversion from the command line, without ever being prompted for user input such as username or password.

By default, whenever the Subversion command-line client successfully responds to a server's authentication challenge, it saves the credentials in the user's private runtime configuration area, which is:

- `~/.subversion/auth/` on Unix-like systems or
- `%APPDATA%/Subversion/auth` on Windows (which translates to `../drive_c/windows/profiles/crossover/Application Data/Subversion/auth` under CrossOver).
For this reason, Sparx Systems recommend that when you checkout a working copy from the Subversion repository, you specify your username and password on the command line, such that your credentials are cached from the outset. Use a Subversion command such as:

- Using a Unix-like client (run the command from the command line):

  ```bash
  svn checkout --username "UserName" --password "myPassword" "svn://myServerName:3690/myProject" "~/drive_c/workingCopyDirectory"
  ```

- Using a Win32 client (run the command within a batch file run under CrossOver):

  ```bash
  "C:\Program Files\Subversion\bin\svn.exe" checkout --username "UserName" --password "myPassword" "svn://myServerName:3690/myProject" "C:\SVN-test\workcopy" >"C:\SVN-test\stdout.txt" &2>"C:\SVN-test\stderr.txt"
  ```

It is a good idea to checkout your Subversion working copy into a folder within the WINE bottle where Enterprise Architect is to run. In this way, the pathnames used for your version controlled package files are much shorter.

If you intend to use the Win32 Subversion client with Enterprise Architect, you should create the local working copy that Enterprise Architect is to use, by performing a Subversion checkout command using the Win32 client under WINE/CrossOver. Similarly, if you plan to use the Unix-like Subversion client with Enterprise Architect, you should perform the initial checkout using that client. In this way, your user credentials are cached in the correct location for the client that Enterprise Architect is using.

It is important to verify that your command line client for Subversion is working correctly before attempting to connect from Enterprise Architect. For guidance on verifying your set up, see the Create a Local Working Copy topic.

The following is an example of a Windows batch file that can be used under CrossOver to run Subversion commands. Simply uncomment the command to execute. Each command should be a single line - the `\` is intended as a continuation character.

```bash
rem  "C:\Program Files\Subversion\bin\svn.exe" checkout --username "UserName" --password "myPassword" \ "svn://myServerName:3690/myProject" "C:\SVN-test\workcopy"   \ >"C:\SVN-test\stdout.txt" &2>"C:\SVN-test\stderr.txt"
rem  "C:\Program Files\Subversion\bin\svn.exe" add "C:\SVN-test\workcopy\myTestFile.xml" \ >"C:\SVN-test\stdout.txt" &2>"C:\SVN-test\stderr.txt"
rem  "C:\Program Files\Subversion\bin\svn.exe" commit -m"a message" "C:\SVN-test\workcopy\myTestFile.xml" \ >"C:\SVN-test\stdout.txt" &2>"C:\SVN-test\stderr.txt"
rem  "C:\Program Files\Subversion\bin\svn.exe" lock "C:\SVN-test\workcopy\myTestFile.xml" \ >"C:\SVN-test\stdout.txt" &2>"C:\SVN-test\stderr.txt"
```

### 6.11.5.4.5 Version Control Configuration

This topic 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 Enterprise Architect model. If this is not the case, please see the Set 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 Enterprise Architect model to place under version control.

To apply version control to your Enterprise Architect model using the Subversion working copy that you have set up, follow the steps below:

1. Launch Enterprise Architect and open the model for which this Working Copy was created.
2. Select the Project | Version Control | Version Control Settings menu option.
3. Click on the New button, enter a suitable name in the Unique ID field, then click on the Type: Subversion radio button.
4. To the right of the Working Copy path field, click on the Select Path button and select the local folder in which to keep local working copies of the XML files to be stored in the Version Control repository.

5. In the Workstation Settings panel, click on the Select Path button to specify the path for your Subversion client executable.

6. Click on the Save button to save the configuration you have defined; the new configuration is added to the Defined Configurations list.

**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.

7. When you have finished defining your version control configurations, click on the Close button.
Additional Information on the dialog fields:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This model is private</strong></td>
<td>Specify whether all users connect to a single shared copy of the model (e.g. a DBMS) or each user connects to their own private copy of the model. When unselected (for shared models), the option disables the <em>File History - Retrieve</em> functionality when the selected package is checked out by another user. This prevents modifications that might have been made by the other user from being discarded through importing a prior revision from version control.</td>
</tr>
<tr>
<td><strong>Save nested version controlled packages to stubs only</strong></td>
<td>Set nested version controlled packages to stubs or fully expanded trees. Defaults to selected. For a full explanation of this option, see the Using Nested Version Control Packages topic.</td>
</tr>
<tr>
<td><strong>Unique ID</strong></td>
<td>Specify a configuration name that readily distinguish this configuration from other configurations. The Unique ID displays as a selection in the list of Version Control configurations a package can connect to. In addition you can select a previous version control configuration from this drop-down menu, providing the configuration is not in the current model.</td>
</tr>
<tr>
<td><strong>Working Copy path</strong></td>
<td>Specify the folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every PC using Subversion version control should have its own Subversion working copy folder in which to store working copies of the XML package files; this should not be a shared network folder. Particularly bear this in mind if you are creating a .EAP file that is to be shared (e.g. a SQL database).</td>
</tr>
<tr>
<td><strong>Subversion Exe Path</strong></td>
<td>Specify the full path name of the Subversion client executable file.</td>
</tr>
</tbody>
</table>

**Note:**

Sparx Systems strongly urge you not to manipulate version controlled package files outside of Enterprise Architect. It is possible to leave the package files in a state that Enterprise Architect cannot recognize.

### 6.11.5.4.6 TortoiseSVN

TortoiseSVN is a Windows shell extension for Subversion.

Enterprise Architect cannot use TortoiseSVN to communicate with the Subversion server; it must use the Subversion command line client.

TortoiseSVN provides icon overlays in Windows Explorer that are useful as a tool for observing the status of your Subversion controlled files. It enables you to create your repository sub-trees and check out local working copies from within Windows Explorer using simple menu commands.

**Note:**

- Sparx Systems recommend that you test your local working copies, by adding and committing a dummy file from the command prompt window.
- Manipulating Enterprise Architect's package files, using tools that are external to Enterprise Architect, could leave those files in a state that Enterprise Architect cannot use.

You can download TortoiseSVN from: [http://tortoisepsvn.tigris.org/](http://tortoisepsvn.tigris.org/).

### 6.11.5.5 Version Control with TFS

In order to use Team Foundation Server for version control with Enterprise Architect, all users must have either the TFS command line client (`tf.exe`) or Microsoft's Team Foundation Server MSSCCI installed on their local machine. Each intended user must also have an account that provides read/write access to a workspace on the server.

This topic covers configuring version control using the TFS command line client. To configure version control with the TFS MSSCCI client, please follow the instructions in the [Version Control with SCC](#) topic.
Each user must 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 Enterprise Architect that uses TFS.

**Connect an Enterprise Architect Model to Version Control using TFS**

1. Open or create the Enterprise Architect model to place under version control.
2. Select the **Project | Version Control | Version Control Settings** menu option. The **Version Control Settings** dialog displays.
3. Click on the **New** button, in the **Unique ID** field enter a suitable name, then select the **TFS** radio button.

   ![Version Control Settings dialog](image)

4. Click on the **Select Path...** button to the right of the **Working Copy path** field, and select the local folder in which to keep local working copies of the XML files to be stored in the Version Control repository.
5. In the **User Name** and **Password** fields, type values that enable access to the TFS workspace associated with the Working Copy path specified above.

**Note:**

Users who automatically log in to TFS through means external to Enterprise Architect (for example, through MS Integrated Security) can leave the **User Name** and **Password** fields blank. If the **Password** field is blank, Enterprise Architect retrieves the current user's Windows username and uses that value when determining whether a package is checked out to them or to some other user.

6. The **TFS Exe Path** field displays the default installation path. Click on the **Select Path...** button if it is necessary to modify this field.

7. Click on the **Save** button to save the configuration you have defined.

8. The new configuration is added to the list in the **Defined Configurations** panel.

**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. When you have finished defining your version control configurations, click on the **Close** button.

### Additional Information on the Dialog Fields

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This model is private</strong></td>
<td>Specify whether all users connect to a single shared copy of the model (e.g. a DBMS) or each user connects to their own private copy of the model. When unselected (for shared models), the option disables the <strong>File History - Retrieve</strong> functionality when the selected package is checked out by another user. This prevents modifications that might have been made by the other user from being discarded through importing a prior revision from version control.</td>
</tr>
<tr>
<td><strong>Save nested version controlled packages to stubs only</strong></td>
<td>Set nested version controlled packages to stubs or fully expanded trees. Defaults to selected. For a full explanation of this option, see <strong>Use Nested Version Control Packages</strong></td>
</tr>
<tr>
<td><strong>Unique ID</strong></td>
<td>Specify a configuration name that readily distinguishes it from other configurations. The <strong>Unique ID</strong> is added to the 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.</td>
</tr>
<tr>
<td><strong>Working Copy Path</strong></td>
<td>Specify the folder where the XML files representing the packages are stored. This folder should already exist before it is specified. Every PC using TFS version control should have its own TFS Local Folder in which to store working copies of the XML package files - this should not be a shared network folder. Particularly bear this in mind if you are creating a .EAP file which is to be shared (e.g. a SQL database).</td>
</tr>
<tr>
<td><strong>Server Name</strong></td>
<td>Specify the name of the Team Foundation Server to connect to.</td>
</tr>
<tr>
<td><strong>Workspace Name</strong></td>
<td>Specify the name of a pre-defined TFS workspace that you are using.</td>
</tr>
<tr>
<td><strong>User Name</strong></td>
<td>Specify 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.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Password</td>
<td>Specify the password associated with the user name you specify. Enterprise Architect stores this password, in encrypted form, as part of the version control configuration data.</td>
</tr>
<tr>
<td>TFS Exe Path</td>
<td>Browse to and select the full path name of the TFS command line client's executable file.</td>
</tr>
</tbody>
</table>

**Notes:**

- Sparx Systems strongly urge you not to manipulate version controlled package files outside of Enterprise Architect. It is possible to leave the package files in a state that Enterprise Architect cannot recognize.
- Visual Studio Integration (MDG Integration for Visual Studio 2005) enhances TFS support by providing access to, for example, work items and bugs within both Enterprise Architect and the MDG Integration product.
6.11.6 Use Version Control

The following topics describe the most common activities using the version control features of Enterprise Architect, accessed through the Package Version Control Menu:

- Configure Controlled Package
- Use Existing Configuration
- Check In and Check out Packages
- Include Other Users Packages
- Apply Version Control to Branches
- Export Controlled Model Branch
- Import Controlled Model Branch
- Review Package History
- Refresh View of Shared Project

General Notes

- The export/import facility is not fast and submitting packages containing many sub-nodes to version control should be avoided. It is recommended version control is applied to individual packages. See the Version Control Nested Packages topic for more information.
- Replication should not be combined with version controlled packages.
- Sparx Systems strongly urge you not to manipulate version controlled package files outside of Enterprise Architect. It is possible to leave the package files in a state that Enterprise Architect cannot recognize.

6.11.6.1 Package Version Control Menu

To display the Version Control menu, right-click on a version controlled package in the Project Browser and select the Package Control menu option. (This menu displays a number of different options if the selected package is not under version control - see the Controlled Package Menu topic.)
### Menu Option & Function Keys

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure [Ctrl]+[Alt]+[P]</td>
<td>Display the <a href="#">Package Control Options</a> dialog which enables you to specify whether this package (and its children) is controlled, which file it is controlled through, and which version control configuration to use.</td>
</tr>
<tr>
<td>Check In Branch [Ctrl]+[Alt]+[B]</td>
<td>For the selected branch of the model, (i.e. the selected package and all of its child packages) display the <a href="#">Select Packages to Check In</a> dialog, listing all version controlled packages within that branch that are checked out to you. You can then select packages in the displayed list, to be submitted for check-in.</td>
</tr>
<tr>
<td>Check In</td>
<td>Submit the currently selected package and all sub-packages to the central repository. Enterprise Architect prompts you to enter optional comments describing changes to the packages.</td>
</tr>
<tr>
<td>Check Out</td>
<td>Retrieve 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.</td>
</tr>
<tr>
<td>Undo Check Out</td>
<td>Cancel all changes you have made to the currently selected package and sub-packages. This restores the model to the state it was in before the package was checked out, leaving the select package and sub-packages locked.</td>
</tr>
<tr>
<td>Get Latest</td>
<td>Retrieve the latest revision of the package from the repository. Available only for packages that are checked in.</td>
</tr>
<tr>
<td>Get All Latest</td>
<td>Retrieve the latest revision of the all version controlled packages in the project. Only retrieves packages that are checked in.</td>
</tr>
<tr>
<td>Put Latest</td>
<td>Update 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.</td>
</tr>
<tr>
<td>File Properties</td>
<td>Ask the version control provider to show the version control properties associated with the XML export file pertaining to the currently selected package. This also identifies who has checked out the package.</td>
</tr>
<tr>
<td>File History</td>
<td>Where the controlling package has been configured by an SCC provider, this provider shows a change history for the package. See your provider’s documentation for details on how to use the control. Otherwise if the version control is CVS the history is shown via Enterprise Architect’s internal CVS history menu.</td>
</tr>
<tr>
<td>Compare with version on disk</td>
<td>Compare the current package with the XMI version on disk.</td>
</tr>
<tr>
<td>Add Branch to Version Control</td>
<td>Apply version control to all packages within a selected model branch, in a single operation. In this context, a model branch is a package that is currently selected in the Project Browser, and all of the packages contained within it.</td>
</tr>
<tr>
<td>Export as Model Branch</td>
<td>Export a newly created model branch from your own private copy of a model.</td>
</tr>
<tr>
<td>Import a Model Branch</td>
<td>Retrieve a model branch and import it into either the source model or another model.</td>
</tr>
<tr>
<td>Get Package</td>
<td>Access packages in the version control repository that are not currently available in your model.</td>
</tr>
<tr>
<td>Version Control Settings</td>
<td>Display the <a href="#">Version Control Settings</a> dialog.</td>
</tr>
<tr>
<td>Update Package Status</td>
<td>Provide a bulk update on the status of a package, including status options such as <a href="#">Proposed</a>, <a href="#">Validate</a> and <a href="#">Mandatory</a>.</td>
</tr>
</tbody>
</table>

**Note:**
This option is a generic package option not specific to version control.
6.11.6.2 Configure Controlled Package

Before working on a package under version control, you must define it as a controlled package and specify the version control configuration to use.

Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Configure Packages permission to configure packages for version control.

Configure a Version Controlled Package

To configure a version controlled package, follow the steps below:

1. In the Project Browser, right-click on the package to place under version control. The context menu displays.
2. Click on the Package Control | Configure menu option. The Package Control Options dialog displays.

![Package Control Options Dialog]

3. Select the Control Package checkbox to indicate that this is a controlled package.
4. Click on the Version Control drop-down arrow and select the version control repository; this connects the package to a specific version control configuration. The XMI Filename field then displays the version control configuration default path.
5. The Version ID field defaults to 1.0; if necessary, change this to the appropriate reference.
6. The Owner field defaults to your user name; if necessary, type or select the name of the user who owns the package.
7. Click on the OK button to set the version control options. The Add Package to Version Control dialog displays.
8. If you do not want to check-out the package immediately, clear the Keep checked out checkbox.
9. Click on the OK button. The Add Comment window displays.

This window displays the date and time at which the package was put under version control.
10. If required, type any further comments in the window. Click on the OK button.

Enterprise Architect places the package under the version control configuration you selected, and marks the package in the Project Browser with the version controlled checked out or not checked out icons, as appropriate.

### 6.11.6.3 Use Existing Configuration

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 steps 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 Browser and select the Package Control | Version Control Settings menu option. The Version Control Settings dialog displays.
3. Click on the New button.
4. In the Unique ID field, click on the drop-down arrow and select one of the previously-defined version control configurations.
5. Click on the Save button to confirm the version control configuration.
6.11.6.4 Check In and Check Out Packages

To work on a version controlled package you must have the package checked out. When a package is checked out to a specific user, a write lock is set on the package and other users cannot make changes to it until it has been checked in again.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have Use Version Control permission to check files in and out using version control.

**Check In/Check Out**

1. In the Project Browser, right-click on the package icon.
2. Select the Package Control | Check In, Check Out or Undo Checkout menu options, as appropriate.
3. If required, enter a comment when prompted to do so.

If you are working in a private model and you select the Check Out menu option, the Import Package dialog displays (in shared models, this dialog only has default values and therefore does not display).

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force reload from XMI</td>
<td>Reload the package content from the XMI file in the central repository, even though the package and XMI file are synchronized. This ensures that links and dependencies that might not have been refreshed are updated as well.</td>
</tr>
<tr>
<td>Accept current package</td>
<td>(The default.) Leave the package content in its current state.</td>
</tr>
<tr>
<td>Refresh model view</td>
<td>Refresh the model view to show any changes from other checked out packages.</td>
</tr>
<tr>
<td>Always apply above settings</td>
<td>Apply the settings in the above three fields every time you check out a package that is found to be up to date, and therefore do not display this dialog again.</td>
</tr>
</tbody>
</table>

**Note:**

To display the dialog if, for example, you want to change the settings, press [Ctrl] while you select the Package Control | Check Out menu option.

The package icon in the Project Browser should change. When you check out a package this is represented by a figure 8 to the left of the package icon. When you check in a package the package icon is overlaid with a colored rectangle and key. In the example below, the upper package is checked out whilst the lower package is checked in.
Notes:
- If you check out a version controlled package whilst offline, the package icon has a red figure 8 in front of it. See Offline Version Control.
- If the packages under version control contain any alternative images and those images are subject to frequent change, you can set the Export alternate images option on the Options dialog to export the images to the version control repository when you check in the packages. If the images are not subject to frequent change, do not select this option and instead use Export/Import Reference Data to manage alternative images.

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 and select the Package Control | Check In Branch context menu option. The Select Packages to Check-in dialog displays, listing all version controlled packages within the branch that you have checked out.

2. Click on the package to check in, or use:
   - [Ctrl]+click to add or remove several individual packages
   - [Shift]+click to select a range of packages
   - All to select all packages listed
   - None to clear all selected packages.
3. Click on the OK button to check-in the selected packages.
4. If required, enter a comment when prompted to do so. (This comment applies to all packages that you have checked in.)
5. Each package icon changes to indicate that the packages have been checked-in.

6.11.6.5 Include Other Users’ Packages
You can retrieve packages that have been created by other users, or by you in another model, from version control and import them into your current model.

Other users might be creating packages to use in your model. If you are not sharing a SQL database or .EAP file, those packages do not automatically become part of your model. If the packages have been placed into
version control, you can retrieve them and import them into your model as children of an existing package using the Get Package command.

1. 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.

2. In the Project Browser, right-click on the package to use as the parent of the incoming package.


4. In the Select a Version Control Configuration field, click on the drop-down arrow and select the version control configuration associated with the package to retrieve. The file list is populated with the names of files available through that configuration, for retrieval and import into your model.

5. Select the package file to import into your model and click on the OK button.

6. **6.11.6.6 Apply Version Control To Branches**

Enterprise Architect enables you to apply version control to all packages within a selected model branch, in a single operation. In this context, a model branch is a package that is currently selected in the Project Browser, and all of the packages contained within it.

The Version Control Configuration to be used in this operation must be defined within the model before selecting this command.

To apply version control to a model branch, follow the steps below:

1. Right-click on the required package and select the Add Branch to Version Control context menu option. The Apply VC to Branch dialog displays.
2. In the **Select a Version Control Configuration** field, click on the drop-down arrow and select the Version Control Configuration to use.

3. If required, select the **Export as Model Branch** checkbox to **export the selected package (and sub-packages) as a Model Branch**.

4. Click on the **OK** button. Enterprise Architect creates a number of sub-folders within the version control working copy folder, before exporting all of the packages within the selected model branch. Enterprise Architect generates package filenames using the package GUIDs, before adding the resulting files to version control.

If you have selected the **Export as Model Branch** checkbox, once the version control operation is complete Enterprise Architect also creates a Model Branch file (.EAB file). You can subsequently **import the version-controlled Model Branch**.

### 6.11.6.7 Export Controlled Model Branch

You might want to export a newly created model branch from your own private copy of a model so that, for example:

- Another user can import that branch into their own private copy of the same model, or
- It can be imported for inclusion as a common branch in a number of different models.

Applying version control to an Enterprise Architect model can result in many XML files placed under version control. It could then be hard to locate and import the file corresponding to the root of a particular model branch. Enterprise Architect's Model Branch Files (.EAB files) overcome this problem by simplifying the retrieval of model hierarchies for use in other models.

The facility is only enabled for packages that are already under version control. The exported Model Branch File is also placed under version control, using the same version control configuration that is controlling the selected package.

To export a model branch, follow the steps below:

1. Right-click on the version controlled package to export as a model branch.

2. Select the **Package Control | Export as Model Branch** context menu option. The **Export as Model Branch** dialog displays.

3. In the **EAB Filename** field, type a name for your Model Branch File. Alternatively, click on [ ... ] and browse for the file location.

   Note that the package name is supplied as a default.
You can specify any file name, including sub-folder names, as long as the file is contained in or below the working folder of your version control configuration.

6.11.6.8 Import Controlled Model Branch

It might be necessary to either:

- Retrieve a model branch created by another user in a private copy of a model, to import it into your own private copy of the same model or
- Retrieve a model branch that is common in many models, for inclusion in a new model.

Applying version control to an Enterprise Architect model can result in many XMI files placed under version control. It could then be hard to locate and import the file corresponding to the root of a particular model branch. Enterprise Architect's Model Branch files overcome this problem by simplifying the retrieval of model hierarchies for use in other models.

The Import a Model Branch context menu option uses Enterprise Architect's Model Branch Files, of which there are few, to retrieve information about the root package file and import the model branch. The Model Branch File records information such as the name and type of the version control configuration for the selected package, and the relative filename of the version controlled XMI file associated with the package.

Before attempting to import a model branch, you must have access to the version controlled XMI files that represent the model branch to be imported. That is, there must be a working copy folder, accessible from the machine on which Enterprise Architect is running, that is associated with the Version Control repository containing those XMI files.

It is not necessary to have the relevant Version Control Configuration set up within Enterprise Architect before issuing this command - Enterprise Architect prompts you to complete specification of the configuration if necessary.

The Import a Model Branch context menu option is only enabled for packages that you (the current user) are able to edit, as the imported model branch is inserted into the model under the selected package.

To import a model branch, follow the steps below:

1. Right-click on the package into which the model branch is to be imported.
2. Select the Package Control | Import a Model Branch context menu option. The Import VC Model Branch dialog displays.
3. Either:
   - Click on the **Find a Model Branch (.EAB) file** button and browse for the Model Branch File. If the version control configuration used by the file has not been fully set up, Enterprise Architect prompts you to complete and save the configuration. The model branch import then proceeds. OR
   - If the version control configuration used by the file has been fully set up in the current model, click on the drop-down arrow in the **Select a Version Control Configuration** field and select the configuration, then select the Model Branch File from the **Select a Model Branch (.EAB) file** list. Click on the **OK** button to import the model branch.

Enterprise Architect imports the root package specified in the Model Branch File and recursively imports and populates all the sub-packages contained in the root package.

### 6.11.6.9 Review Package History

Reviewing package history enables you to view the history of checked in package revisions. It also enables you to import selected prior revisions of the package into your model. The package revision is retrieved in read-only format, enabling you to view the package contents but not make any changes to the package.

To review package history follow the steps below:

1. In the **Project Browser**, right-click on the package configured for version control. The context menu displays.
2. Select the **Package Control | File History** menu option.
   - If the package has been configured through SCC, you access the history through the mechanism offered by the third party SCC provider.
   - If the package has been configured for CVS, Subversion or TFS, Enterprise Architect's **File Version History** dialog displays and the following steps apply.
3. In the **Revisions** field, click on a revision number to view the log entries for that revision.

4. To view the package history select a revision and then click on the **Retrieve** button. A warning dialog displays, indicating that the package is being opened in read-only mode.

5. Click on the **Yes** button to continue or the **No** button to cancel the action. The package is retrieved in read only mode, enabling you to view the history of the package at the specified version.

6. To go back to the latest version, in the **Project Browser** right-click on the package and select either the **Package Control | Check Out..** menu option or the **Package Control | Get Latest** menu option.

### 6.11.6.10 Refresh View of Shared Project

When a user of a shared model checks out a package and makes changes, other users can see those changes by refreshing their view of the package or the changed diagram within the package.

You can refresh your view of the **Project Browser** in the following ways:

- Right-click on the package name in the **Project Browser** and select the **Contents | Reload Current Package** context menu option

- Select the **File | Reload Current Project** menu option (or select the **Reload Project** icon in the **Project Toolbar**, or press **[Ctrl]+[Shift]+[F11]**)

- Close the project and reopen it.

You can refresh the current diagram in the following ways:

- Select the **Window | Reload Current View** menu option

- Right-click on the opened **diagram tab** in the diagram view, and select the **Reload <diagram name>** context menu option.
6.11.7 Offline Version Control

When loading a model that uses version control, Enterprise Architect normally initializes a connection to the version control system for each Version Control Configuration defined in the model. If Enterprise Architect is unable to connect a Version Control Configuration for any reason, it displays warning messages to notify you and provides 'offline' version control functionality for all packages associated with the failed connection.

You can prevent Enterprise Architect from attempting to make any version control connections by selecting the Project | Version Control | Work Offline menu option before loading a model. This is useful if you know that Enterprise Architect cannot connect to your version control system. For example, if you are working on a laptop computer that is disconnected from your network and you have an Enterprise Architect model that uses a large number of Version Control Configurations, choosing to work offline before you load the model enables you to avoid all the error messages that Enterprise Architect would normally display as each version control connection attempt fails.

You can switch between working offline and working online at any time, either before or after a model is loaded. Toggle the Project | Version Control | Work Offline menu option. Enterprise Architect disconnects or reconnects version control (depending on connection availability) according to your selection.

Use Version Control Whilst Disconnected From Your Version Control Server

Enterprise Architect 'remembers' the status of a model’s version controlled packages. Packages that were checked out to you prior to disconnecting from the server are still shown as checked out to you, even though you are no longer connected to the server. You can still edit these packages as you normally would.

Packages that were not checked out to you prior to disconnecting from the server are shown as version controlled and locked. You cannot edit these packages until you check them out.

Offline Check Out

In releases of Enterprise Architect from release 6.0 onwards, you can 'check-out' and edit a version controlled package even when your machine is disconnected from the version control server. In the example below, the colored figure 8 icon for Pkg2 indicates that you have checked it out whilst offline (the gray 'figure 8' icon shown against Pkg1 indicates that you have checked out a version-controlled package online).

Important:
You should be aware that the version control system - and therefore other users - 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. If an offline checkout leads to two people editing the same package at the same time, when the changes are brought back online the first-saved set of changes is lost.

Check 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, you can check in that package. However, before Enterprise Architect checks in such a package, it compares the local working copy of the package file with the latest revision in the repository. (These package files remain unchanged in your work area until Enterprise Architect exports the package again before checking in.) If the repository version remains unchanged from when you last updated your local copy, Enterprise Architect exports and checks in your package without further prompting.
On the other hand, if the repository now contains a file that has changed since you last updated your local copy, checking in your package overwrites whatever those changes might be. Enterprise Architect displays a message warning you of the pending data loss and giving you the opportunity to abort the check in. At this point, you must decide whether to discard your own changes, using the **Undo Check Out** command, or continue with your check in and overwrite the 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 might help you to discover what changes have been uploaded and decide whose changes take precedence.

**Update Before You Disconnect**

Whenever you are connected to the version control server, you are always working with the latest version of a 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 are working on whatever versions you have on your machine, dating back to the last time you updated your local copy of each version controlled package. 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** option makes this a simple task.
6.12 User Security

What is User Security in Enterprise Architect?
User security in Enterprise Architect can be used to limit the access to update functions within the model. Elements can be locked per user or per group. Where user security is enabled a password is required to log in to the model. Security in Enterprise Architect is not designed to prevent unauthorized access; rather it is intended as a means of improving 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 is available only in the Corporate edition of Enterprise Architect. User security in Enterprise Architect offers two policies, the standard security model and the rigorous security model. In the standard security model all elements are considered unlocked and, as necessary, a user can 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 Enterprise Architect model is read-only until a user applies an editing lock on one or more elements. For more detailed information on the security policies see the Security Policy topic.

User Security Tasks
A number of security tasks can only be performed by users with Administrative rights to the model. These tasks include:

- Security Policy
- Enable Security
- Maintain Users
- Change User Passwords
- Assign User To Groups
- View All User Permissions
- Maintain Groups
- View and Manage Locks
- Password Encryption (for the third-party DBMS connection password; only available for Oracle and SQL Server Repositories for Enterprise Architect releases prior to 7.1).

Other Security tasks can be performed by users who do not have Administrative rights. These tasks include:

- Lock Model Elements
- Lock Packages
- Apply a User Lock
- Identify Who Has Locked An Object
- Locked Element Indicators
- Manage Your Own Locks
- Change Your Own Password

Notes:
- User security is not enabled by default in Enterprise Architect; you must enable it first.
- For a number of operations in Enterprise Architect, if security is enabled a user must have the appropriate user or group access permission to perform the operation. However, if security is not enabled, the user does not have to have access permissions. See the List of Available Permissions topic.
6.12.1 Enable Security

User security is not enabled by default in Enterprise Architect. To enable security for a project in Enterprise Architect for the first time, follow the steps below.

1. Access the Registered Users section of the Sparx Systems website (http://www.sparxsystems.com/registered/reg_ea_corp_ed.html), and obtain the Authorization Key. (You must have the Registered Users login and password to access this web site.)

2. Select the Project | Security | Enable Security menu option.

   The Authorization dialog displays.

3. In the Enter authorization key field, type the authorization key from the Sparx Systems website.

4. Click on the OK button. Security is enabled, and an Admin user and user group are created with full permissions (all access rights listed in List of Available Permissions) and a password of password.

5. Select the Project | Security | Login as Another User menu option, and log in as Admin with the initial password of password.

   Note: To change the Admin password, see the Change Password topic.

6. Set up users and permissions as required.

   Note: Once security has been enabled, you must have the Security - Enable/Disable access right to turn it off. The initial administrator automatically has this access right.

7. To disable security, click on the Enable Security menu option, and again type the authorization key in the Authorization dialog. Click on the OK button. Security is disabled.

   Notes:
   - The system prompts you to log off the project and log on again, but this is not strictly necessary.
   - To re-enable security, follow the procedure above, but be aware that any changes you have made to the admin user (password and reduced access rights) are reset to password and full access.
6.12.2 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 necessary, you can 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 Enterprise Architect model is read-only until you apply 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 requirement to ensure only one person can edit a resource at one time. This is suitable for much larger projects where there might be less communication between users.

Toggle between these modes using the Project | Security | Require User Lock to Edit menu option:

- Manage Users...
- Manage Groups...
- Manage Locks...
- Change Password...
- Login as Another User...
- My Locks... Ctrl+Shift+L
- Enable Security...
- Require User Lock to Edit
- Encrypt Password...

Notes:

- When you add new elements in Mode 1 (elements editable by default), no user lock is created automatically for the newly created element.
- When you add new elements in Mode 2 (elements locked by default), a user lock is created on the new element to enable instant editing.
6.12.3 Maintain Users

If you enable security you have access to the Security Users dialog, which you can use to set up more users for your model.

**Note:**

You must have **Security - Manage Users** permission to maintain users, and **Change Password** permission to change the password of the current user; the initial **Admin** administrator automatically has these permissions.

### Set Up a User

To set up a user for your model, follow the steps below:


   ![Security Users Dialog](image)

2. 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** or **set up Single Permissions** or **View All** permissions for the currently selected user.

3. To set the user's password, click on the **Change Password** button. The Change Password dialog displays.

   ![Change Password Dialog](image)

4. In the **New password** field, type the user's password.

5. In the **Retype new** field, type the user's password again, for confirmation.

6. Click on the **OK** button.
7. A 'Password Changed' message displays. Click on the OK button.

**Notes:**
- If you select the **Accept Windows Authentication** checkbox, on opening the model Enterprise Architect checks the users database for your Windows ID and, if it matches automatically, logs you in without prompting for a password. The Enterprise Architect model administrator must ensure that all Windows IDs 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>`.
- You can transport these user definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
6.12.4 Assign User To Groups

To set up user groups follow the steps below:

1. Select the **Project | Security | Manage Users** menu option. The **Security Users** dialog displays.
2. Click on the **Group Membership** button. The **User Groups** dialog displays.
3. Select the checkbox against each group this user belongs to.
4. Click on the **OK** button to assign the user to each group.

**Notes:**

- To create new user groups, see the **Maintain Groups** topic.
- You can transport these user groups between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu.
6.12.5 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.

**Note:**

You must have **Security - Manage Users** permission to assign permissions to users; the initial Admin administrator automatically has this permission.

To set up single permissions for a user follow the steps below:

2. Click on the **Single Permissions** button. The User Permissions dialog displays.

3. Select the checkbox against each specific permission to apply to this user. Click on the **Select All** button to select all permissions for the user, or click on the **Deselect All** button to clear all selected permissions.

4. Click on the **OK** button to assign the selected permissions to the user.

**Notes:**

- A user's total permissions are those granted by Group Membership plus those granted by specific permission assignment.
- You can transport these user permissions between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu.
6.12.6 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. To display the dialog, select the Project | Security | Manage Users menu option, then select the required user and click on the View All button.

Assigned permissions:
- Configure Stereotypes
- Configure Version Control
- Export XMI
- Generate Documents
- Generate Source Code and DDL
- Import XMI
- Lock Elements
- Manage Diagrams
- Manage Issues
- Manage Project Information
- Manage Reference Data - Update
- Manage Replicas
- Manage Tests
- Reverse Engineer from DDL and Source Code

[Image of the dialog with checkboxes for different permissions]
6.12.7 Maintain Groups

Security groups make it easy to configure sets of permissions and apply them to a number of users in one action.

Notes:

- You must have Security - Manage Users permission to manage user groups; the initial Admin administrator automatically has this permission.
- You do not define groups as group logins with passwords. If you intend to use a group login, you can define a single-user login and password that all group members use (that is, Enterprise Architect allows multiple logins under one user ID).

Set Up a Security Group

To set up a security group, follow the steps below:


   ![Security Groups Dialog](image)

   - **Group Name**: BusinessDept
   - **Description**: Business department users

2. In the Group Name and Description fields, type the security group name and a description of the group.

3. Click on the Save button.

Note:

You can transport these security group definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
6.12.8 Set Group Permissions

Note:
You must have Security - Manage Users permission to assign permissions to user groups; the initial Admin administrator automatically has this permission.

To set up permissions to apply to a security group, follow the steps below:

2. Click on the Set Group Permissions button. The Group Permissions dialog displays.

3. Select the checkbox against each required permission. Click on the Select All button to select all permissions for the user, or click on the Deselect All button to clear all selected permissions.
4. Click on the OK button to assign the permissions. All of the users assigned to this group share in this set of permissions.

Note:
You can transport these group permission definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
### 6.12.9 List of Available Permissions

The following table lists the available permissions in the Corporate edition of Enterprise Architect. These permissions are required for the corresponding operations if security is enabled.

**Note:** Some permissions take precedence over others. For example, if you set Use Version Control permission for a user, that user can modify model elements on import even if they do not have Update Element permission.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Enables the user to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administer Database</td>
<td>Compact and repair project database.</td>
</tr>
<tr>
<td>Audit Settings</td>
<td>Change the audit settings in the Audit Settings dialog.</td>
</tr>
<tr>
<td>Audit View</td>
<td>Enable auditing and display data in the Audit View and Audit History tab.</td>
</tr>
<tr>
<td>Baselines - Manage</td>
<td>Create, delete, import and export Baselines</td>
</tr>
<tr>
<td>Baselines - Restore</td>
<td>Merge data into the project model from a Baseline or XML file.</td>
</tr>
<tr>
<td>Change Password</td>
<td>Change your own password or (Administrator) another user's password.</td>
</tr>
<tr>
<td>Check Data Integrity</td>
<td>Check and repair project integrity</td>
</tr>
<tr>
<td>Configure Datatypes</td>
<td>Add, modify and delete datatypes</td>
</tr>
<tr>
<td>Configure Images</td>
<td>Configure alternative element images</td>
</tr>
<tr>
<td>Configure Packages</td>
<td>Configure controlled packages and package properties.</td>
</tr>
<tr>
<td>Configure Resources</td>
<td>Create and manage Resources and items: RTF templates, patterns, profiles, favorites.</td>
</tr>
<tr>
<td>Configure Stereotypes</td>
<td>Add, modify and delete Stereotypes</td>
</tr>
<tr>
<td>Configure Version Control</td>
<td>Set up version control options for the current model.</td>
</tr>
<tr>
<td>Export XMI</td>
<td>Export model to XMI.</td>
</tr>
<tr>
<td>Generate Documents</td>
<td>Generate RTF and HTML documents from model packages.</td>
</tr>
<tr>
<td>Generate Source Code and DDL</td>
<td>Generate source code and DDL from model element. Synchronize code against model elements if it already exists.</td>
</tr>
<tr>
<td>Import XMI</td>
<td>Import model from XMI.</td>
</tr>
<tr>
<td>Lock Objects</td>
<td>Lock an element or package</td>
</tr>
<tr>
<td>Manage Diagrams</td>
<td>Create new diagrams, copy existing and delete diagrams. Also save diagram as UML Pattern.</td>
</tr>
<tr>
<td>Manage Issues</td>
<td>Update and delete Issues</td>
</tr>
<tr>
<td>Manage Project Information</td>
<td>Update and manage resources metrics, risks.</td>
</tr>
<tr>
<td>Manage Reference Data - Update</td>
<td>Update and delete reference items</td>
</tr>
<tr>
<td>Manage Replicas</td>
<td>Create and synchronize replicas.</td>
</tr>
<tr>
<td>Manage Tests</td>
<td>Update and delete Test records</td>
</tr>
<tr>
<td>Reverse Engineer from DDL and Source Code</td>
<td>Reverse engineer from source code or ODBC, and synchronize model elements against code.</td>
</tr>
<tr>
<td>Security - Manage Locks</td>
<td>View and delete element locks.</td>
</tr>
<tr>
<td>Security - Manage Users</td>
<td>Maintain users, groups and assigned permissions</td>
</tr>
<tr>
<td>Spell Check</td>
<td>Spell check package and set spell check language.</td>
</tr>
</tbody>
</table>

© 1998-2008 Sparx Systems Pty Ltd
<table>
<thead>
<tr>
<th>Permission</th>
<th>Enables the user to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer Data</td>
<td>Transfer model between different repositories.</td>
</tr>
<tr>
<td>Transform Package</td>
<td>Perform transformations of packages and elements.</td>
</tr>
<tr>
<td>Update Diagrams</td>
<td>Update diagram appearance, properties and layout, including the Page Setup dialog.</td>
</tr>
<tr>
<td>Update Element</td>
<td>Save model changes (including delete) for elements, packages, and relationships.</td>
</tr>
<tr>
<td>Use Version Control</td>
<td>Check files in and out using version control.</td>
</tr>
</tbody>
</table>
6.12.10 View and Manage Locks

From time to time it might be necessary to examine or delete locks placed on elements by users. Enterprise Architect provides a function to view and manage active locks.

Notes:

- You must have Security - Manage Locks permission to view and delete user locks; the initial Admin administrator automatically has this permission.
- If an element is locked, connectors attached to it are also locked. To unlock the connector, you must unlock the element. However, under certain circumstances you can add new connectors to a locked element.

Delete a Lock

To view locks and, if necessary, delete them, follow the steps below:

1. Select the Project | Security | Manage Locks menu option. The Active Locks dialog displays.

2. In the View Locks For panel, click on the radio button for the type of lock to view: All, Groups Only or Users Only. Locks of the appropriate type are listed in the Active Locks panel.

3. To remove a lock, click on it and click on the Unlock Selected button.

4. When finished, click on the Close button to close the dialog.
6.12.11 Password Encryption

**Note:**

This topic is retained to support regression to releases of Enterprise Architect prior to version 7.1. For password encryption for all repositories at and beyond release 7.1, see the [Save Model Copy or Shortcut](#) topic.

Users of SQL Server or Oracle repositories have the option of encrypting the password used to set up the connection between Enterprise Architect and the repository. The Enterprise Architect user does not have the real password, thereby preventing them from accessing the repository using other tools such as Query Analyzer or SQLPlus.

Once security is enabled, the administrator must log on to access the dialog to create encrypted passwords. To encrypt a password, follow the steps below:

1. Select the **Project | Security | Encrypt Password** menu option. The following dialog displays:

   ![Password Encryption Dialog](image)

   - **Password:** `password123`
   - **Encrypted:** `qcvtyrsf435`

2. In the example above, the password `password123` is used to access the repository.

3. To connect Enterprise Architect to the repository, the user enters the encrypted password prefixed with `$$, so the encrypted password becomes `$qcvtyrsf435$`.

   For more information relating to connecting to Oracle and SQL Server, see the [Connect to Oracle Data Repository](#) and [Connect to SQL Server Data Repository](#) topics respectively.

**Notes:**

- Do not use the **Test Connection** button as it can cause an error with encrypted passwords.
- For SQL Server repositories, you must enter the **Initial Catalog** details from the All tab of the Data Link Properties dialog.
These are the initialization properties for this type of data. To edit a value, select a property, then choose Edit Value below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect Timeout</td>
<td></td>
</tr>
<tr>
<td>Data Source</td>
<td></td>
</tr>
<tr>
<td>Extended Props</td>
<td></td>
</tr>
<tr>
<td>General Timeout</td>
<td></td>
</tr>
<tr>
<td>Initial Catalog</td>
<td>EA_Repository</td>
</tr>
<tr>
<td>Locale Identifier</td>
<td>3081</td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>Persit Security Info</td>
<td></td>
</tr>
<tr>
<td>User ID</td>
<td></td>
</tr>
</tbody>
</table>

[Edit Value...]
### 6.12.12 Change Password

There are two ways in which a user's password can be changed, when security is set:

- A user can select the **Change Password** menu option and change their own password
- The Administrator can set or change any user's password, on the **Maintain Users** dialog.

**Note:**
A user must have **Change Password** permission to change a password; the initial Admin administrator automatically has this permission.

#### User Change

If security is set and you want to change your own password, follow the steps below:

1. Select the **Project | Security | Change Password** menu option. The **Change Password** dialog displays.

2. In the **Enter old password** field, type your current password.
3. In the **New password** field, type your new password.
4. In the **Retype new** field, type your new password again, for confirmation.
5. Click on the **OK** button.
6. A 'Password Changed' message displays. Click on the **OK** button to clear the message.

Your new password is effective next time you log in.

#### Administrator Change

To set or change any user's password, follow the steps below:

1. Select the **Project | Security | Manage Users** menu option. The **Security Users** dialog displays.
2. Click on the user name in the **Users** panel, to display the user details in the dialog fields.

3. Click on the **Change Password** button. The **Change Password** dialog displays.

4. In the **New password** field, type the user's password.

   **Note:**
   
   You do not have to enter the user's current password, as they might have forgotten it and therefore it is possible that nobody can provide that value.

5. In the **Retype new** field, type the user's password again, for confirmation.

6. Click on the **OK** button.

7. A 'Password Changed' message displays. Click on the **OK** button.
6.12.13 Lock Model Elements

**Note:**
When security is enabled, you must have Lock Objects permission to lock an element.

You can lock a package, element or diagram using the corresponding Lock context menu option in the Project Browser, and you can lock an element or diagram using the corresponding Lock context menu option in the diagram.

When you select the Lock option, the Element Lock dialog displays:

The four lock options available are:

- **No lock** - do not lock this element; clear any existing lock
- **Full lock** - lock this element so that no-one can edit it
- **User lock** - lock this element so that only the locking user can make further edits
- **Group lock** - lock this element so that any member of the specified group (in the GroupID field) can update the element, but others are excluded.

Select the appropriate lock and click on the OK button.

If the item is already locked, only the appropriate lock option and No lock are available. You have to release the lock in order to set a different type of lock.
6.12.14 Add Connectors To 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 can be added
- **Source unlocked, target locked**: allowed, except for composition connectors
- **Source locked, target unlocked**: prohibited, except for composition connectors
- **Source locked, target locked**: prohibited for all connectors.

That is, 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).

Connectors with locked source or target elements are also locked. To unlock the connector, you must unlock the source and/or target element.
6.12.15 Lock Packages

Note:
If security is enabled you must have Lock Objects permission to lock a package.

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, follow the steps below:

1. Deselect the Project | Security | Require User Lock to Edit menu option.
2. In the Project Browser, right-click on the package to lock. The context menu displays.
3. Select the Lock Package menu option. The Lock/Unlock Package(s) dialog displays.

   ![Lock Type](image)

   - No lock, general editing allowed
   - Full lock, no one may edit
   - User lock, locking user may still edit
   - Group lock, locking group may still edit

   GroupID: Administrators

   ![What to Process](image)

   - Lock Elements
   - Lock Diagram
   - Process Child Packages

4. In the Lock Type panel, select the appropriate radio button for the lock to apply.
5. As required, select the checkboxes to lock elements and/or diagrams, and to process child packages (i.e. lock the whole branch).
6. Click on the OK button to apply the lock.
6.12.16 Apply a User Lock

In the Require User Lock to Edit security mode, where a User Lock is required before any edit can occur, you can set or release the lock in either a diagram or the Project Browser. Enterprise Architect adjusts the lock for the element, or for the diagram and any elements contained in the diagram.

In a diagram, you right-click on the element or diagram; in the Project Browser, you right-click on the package, diagram or element. In each case, select the Apply/Release User Lock context menu option for the selected item. The following dialog displays.

Select the appropriate radio button to apply or release a user lock on the selected item.

**Note:**
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, a list of elements that couldn't be locked displays at the end of the process.
6.12.17 Locked Element Indicators

When an item is locked through Project Security, the lock is indicated in the Project Browser by a marker against the item, as shown below.

The meaning of the marker depends on the security mode.

If you are using the Require User Lock to Edit security mode:

- **No marker** - there is no lock, the item is not editable, but any user can now apply a user lock to edit the item.
- **Blue exclamation mark** - the current user has applied a user lock and can edit the item; no other user can edit the item.
- **Red exclamation mark** - another user has applied a user lock, and the current user cannot edit the item.

If another user has locked an item, you can identify who has locked it.

If you are using the standard security mode:

- **No marker** - there is no lock, the item is editable, but any user can now apply a user or group lock.
- **Blue exclamation mark** - the item has a lock set by the current user or a group having the current user as a member, and the user can edit the item.
- **Red exclamation mark** - the item has a lock set by another user, or a group of which the current user is not a member; the current user cannot edit the item.

If another user has locked an item, you can identify who has locked it.
6.12.18 Identify Who Has Locked An Object

If you find that a diagram, package or element is locked, you can find out which group or user currently holds the lock on that item. To do this, follow the steps below:

1. In the Project Browser, right-click on the diagram, package or element that is locked by another user or user group. The context menu displays.
2. Select the Lock... menu option.

A message box displays showing which group or user currently holds the lock on that item.
6.12.19 Manage Your Own Locks

You can view and delete your own user-level locks in Enterprise Architect. This is especially useful when working in Mode 2 security (user locks required to edit).

To manage your locks select the Project | Security | My Locks menu option. The My Locks dialog displays.

In the My Locks dialog you can select one or more locks and delete them (i.e. unlock the object) by clicking on the Unlock Selected button.
6.13 Auditing

Auditing is a project-level feature, available in the Corporate Edition, that enables model administrators to record model changes in Enterprise Architect. By enabling this option, you can view information on changes such as:

- Who changed an element
- How many elements they changed
- When they changed the data
- What the previous values were, and
- What type of elements they changed.

Auditing does not record changes to RTF Templates, Model documents, Baselines or Profiles.

**Warning:**
If your site runs separate editions of Enterprise Architect, when Auditing is turned on in a project any Desktop or Professional edition users are locked out of the project. To restore access, turn Auditing off in the project from a Corporate edition instance of Enterprise Architect.

**Auditing Quickstart**
To quickly enable auditing and see it in action, see Auditing Quickstart.

**Audit Settings**
Once auditing is enabled within a project, you have a variety of options available for customizing what is recorded by the audit. See Audit Settings for more information on the different settings available.

**The Audit View**
To view what has been recorded by the audit, use the Audit View, which provides an interface to everything recorded by auditing.

**Note:**
If security is enabled, you must have Audit View permission to display data in the Audit View.

**RTF Report**
You can generate an RTF report that includes the audit history information for the selected element or package, by choosing the basic + audit RTF template.

**Audit History**
Using Auditing, you can track changes to an element or connector over time. However, enabling Auditing also enables an Audit History tab in the Output window, which summarizes all changes made to the selected element or connector.

**Performance Issues**
By enabling auditing on a project, you increase the time taken for most actions. For most modeling tasks, this increase is insignificant. However, there are some instances where the difference is more substantial. See Performance Issues for more information.
6.13.1 Auditing Quickstart

To quickly enable auditing and see it in action, follow the instructions below.

Enable Auditing

To enable Auditing:
1. Select the **View** | **Audit View** menu option to open the **Audit View** dialog.
2. Click on the **Audit Settings** button. The **Audit Settings** dialog displays.
3. Select the **Enable Auditing** checkbox.
4. Click on the **OK** button to close the **Audit Settings** dialog.
5. Close the **Audit View** dialog.

You can also open the **Audit View** by clicking on the **Audit View** button in the **Other Views** toolbar.

Make Changes

Change and save your project; for example:
- Add a new package
- Add a new Class
- Add a new connector
- Change the name of an element
- Delete an element.

View Changes in the Audit View

Open the **Audit View** again (**View** | **Audit View**), and click on the **Refresh** (or **Load**) button to display a record of the changes you made.
6.13.2 Auditing Settings

The Audit Settings dialog enables you to change what is recorded by the Auditing facility.

**Note:**
If security is enabled, you must have Audit View permission to turn Auditing on, and Audit Settings permission to change Audit settings.

To open the Audit Settings dialog:
1. Select the View | Audit View menu option to open the Audit View.
2. Click on the Audit Settings button. The Audit Settings dialog displays.

![Audit Settings Dialog](image)

The settings on this dialog are described in the following topics:
- Audit Scope
- Audit Logs
- Auditing Level
- Audit Options

6.13.2.1 Audit Scope

The Audit Settings dialog provides checkbox options for turning Auditing on, and for including or excluding areas of processing in Enterprise Architecture.

- **Enable Auditing** - select this checkbox to turn the Auditing facility on
- **Audit XMI Import** - select this checkbox to include XMI importing in the audit
- **Audit XMI Export** - select this checkbox to include XMI exporting in the audit
- **Audit Reverse Engineering** - select this checkbox to include reverse engineering in the audit

**Note:**
As version control uses XMI, these options must be selected to record changes from checking out packages.

- **Audit Reverse Engineering** - select this checkbox to include reverse engineering in the audit
- **Use Database Timestamp** - select this checkbox to use the database server's timestamp instead of each user's local timestamp; this improves security.

**Note:**
The Use Database Timestamp option is not available for projects stored in .EAP Files.
6.13.2.2 Audit Logs

The Audit Settings dialog enables you to administer your audit records. As the number of records increases, the performance of the Audit View reduces. It is recommended that audit records that are not regularly required are saved to file, then cleared from the project. This helps ensure high performance.

- **Clear Logs** - removes all log data from the current project; all data is permanently deleted. To keep the audit records outside the project, click on the **Save Logs** button to save the records before clearing them from the project.
- **Save Logs** - saves a copy of the log items currently held in the database; these items remain in the database. To remove the items after saving the copy, click on the **Clear Logs** button.
- **Load Logs** - enables you to load a previously saved set of logs back into the project. The file is not deleted by this operation. If duplicate logs exist in both the project and the file, these are skipped.

**Note:**

Some of these functions can be accessed through the Automation Interface. For more information, see the Repository topic in SDK for Enterprise Architect.

If you save or clear the log items, Enterprise Architect prompts you to specify whether to save or clear the items covering a specific period of time.

- If you click on the **No** button, you save or clear all log items currently held in the database.
- If you click on the **Yes** button, the Time Filter dialog displays, on which you select a standard time period or define your own.

6.13.2.3 Auditing Level

The Auditing Level panel provides three options that determine what kind of model changes are recorded.

- **Core** - select this radio button to record changes to elements (including attributes and operations), packages, connectors and some model-level information.
- **Standard** - select this radio button to record the same changes as the **Core** option, plus changes to diagrams.
- **Extended** - select this radio button to record the same changes as the **Standard** option, plus changes to security.

6.13.2.4 Audit Options

The following elements are always audited:

- Packages
- Notes
- Boundary
- Text
- Diagrams (if on **Standard** level)
- Security (if on **Extended** level)
The audit options enable you to configure auditing to record changes to only certain types of elements.

- **Maintenance** - select this radio button to audit maintenance elements; that is:
  - Package
  - Requirement
  - Feature
  - Use Case
  - Actor
  - Note
  - Issue
  - Change

- **Core Structural** - select this radio button to audit maintenance elements plus some structural elements; that is:
  - Package
  - Class
  - Interface
  - Signal
  - Node
  - Component
  - Artifact
  - Part
  - Port
  - Device

- **All** - select this radio button to audit all elements

- **Custom** - select this radio button to audit element types that you specify.

If you select the **Custom** option, the **Customize** button is made available. Click on this button to display a list of element types, and select the checkbox against each element type to include in the audit (or click on the **Select All** button to select every element type). Click on the **OK** button to save the selection.

**Note:**

Connectors are audited when they are connected to an element that is included in the Audit Options.
6.13.3 The Audit View

The Audit View provides an interface to the information that has been recorded by auditing. Open the Audit View by clicking on the:

- View | Audit View menu option, or
- Audit View button in the Other Views toolbar.

The Audit View is divided into three main areas:

- View controls
- Audit tree
- Record display.

The data in the Audit tree and Record display is determined by the view controls and mode and, if synchronizing with the Project Browser, by the package, diagram or element you have selected.

View Controls

The view controls provide a variety of settings for controlling auditing and the display of audit records.

Audit Tree

The audit tree displays the logs that have been recorded by auditing. What is displayed in the tree is affected by the settings of the View Controls section, such as:

- Sorting
- Filter (by time)
- Mode
- Auditing settings (what was actually recorded).

In the audit tree:

- The green tick indicates a creation
- The yellow pencil indicates an edit
- The red cross indicates a deletion.

If you right-click on an element in the audit tree (e.g. MyClass) a context menu displays. This menu enables you to locate the selected element in:

- The Project Browser
- Any diagrams in which it exists.
Record Display

The record display is in two parts: the identity of the selected change, and the actual change made.

Identity

The identity of a change consists of:

- The Windows username of the user that made the change
  
  **Note:**
  
  If security is enabled, the name is of the format: WindowsUsername(SecurityUser).

- When the change was made
- The path of the change (e.g. Class Class1 - Attribute Att1 - Attribute Constraint Constraint).

Changes

The changes are displayed in a table format, showing the Property (or data item) name, its original value before the change and its value after the change.

If you double-click on an item in the list of changes (or right-click and select the **Show Differences** context menu option) the **Difference** dialog displays. This shows the specific changes that have been made, highlighting the edited, created or deleted characters.

6.13.3.1 Audit View Controls

The **Audit View** controls provide a variety of settings for controlling auditing and the display of audit records.

- The **Load** (Advanced or Raw modes) or **Refresh** (Standard mode) button reloads the Audit Tree, updated with any new audit results.
- The **Search** button enables you to search through log items for a particular area. Log Items can be searched by Name, Type or GUID. The items are loaded and filtered with the current **Sort By**, **Time Filter**
and Mode settings. If you refresh the Audit View, you must run the search again.

- The Audit Settings button opens the Audit Settings dialog.

Sort-by Panel

The Sort By panel enables you to select one of three display settings:

- Type - changes are grouped under element type (such as Class or Requirement), and then grouped under the changed element.

- User - changes are grouped under user name. The number of changes for each user is also displayed. Under each user name, changes are grouped as for the Type sort.

- Deletions - displays only deletions, shown in chronological order. If used with the Search button, this can be useful for recovering information on missing elements.

Filter by

The Filter By Date/Time checkbox enables filtering by time periods, which you set using the Time Filter dialog; click on the Filter Settings button to display this dialog.

Select:

- Today - to display changes occurring today
• **Previous Hour** - to display changes occurring in the last 60 minutes
• **Previous 24 Hours** - to display changes occurring in the last 24 hours
• **Previous Week** - to display changes occurring in the last 7 days
• **Previous 30 Days** - to display changes occurring in the last 30 days
• **Previous Year** - to display changes occurring in the last 365 days
• **Custom** - to define your own time period, using the **From** and **To** fields.

**Note:**
The six pre-configured time periods automatically update when you click on the **Refresh** button. Custom periods are static and do not automatically update.

Changes that occur outside the filter period you select are not shown in the **Audit View**. Once you have set a filter period, it remains set if you deselect the **Filter By Date/Time** checkbox. The Custom time period, too, is retained so that you can re-use it or modify it later if required.

**Status Text**
Beneath the **Filter By Date/Time** checkbox, status text displays to show which **mode** has been selected, and which **time filter** is being applied to the data.

**Mode Button**
The **Mode** button enables you to change the mode of the **Audit View**. The button displays a drop-down menu from which you can select:

• **Standard** - to interact with the **Project Browser**
• **Advanced** - to load large sets of log items

**Note:**
When in **Advanced** mode, a special Audit Settings group can be displayed in the **Audit Tree**. This records when Auditing has been enabled and disabled, along with who made the change, and the date and time of the change.

• **Raw** - to display all audit records without sorting (although any search and filtering you define still apply). Additional database information is displayed; this additional information might be unimportant.

**Standard Mode**
In **Standard** mode, Auditing is automatically synchronized with the **Project Browser**.

When synchronized, and where changes have been made, the **Audit View** reflects your selection from the **Project Browser**. If you click on:

• An element, the **Audit View** displays the history for that element
• A package, the **Audit View** displays the history for that package and its immediate children (but not the contents of nested packages)
• A diagram, the **Audit View** displays the history for that diagram and its contents (which could be drawn from a wide area of the **Project Browser**).

**Advanced Mode**
In **Advanced** mode, you can load sets of audit data independent of the **Project Browser**. These sets of data display all significant changes, but you can reduce the selection by filtering by time or by running a search. **Advanced** mode also displays:

• Changes to the Audit Settings
• When Audit Operations are executed
• Security changes (which can be browsed in the same way as other changes).

**Raw Mode**
In **Raw** mode, all data recorded by auditing is displayed in chronological order. This enables you to see a progression of changes, which can be especially useful in determining date-time inconsistencies. Search and filters can still be applied, enabling you to view all of today’s changes in order, or all changes for a particular element in order, or both.
Note:
Some information displayed in Raw mode might be insignificant or only in machine-readable format.
6.13.4 Audit History Tab

When Auditing is turned on, an Auditing History tab is enabled in the Output window. To see this tab, you must have one or both of the Auditing and Element List views open.

Note:

If security is enabled, you must have Audit View permission to display data on the Audit History tab.

The information in the Audit History tab provides a history of changes to whichever element or connector you have selected in the:

- current diagram
- Project Browser
- Audit View, or
- Element List.

As you select different elements or connectors, the Audit History automatically updates to reflect your current selection. The information shows, for each change made to the element or connector:

- Who made the change
- When the change was made
- Where the change was made
- The value of the characteristic before the change
- The value of the characteristic after the change.
6.13.5 Auditing Performance Issues

**Impact of Auditing on Other Facilities**
Enabling auditing on a project increases the time taken for most actions. For most modeling tasks, this increase is insignificant; however, there are some situations where the difference is more substantial.

**Large Deletions**
Deleting large packages or package structures, or large numbers of elements, takes significantly longer with auditing on. You might disable auditing before performing such a deletion.

**XMI Imports**
Importing XMI takes longer with auditing enabled. A project-level option is provided for disabling auditing of XMI Imports.

**Reverse Engineering**
Reverse engineering code takes longer with auditing enabled. A project-level option is provided for disabling auditing of reverse engineering.
6.13.6 Audit View Performance Issues

Most operations in the Audit View are affected by the volume of use of the database - both by other facilities and by auditing itself. Some potential problems and their solutions are outlined below.

Navigating the Project Browser Within Auditing is Slow
- Try setting the time filter to a period in the immediate past, such as Today, Previous 24 Hours or Previous Week. This time period updates each time you open or refresh the Audit View.
- Save log items outside the project with the Save Logs button. If you then clear the logs you have just saved, the load time of the Audit View is reduced. You can reload logs into the project at any time, using the Load Logs button.

Navigating the Audit Tree is Slow
- Close the Audit History tab in the Output window.

The Audit View is Slow in Loading and Changing Modes
- Try setting the time filter to a period in the immediate past, such as Today, Previous 24 Hours or Previous Week. This time period updates each time you open or refresh the Audit View.
- Save log items outside the project with the Save Logs button. If you then clear the logs you have just saved, the load time of the Audit View is reduced. You can reload logs into the project at any time, using the Load Logs button.
6.14 Baselines, Differencing and Merges

Enterprise Architect (Corporate Edition) includes tools to help you manage and review changes to your models over time. These tools apply the concepts of Baselines, Differencing and Merges.

Baselines

Enterprise Architect (Corporate edition) provides a facility to create a Baseline or snapshot of the contents of a selected package and its child packages at a particular point in time, enabling you to later compare that branch of the model at that time with the current state of the branch. Baselines are stored in the same XML format as is used for version control, but are stored within the project in compressed format. You can also have parallel copies of parts of your model for team development, and create Baselines within each copy to merge changes into the project master.

Differencing

Differencing (Diff, or Compare) enables you to explore the differences between the current state of a specific part of your project, and previous or parallel versions captured in a Baseline or an XML file on disk.

Merges

Once Differencing is complete, you can merge information from the Baseline into the current project; it is not possible to go the other way. You can merge information manually, change by change, or automatically by electing to merge in all changes in one batch procedure. You can also revert completely to the original Baseline by importing the stored XMI directly, and merge in information and elements from a Baseline in a different project, making it possible to keep multiple versions of a single model in synch.

Notes:

- You use Baselines, Differencing and Merges essentially to compare two snapshots of a specific part of your project, to capture the differences between them and either roll back or incorporate selected changes or all changes. Enterprise Architect Corporate edition has another facility, Auditing, which you can switch on to perform continuous monitoring of changes across the project. You can dovetail your use of each facility to meet the range of your change management requirements.
- If a package under version control forms part of a Baseline, and that package is checked in to the model, you cannot merge the original data from the Baseline into that package.
- If security is enabled you must have Manage Baselines permission to create, import and delete Baselines, and Restore From Baseline permission to merge data from a Baseline. Security permissions are not required to select an existing Baseline and perform a comparison with the model.
6.14.1 Baselines

Enterprise Architect (Corporate edition) provides a facility to ‘Baseline’ (snapshot) a model branch at a particular point in time for later comparison with 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 stored within the model in compressed XML format. More than one Baseline can be stored against a single Enterprise Architect package.

You can also save a Baseline to an XML file for storage or archive, or for distributing to other users working on models derived from a master project.

Baselines are particularly useful during requirements management to check for changes, additions and deletions that 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 built into the Corporate and Professional versions of Enterprise Architect.

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 0.1a.
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, for example, the last 24 hours.
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 can be deleted if required to save space.

Important Considerations

- Baselines are based on the GUID or unique ID of a particular package. Enterprise Architect checks for that ID as the root element within the XML document being used as a Baseline. When you export a package to XML, the package you export is the root element. Likewise when you create a Baseline, the current package is the root package of the XML Baseline. When you save information in a version control system, the current version controlled package is again the root package of the document.
- Version controlled packages that themselves have version controlled child packages are not useful for Baseline comparisons, as Enterprise Architect expects a Baseline to contain all the information about child elements and child packages. Future versions of Enterprise Architect might relax this condition.
- XML files must be in the same format used by the Baseline engine - currently the UML 1.x format (plus Enterprise Architect extensions), which contains all the information necessary to reconstruct a UML model, even a UML 2.x model.

6.14.1.1 Manage Baselines

Enterprise Architect provides a range of facilities for working with and managing Baselines, through the Manage Baselines dialog.

To open this dialog, right-click on the package at the head of the appropriate model branch and select the Package Control | Manage Baselines menu option.
Option | Use to
--- | ---
Current Baselines For Package: <Name> | Review the baselines for the current model branch, listed by version reference with the highest alphabetical/numerical value at the top. If an entry is longer than the display area, a horizontal scroll bar displays at the bottom of the panel. Use this to scroll to the text that is not shown.
Show Differences | Run the Compare utility on the selected Baseline and the current model branch, and to display the differences between the two.
Restore to Baseline | Completely restore the model branch from the selected Baseline.
New Baseline | Create a new Baseline
Delete Selected | Delete the selected Baseline.
Load Other Baselines | Display a drop-down menu that enables you to load Baselines from another model, in either an .EAP file or a DBMS file.
  * For .EAP files, a browser displays; locate the required project file
  * For DBMS files, the Windows Data Link Properties dialog displays; select the data provider and click on the OK button to display the Select Data Source dialog, from which you select the required project.
In either case, the Connected To: message at the bottom of the Manage Baselines dialog changes to the name of the alternative model. To return the dialog to the original project, select the third option on the drop down list: Load From Selected Package.
Import File | Import an XML file from the file system as a new Baseline for this current model branch.
Export File | Export the selected Baseline to an XML file on disk.
Compare Model to File | Compare the selected model branch with an XML file on disk. A browser displays, which you use to locate the file.
### 6.14.1.2 Create Baselines

Open the New Baseline dialog by clicking on the New Baseline button on the Manage Baselines dialog.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Display the package name of the currently selected model branch.</td>
</tr>
<tr>
<td>Version:</td>
<td>Type a unique version reference for this Baseline. This can consist of any alphanumeric characters. The Manage Baselines dialog sorts the Baselines according to the value of this field.</td>
</tr>
<tr>
<td>Note:</td>
<td>Edit the default current time and date to any other value. The field is a single-line entry, for display on the Manage Baselines dialog (a one-line-per-entry list).</td>
</tr>
</tbody>
</table>

Click on the OK button to create a new baseline and return to the Manage Baselines dialog.
6.14.2 The Compare Utility (Diff)

Note:
This utility is available in the Corporate and Professional editions of Enterprise Architect.

Enterprise Architect has a comprehensive and powerful differencing utility built in. This utility enables you to compare a model branch in Enterprise Architect with:

- A Baseline created using the Baseline functionality (Corporate edition)
- A Baseline stored in a different model
- A file on disk created previously using the Enterprise Architect XML export facility (user selects file), or
- The current version-controlled XMI file on disk as created when using Version Control in Enterprise Architect (file automatically selected).

Compare (diff) lets you explore what has changed within a model over time and how previous versions of a model branch differ from what is currently in the model. It is even possible to do a full model comparison 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 the Package Control | Manage Baselines option.
2. The Project Browser context menu; select Package Control | Compare with XMI File (for a package not under version control).
3. The Project Browser context menu; select Package Control | Compare with Controlled Version (for a package under version control).

Differencing With Baselines

As a Baseline is stored within a model and contains all the information, elements and connections for a package at a point in time, it can be used within Enterprise Architect to track changes to model elements over time. The Differencing engine first builds a representation of the current package in memory, based on what is currently in the model. It then compares this with the stored Baseline, highlighting changes, new elements, missing elements and elements that have been moved to other packages. It is possible to filter the resultant output to display only one particular kind of change: for example, additions to the model.

See Example Comparison for an example of a model comparison.
6.14.3 Compare Options

The Compare Options dialog enables you to refine the output of the Compare utility when it compares the current model with a Baseline. To display the dialog, either:

- Click on the Options button on the Manage Baselines dialog, or
- Click on the Compare Options icon on the Compare Utility tab toolbar.

### Compare Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always Expand to Differences</td>
<td>Always display the list of elements fully expanded to show changes.</td>
</tr>
<tr>
<td></td>
<td>If you deselect the checkbox, when the Compare Utility tab is first opened it lists the package contents to element level, and you expand each element as required to show the changed items.</td>
</tr>
<tr>
<td></td>
<td>For large branches of the model, it is better to leave the checkbox unselected.</td>
</tr>
<tr>
<td>Show Elements that are</td>
<td>List elements that:</td>
</tr>
<tr>
<td></td>
<td>- Have been changed since the Baseline was created</td>
</tr>
<tr>
<td></td>
<td>- Are in the Baseline only (that is, have been deleted from the model since the baseline was created)</td>
</tr>
<tr>
<td></td>
<td>- Are in the model only (that is, have been created since the Baseline was created)</td>
</tr>
<tr>
<td></td>
<td>- Have not changed since the Baseline was created (you might generally leave this checkbox unselected).</td>
</tr>
<tr>
<td>Suppress these Changes</td>
<td>Exclude:</td>
</tr>
<tr>
<td></td>
<td>- Changes to diagrams</td>
</tr>
<tr>
<td></td>
<td>- Changes to the Date Modified field for an item</td>
</tr>
<tr>
<td></td>
<td>- Changes to the Date Created field for an item</td>
</tr>
<tr>
<td></td>
<td>- Child items of a deleted item.</td>
</tr>
</tbody>
</table>

If the Compare Utility tab shows the results of a Baseline comparison, when you click on the OK button the display refreshes to refine the information according to the options you have selected.
### 6.14.4 Example Comparison

The diagram below shows the result of a comparison between a package (Class Model) in the current project and that package in a Baseline (version 0.1) captured at an earlier date. Notice that:

- The results are displayed on the Compare Utility tab.
- 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 items in the hierarchy have been changed (Status Changed), deleted from the model (Baseline only), added to the model (Model only) and switched to different packages (Moved) since the Baseline was captured.
- If you click on an item with a Status entry in the left hand pane, the right-hand pane displays a table of properties showing the values of those properties in the current model and in the Baseline. For each property where there is a difference between the model and the Baseline, the row is highlighted. This example shows that the Entity element now named CreateLog was called Object1 in the Baseline, and the element was last modified on November 7, 2007.

The right panel might, for some fields, display only part of the value (such as Advanced Properties, above). It might also not be immediately obvious what a change is. In either case, you can double-click on the property to display full details and to highlight the exact differences. The following example shows the expanded and highlighted changes to Advanced Properties.
The Compare Utility tab enables you to perform operations on the reported information, using the toolbar, context menu and keyboard.

### 6.14.4.1 Compare Utility Tab Options

The Compare Utility tab enables you to perform operations on the reported information, using the toolbar, context menu and certain keyboard keys.

**Note:**
If a package under version control forms part of a Baseline, and that package is checked in to the model, you cannot merge the original data from the Baseline into that package.

**Toolbar**
The toolbar is at the top of the left-hand panel. The icons operate either on the comparison as a whole or on the currently-selected item in the left hand panel.

The toolbar icon names and functions are, from left to right:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Re-run the comparison to refresh the current display.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Merge To Model</td>
<td>Merge the values of the currently-selected item in the Baseline back into the model.</td>
</tr>
<tr>
<td>Next Change</td>
<td>Highlight the next changed item (skips Moved items).</td>
</tr>
<tr>
<td>Previous Change</td>
<td>Highlight the previously-changed item.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Fully expand the selected item.</td>
</tr>
<tr>
<td>Collapse All</td>
<td>Collapse the changed items in the selected item.</td>
</tr>
<tr>
<td>Expand To Changed Items</td>
<td>Expand the selected item to show changed items only.</td>
</tr>
<tr>
<td>Find in Project Browser</td>
<td>Highlight the item in the Project Browser.</td>
</tr>
<tr>
<td>Log To XML</td>
<td>Log the changes to an XML file. A browser displays, on which you specify the file name and location.</td>
</tr>
<tr>
<td>Compare Options</td>
<td>Display the Compare Options dialog.</td>
</tr>
<tr>
<td>Manage Baselines</td>
<td>Display the Manage Baselines dialog.</td>
</tr>
</tbody>
</table>

**Context Menu**

Each item in the hierarchy has a context menu, which you display by right-clicking on the item. The options displayed depend on the level of the item in the hierarchy, but include some or all of the following:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge from Baseline</td>
<td>Restore the item in the model to the Baseline state, or restore a deleted item from the Baseline.</td>
</tr>
<tr>
<td>Add from Baseline</td>
<td>(For the root node of the hierarchy on the Compare Utility tab.) Display the Merge dialog (see below) which enables you to specify options for rolling back the whole model branch to the Baseline state.</td>
</tr>
<tr>
<td>Delete from Model</td>
<td>Remove a recently-created item from the model.</td>
</tr>
<tr>
<td>Merge From Baseline (with Options)</td>
<td>(Object-level items). Re-run the comparison to refresh the current display.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Locate and highlight the item in the Project Browser.</td>
</tr>
<tr>
<td>Find in Project Browser</td>
<td>Fully expand the selected item.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Expand the selected item to show changed items only.</td>
</tr>
<tr>
<td>Collapse All</td>
<td>Collapse the changed items in the selected item.</td>
</tr>
<tr>
<td>Log To XML</td>
<td>Log the changes to an XML file. A browser displays, on which you specify the file name and location.</td>
</tr>
<tr>
<td>Compare Options</td>
<td>Display the Compare Options dialog.</td>
</tr>
</tbody>
</table>
Select the appropriate checkbox against each type of difference to roll back in the model from the Baseline.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changed</td>
<td>Restore all changed items in the model branch to the Baseline state.</td>
</tr>
<tr>
<td>In Baseline Only</td>
<td>Restore all deleted items to the model branch from the Baseline.</td>
</tr>
<tr>
<td>In Model Only</td>
<td>Remove all recently-created items from the model branch.</td>
</tr>
<tr>
<td>Moved</td>
<td>Restore all moved items to their original locations, as identified in the Baseline.</td>
</tr>
<tr>
<td>Full Restore from XMI</td>
<td>Completely restore the model branch to the version held in the Baseline XMI file, (using the XMI Import function). (Automatically selects all the other options.)</td>
</tr>
</tbody>
</table>

**Keyboard**

You can also use the following keyboard keys to move up and down the hierarchy, or to roll back changes:

- `[Ctrl]`+`[1]` - expand and highlight the next changed item
- `[Ctrl]`+`[1]` - expand and highlight the previous changed item
- `[Ctrl]`+`[~]` - undo the changes for a selected item (roll back to the Baseline values).
6.15 Traceability

Traceability identifies the way a given process has been, or is to be, developed in a system. The process can be an internal, model-management process, where you monitor work by asking questions such as 'what work has been done to realize this Requirement or Use Case?', or a business or system process that is being modeled, where you ask questions such as 'what Requirements, Use Cases, Classes, Components, Test Cases and other elements define the implementation and deployment of this process?'

Traceability also helps to clarify the aspects of a process that the model does not address. A process typically includes a range of manual, automated and external procedures. A correctly-structured model illustrates exactly what requirements and functionality service a particular process; any missing functionality must come from other systems, developments or procedures.

There are various tools in Enterprise Architect that enable you to trace the definition and implementation of a process from initial requirement to generated code or technical deployment, or vice versa. If you have performed any Transformations in developing your model and code, Enterprise Architect automatically creates Transformation Dependency connectors that you can trace - with the Hierarchy window - to establish what objects and code have been generated from each PSM element, or what the initial PSM element was for a generated object. Whether you use transformations or develop the stages of the model in other ways, you can build up a range of Traceability diagrams (Custom diagrams) to identify the development pathway and the dependencies between entities such as Requirements, Use Cases, Classes, Packages, Test Cases and other model artefacts, or possibly between these entities and the overall business process model.

Maintaining Traceability

The following Project Browser hierarchy represents a model that is structured to enable traceability. Notice firstly that themes are developed in the structure - the Requirements Model, Use Case Model and Abstract Implementation Model each contain units with the same functional names, such as Manage Users and Manage Inventory. These units help you to quickly develop a Traceability Model within your project, to trace development and implementation.

Traceability in this model is examined further in the following topics:

- Packages and Elements

Enterprise Architect User Guide
- Create Traceability Diagrams
- Traceability Tools
6.15.1 Packages and Elements

Analyzing the requirements of a system or process helps you identify units of functional activity in that system or process. You can represent these units with Packages, which you then populate with elements of relevance to the functional units and to the stage of development.

In the following four diagrams, you can see how Requirements and Use Case Packages are created to group the Requirements elements and Use Case elements that define and start to implement each of several functional areas. By maintaining the same structure in each model in the project (as mentioned in the Traceability topic), you also make it easy to trace the project development through the stages.

Requirement Packages for Online Bookstore Process
The Requirements diagram also makes it clear what Requirements form subsets of others, or are components of more than one other Requirement.
Use Case Packages For Online Bookstore Process
Use Cases in Manage Users Unit of Online Bookstore Process

The Use Case diagrams can also clarify what aspects of a process require or enable human intervention, and which require or enable system intervention.

Implementation Stage
For completeness, you could also consider the next stage, the implementation of some of these Use Cases, as represented by Class elements associated with this functional unit.
### 6.15.2 Create Traceability Diagrams

Having structured the models of a project to indicate directions and theoretical relationships between the models and packages, you can formalize these directions on a Traceability diagram, using **Realize**, **Trace** and similar relationships.

You initially create a Traceability diagram as a Custom diagram, but if you are creating the diagram manually you can use elements and relationships from other Toolbox pages to develop the diagram as broadly as is necessary.

You can also generate the diagram using the **Add | Related Elements** context menu option to automatically bring in elements linked to the selected element. It is probably better to add the elements in stages, one level at a time, but you could add several levels in one go to see how far the hierarchy extends and to identify relationship and element types to exclude from the ‘clean’ diagram. You could perform a similar operation, one element at a time, using the **Relationships window**.

The above diagram instantly shows how two levels of Requirements are realized by Use Cases, and which Requirement is realized by which Use Case(s). It also shows how some of the Use Cases are implemented by Class elements. Further, you can drill down on the Use Cases (or, in other Traceability diagrams, any other composite elements) to display more detailed diagrams showing how the Use Case meets the Requirement. The Close Account Use Case, for example, contains a Communication diagram and a Sequence diagram.

You can tailor your Traceability diagrams to depict any level of granularity and any stages of development that are appropriate. You might narrow the above diagram, for example, to show development from just the **Remove User** Requirement, and extend it to include Interfaces, Components, Test Case elements or any other facet of the system or process.

Whilst the Traceability diagram itself provides information on the definition, design and implementation of a business process feature, much more information can be obtained using tools such as the **Relationships Matrix** and **relationships Hierarchy window**.
6.15.3 Traceability Tools

The model structure and Traceability diagram act as starting points for tracing the definition, design and implementation of a feature of a system or process. By applying tools such as the Relationship Matrix and Hierarchy window to these, you can follow threads throughout the model to determine how the feature is implemented and tested. You can also obtain information on what elements realize and are realized by the elements in a given package, using the Dependency report and Implementation report, respectively.

Hierarchy Window

The Hierarchy window is a most useful and versatile traceability tool. Starting with a Traceability diagram or a package structure in the Project Browser, you can use the Hierarchy window to quickly explore the relationship chain of which any element is a component. When you click on the element, it immediately becomes the top point in the Hierarchy window.

If you require a rapid, broad-brush view of relationship flows in the project structure, starting with a general list of - say - all functional Requirements, you can use a combination of Model Search, Project Browser and Hierarchy window; this is a powerful tool for scanning your project, identifying how elements have been organized, and how they interact. For example, the Model Search would list all the requirements, you could rapidly click on each element and immediately see in the Project Browser where it has been grouped, and at the same time - in the Hierarchy window - how that element interacts with other elements in the model.

You can select any or all of the relationship types available in the Hierarchy window toolbox. The single type selected below is Realizes (Implements), and the selected element is the Delete User Use Case. The Hierarchy window then shows that Delete User implements REQ017, but this is also partially realized by the Close Account Use Case.

By moving the cursor around a diagram or the Project Browser, and/or changing the relationship type combinations in the Hierarchy window, you can quickly see how elements are connected and how they influence each other. For example, you can see that REQ017 is realized by two Use Cases, so you might then explore what else influences and is influenced by these two Use Cases. The Hierarchy window takes you well beyond what is likely to be depicted on any single diagram.

If you have used transformations to develop your model, the T icon displays the transformation dependencies that exist between an element in a PIM and elements in the PSMs.

Relationships Matrix

Using the Relationships Matrix, you can both create and study the relationships between, for example, the Requirements and Use Cases for a module. You might identify the 'theme' package (in this case, Manage Users) in the Requirements model and the Use Case model as the source and target packages, and explore the likely element and connector types in the packages. This, like the Traceability diagram, identifies which Requirements are (or should be) realized by which Use Cases. You can then perform similar checks with the Manage Users packages in, say, the Use Case and Implementation models.

The Source and Target field browsers enable you to examine child packages within the 'theme' package, and obtain further detail on how the feature at this stage is defined.
Other Tools

You can also obtain information on what elements realize and are realized by the elements in a given package, using the Dependency report and Implementation report, respectively.

You can trace how a Class or Interface element in a diagram or the Project Browser is implemented in code or, for tables, in DDL, using the Source Code viewer. For code, as you click on features in the element, the corresponding code is highlighted in the viewer.

From the perspective of model management or project management, you could also use the Audit View as a means of tracing the change history of a package or element.
6.16 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, including:

- People
- General Types
- Maintenance
- Metrics and Estimation
- UML
- Data Types
- Import and Export Reference Data

Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Manage Reference Data Update permission to update and delete reference items.
6.16.1 People

The People dialog enables you to control the following for your project:

- Project Authors
- Project Roles
- Project Resources
- Project Clients

To display this dialog, select the Settings | People menu option.

6.16.1.1 Project Authors

You can define the people who are working on a project, such as the authors of specific elements.

To define the project authors, select the Settings | People menu option. The People dialog displays, defaulted to the Project Author(s) tab.
Complete the fields as described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
</table>
| **Name** | Type the name of the person registered as a Project Author.  
If you are using a Windows Active Directory, you can select names from the directory.  
Click on the [ ... ] (Browse) button to display the Select Users dialog.  
You can also type a list of names separated by semi-colons. This enables you to define a group of people sharing a role, such as a team of Developers, Testers or Analysts. Do not leave any spaces between the names and the semicolon.  
**Note:** If you enter multiple names, Enterprise Architect adds them separately and in alphabetical order to the Defined Authors list. If you then click on one of these names, Enterprise Architect displays that name only in the Name field. |
| **Role** | Specify the role the Project Author plays in the project (e.g. Designer, Analyst, Architect).  
You can type a role name or click on the drop-down arrow and select a role defined through the Project Roles tab. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>If you type a role, this is not added to the roles on the Project Roles tab.</td>
</tr>
<tr>
<td><strong>Defined Authors</strong></td>
<td>Type any additional notes concerning the Project Author.</td>
</tr>
<tr>
<td></td>
<td>Review the Project Authors already defined.</td>
</tr>
</tbody>
</table>

**Notes**

Click on the **Save** button to add the new names to the Defined Authors list.

To add further Authors, click on the **New** button.

To delete a Project Author, click on the name in the Defined Authors list and click on the **Delete** button.

**Note:**

You can transport these author definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

---

**Select from User Directory**

If your company is using a Windows Active Directory, you can select the Project Author names from the local or corporate-wide directory. To do this, follow the steps below:

1. On the Project Author(s) tab, click on the [...] button. The Select Users dialog displays.

   - Select this object type:
     - Users
   - From this location:
     - RCHESTER01
   - Enter the object names to select (examples):
     - 
   - Advanced
   - OK
   - Cancel

2. Click on the **Object Types** button and select the checkbox for the object type **User**.
3. Click on the **OK** button to return to the Select Users dialog.
4. Click on the **Locations** button. The Locations dialog displays.
5. Click on the appropriate area or level of the directory, and click on the OK button. The Select Users dialog redisplays.

6. In the Enter the object names to search field, type the first letter of the user name to search for.

7. Click on the Check Names button. The Multiple Names Found dialog displays, listing the names starting with the specified letter found in the directory location.

8. Click on the required name (or press and hold [Ctrl] and click on several names), and click on the OK button. The simple Select Users dialog redisplays, with the selected names listed.

9. Click on the OK button. The Project Authors tab redisplays, with the selected name or names in the Name(s) field.

6.16.1.2 Project Roles

People associated with a project play a role in analysis, design or implementation, such as Application Analyst, Architect, Developer and Project Manager. Project roles define the activities that resources can undertake.

To define the role types that are captured within Enterprise Architect, select the Settings | People menu option and, on the People dialog, click on the Project Roles tab.
To add further roles, click on the **New** button and complete the fields as described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role</td>
<td>Type the name of the role.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a short description of the role.</td>
</tr>
<tr>
<td>Notes</td>
<td>Type any additional information related to the role.</td>
</tr>
<tr>
<td>Defined Roles</td>
<td>Review all roles that have been previously defined in Enterprise Architect.</td>
</tr>
</tbody>
</table>

Click on the **Save** button to add the new role to the Defined Roles list.

The Defined Roles list is available for selection for any element in the model; for example, you can select roles on the Project Author tab of the People dialog, and the Resource Allocation tab of the Project Management window. You can also specify other roles on these dialogs, but such roles are not added to the Defined Roles list.

To delete a role, click on the role type in the Defined Roles list and click on the **Delete** button.

**Notes:**

- Deleting a role has no effect on any Project Author definition having this role; the deleted role becomes a simple text entry in the Project Author definition.
- You can transport these role definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
### 6.16.1.3 Project Resources

*Resources* are, for example, project authors, analysts, programmers and architects. That is, anyone who might work on the system over time, either adding to the model or programming and designing elements of the system outside Enterprise Architect.

To record information on project resources, select the **Settings | People** menu option and, on the **People** dialog, click on the **Resources** tab.

![Resources Tab](image)

Complete the fields as described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type the name of the person listed as a resource. The resource name is available for use in Resource Management.</td>
</tr>
<tr>
<td>Organization</td>
<td>Type the name of the organization employing the resource.</td>
</tr>
<tr>
<td>Role(s)</td>
<td>Type the role the resource plays in the project (e.g. Designer, Analyst, Architect).</td>
</tr>
<tr>
<td>Phone 1, Phone 2, Mobile, Fax</td>
<td>Type the contact telephone numbers for the resource.</td>
</tr>
<tr>
<td>Email</td>
<td>Type the email address for the resource.</td>
</tr>
<tr>
<td>Notes</td>
<td>Type any additional notes on the resource.</td>
</tr>
<tr>
<td>Available Resources</td>
<td>Review resources that have already been defined.</td>
</tr>
</tbody>
</table>
Click on the **Save** button to add the new resource to the **Available Resources** list.

To add further resources, click on the **New** button.

To delete a resource, click on the name in the **Available Resources** list and click on the **Delete** button.

**Note:**
You can transport these resource definitions between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu.

### 6.16.1.4 Project Clients

*Project clients* are the eventual owners of the software system.

To capture client details associated with the current model, select the **Settings | People** menu option and, on the **People** dialog, click on the **Project Clients** tab.

#### Complete the fields as described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type the name of the client.</td>
</tr>
<tr>
<td>Organization</td>
<td>Type the name of the organization that employs the client.</td>
</tr>
<tr>
<td>Role(s)</td>
<td>Type the role the client plays in the project (e.g. Manager, Sponsor).</td>
</tr>
</tbody>
</table>
### Option | Use to
--- | ---
Phone 1, Phone 2, Mobile, Fax | Type the contact telephone numbers for the client.
Email | Type the email address of the client.
Notes | Type additional notes on the client.
Defined Clients | Review clients that have already been defined.

Click on the **Save** button to add the new client to the Defined Clients list.

To add details of further clients, click on the **New** button.

To delete a client record, click on the name in the Defined Clients list and click on the **Delete** button.

**Note:**

You can transport these client definitions between models, using the **Export Reference Data** and **Import Reference Data** options on the Tools menu.
6.16.2 General Types

The General Types dialog enables you to configure:

- Status types
- Constraint types
- Constraint Status types
- Requirement types
- Scenario types

To display the General Types dialog, select the Settings | General Types menu option.

6.16.2.1 Status Types

You can configure a basic list of status types used in Enterprise Architect. Note that whilst most dialogs use this list, not all do so.

To configure status types, select the Settings | General Types menu option. The General Types dialog displays at the Status tab.
Create New Status Type

When you display the Status tab, the fields default to the definition of the first type in the Type list. To add a new type, click on the New button and:

- In the Status field, type the name of the status
- In the Description field, type a short description of the status
- Click on the Save button.

The status type displays in the Type list. Add the status definition as described in the following sections.

**Note:**
You can transport the status types (and the colors assigned to status types) between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

Status Type Color

It is possible to assign a color to each status type, which gives a visual indication of the status of each diagram object. Select a named status type from the Type list, click on the Status Type Color drop-down list and select the color for that status. Click on the Save button to keep your changes.

**Note:**
To ensure status colors display on your diagrams, open the Options dialog at the Objects page and select the Show status colors on diagrams checkbox.

Apply Colors to UML Elements

By default, status colors only apply to Requirement, Issue and Change elements. You might decide to also...
apply these colors to other UML elements, such as Use Cases or Classes. To do this, click on the **Applies to** button and select the checkbox against each required element type in the **Applied Status Colors** list.

![Applied Status Colors](image)

**Note:**

Requirement, Issue and Change elements have a status color compartment, but other elements do not. The status color for these elements is applied to the element shadow. Therefore, on the **Options** dialog **Objects** page you must also select the **Shadows on** checkbox.

### 6.16.2.2 Constraint Types

The **Constraint** tab of the **General Types** dialog enables you to define constraints. These are picked up in a variety of places where constraints might fall into more categories than the basic (default) **Pre-**, **Post-** and **Invariant** conditions.

To access this dialog, select the **Settings | General Types** menu option. Click on the **Constraint** tab.
To add a new constraint, click on the New button and:

- In the **Constraint** field, type the name of the constraint; for example, *Assumption*
- In the **Description** field, type a brief description of the constraint
- In the **Note** field type any additional information required
- Click on the **Save** button.

The constraint displays in the **Defined Constraint Types** list.

**Note:**

You can transport these constraints between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

### 6.16.2.3 Constraint Status Types

You can configure the basic list of **constraint status types** used in Enterprise Architect, using the **Constraint Status** tab of the General Types dialog.

To access this dialog, select the Settings | General Types menu option. Click on the **Constraint Status** tab.
To add a new constraint status type, click on the New button, type the status type in the Status field, and click on the Save button. The constraint status type displays in the Status list.

**Note:**
You can transport these constraint status types between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

### 6.16.2.4 Requirement Types

The Requirement tab of the General Types dialog enables 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 the Settings | General Types menu option. Click on the Requirement tab.
To add a new requirement, click on the **New** button and:

- In the **Requirement** field type the name of the requirement
- In the **Description** field type a short description of the requirement
- In the **Weight** field type the weighting to apply to the requirement
- In the **Note** field, type any additional information on the requirement
- Click on the **Save** button.

The requirement displays in the Defined Requirement Types list.

**Note:**

You can transport these requirement types between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu.

### 6.16.2.5 Scenario Types

A drop-down list of scenario types is available in the scenario dialog, with the standard types **Basic Path** and **Alternate Flow**. You can set additional scenario types using the **Scenario** tab of the **General Types** dialog.

To access this dialog, select the **Settings | General Types** menu option. Click on the **Scenario** tab.
To add a new scenario, click on the **New** button and:

- In the **Scenario Type** field type the name of the scenario type
- In the **Description** field type a short description of the scenario type
- In the **Weight** field type the weighting to apply to the scenario type
- In the **Note** field, type any additional information on the scenario type
- Click on the **Save** button.

The constraint displays in the **Defined Scenario Types** list.

**Note:**

You can transport these scenario types between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu.
6.16.3 Maintenance

To control Problem types and Testing types for your project, select the Settings | Maintenance menu option to display the Maintenance dialog.

Note:
You can transport problem types and test types together between models, using the Export Reference Data and Import Reference Data options on the Tools menu. The combined file is a Maintenance Types file.

6.16.3.1 Problem Types

For the maintenance and change control screens, you can use the Maintenance dialog to 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 the Settings | Maintenance menu option. The Maintenance dialog displays, defaulting to the Problem Types tab.
To add a new problem type, click on the **New** button and:

- In the **Problem Type** field type the name of the problem type
- In the **Description** field type a short description of the problem type
- In the **Weight** field type the weighting to apply to the problem type
- In the **Note** field, type any additional information on the problem type
- Click on the **Save** button.

The problem type displays in the **Defined Types** list.

**Note:**

You can transport these problem types between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu. You transport the problem types together with test types as a **Maintenance Types** file.

### 6.16.3.2 Testing Types

Use the **Test Types** tab of the **Maintenance** dialog to add testing types to the basic set that comes with Enterprise Architect. Typical test types are load tests, performance tests and function tests.

To access this dialog, select the **Settings | Maintenance** menu option. The **Maintenance** dialog displays. Click on the **Test Types** tab.
To add a new test type, click on the **New** button and:

- In the **Test Type** field type the name of the testing type
- In the **Description** field type a short description of the testing type
- In the **Weight** field type the weighting to apply to the testing type
- In the **Note** field, type any additional information on the testing type
- Click on the **Save** button.

The testing type displays in the Defined Types list.

**Note:**

You can transport these test types between models, using the **Export Reference Data** and **Import Reference Data** options on the **Tools** menu. You can either export the test types together with problem types, as a **Maintenance Types** file, or separately as a **Test Types** file.
6.16.4 Metrics and Estimation

TCF values, EFC values and Default Hour Rate for a project are controlled from the Estimation Factors dialog.

Risk, metric and effort types for a project are controlled from the Project Indicators dialog.

For further information on these see the Project Management and Resource Management topics, or specifically:

- Technical Complexity Factors
- Environment Complexity Factors
- Default Hours
- Effort Types
- Metric Types
- Risk Types.
6.16.5 UML Types

The UML Types dialog enables you to configure stereotypes, Tagged Value types, and the cardinality list for your project.

Select the Settings | UML menu option to display this dialog.

6.16.5.1 Stereotype Settings

Enterprise Architect has an extensive set of Standard Element Stereotypes that you can apply to any UML construct. Using the Stereotypes tab of the UML Types dialog, a Technical Developer can also customize the stereotypes for your project by adding, modifying and deleting them.

Stereotypes can be modified to make use of metafiles (image files) or customized colors, or to make use of the Enterprise Architect Shape Scripts to make new element shapes to determine the shape and dimensions of the element.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have Configure Stereotypes permission to add, modify or delete stereotypes.

To display the Stereotypes tab, select the Settings | UML menu option. The UML Types dialog displays, showing the Stereotypes tab.

For information on customizing stereotypes, see SDK for Enterprise Architect.
6.16.5.1.1 Shape Editor

The Shape Editor enables a Technology Developer 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 is drawn for every element of that stereotype.

```java
// BASIC SHAPES
shape main
{
    setfillcolor(255,0,0); // (R,G,B)
    rectangle(0,0,90,30); // (x1,y1,n2,y2)
    setfillcolor(0,255,0); // (R,G,B)
    ellipse(0,30,90,40); // (x1,y1,x2,y2)
    setfillcolor(0,0,255); // (R,G,B)
    rectangle(0,60,20); // (x1,y1,x2,y2)
}
```

Notes:
- Shape Scripts adopt the same color gradient settings as normal elements, as defined in the Standard Colors page of the Options dialog.
- Custom shapes do not function in Sequence (Interaction) diagrams.

For information on creating Shape Scripts, see SDK for Enterprise Architect.

6.16.5.2 Tagged Value Types

Tagged Values are used in a variety of places within Enterprise Architect to specify additional information about an element. The Tagged Value Types tab of the UML Types dialog enables a Technology Developer to rapidly create masked Tagged Values, using a range of predefined Tagged Values, and then to create structured tags that adhere to a specific format. For example, for model features that use the predefined tag Boolean you can use the Tagged Values window to assign a value of True or False and no other value.

You can also add default Tagged Value names and create predefined reference data Tagged Value types and custom Tagged Value types.

Any Tagged Value names created display in the drop-down lists of Tagged Value names in the Tagged Values dialogs for elements, operations and attributes. For more information regarding the use of Tagged Values see the Tagged Values Window topic.

To display the Tagged Value Types tab, select the Settings | UML menu option to display the UML Types dialog, and click on the Tagged Value Types tab.
For further information on adding and modifying Tagged Values, see [SDK for Enterprise Architect](#).

**Note:**

You can transport these Tagged Value Type definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu. Tagged Value Types are exported as Property Types.

### 6.16.5.3 Cardinality

The Cardinality Values tab of the UML Types dialog enables you to add, modify and delete values in the default cardinality list.

The cardinality values are used to define the multiplicity of source and target elements in relationships. This is the range of instances of the role that can be active in the relationship; for example, one employee can be assigned to tasks; for the target role you define the range of instances (e.g. tasks) the employee could be assigned to.

The values have the following formats:
- `*`, or `0..*` - zero, one or many instances
- `0..n` - zero or up to n instances, but no more than n
- `n` - exactly n instances
- `n..*` - n, or more than n instances.

To access this dialog, select the Settings | UML menu option. Click on the Cardinality Values tab.
To add a new cardinality value, click on the New button. To modify an existing value, click on it in the Cardinality list.

In the Cardinality field, type the required cardinality value. Click on the Save button.

**Note:**

You can transport these cardinality values between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
### 6.16.6 Data Types

Different programming languages support different inbuilt data types. The [Programming Languages Datatypes](#) dialog enables you to extend and manage the set of inbuilt data types associated with a language as well as create new programming languages for use within Enterprise Architect.

**Notes:**
- In the Corporate edition of Enterprise Architect, if security is enabled you must have [Configure Datatypes](#) permission to update and delete data types.
- You can delete data types that you have defined, but you cannot delete any of the predefined data types.

To access this dialog, select the **Settings | Code Datatypes** menu option.

![Datatypes dialog](image)

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Name</td>
<td>Specify the name of the programming language.</td>
</tr>
<tr>
<td>Add Product</td>
<td>Add a new programming language to the drop-down fields for Class elements within the Enterprise Architect model and enable the new language to be made available to the <a href="#">Code Template Editor</a> once at least one datatype has been added to the language.</td>
</tr>
<tr>
<td>Datatype</td>
<td>Specify the name of the datatype; this is the language-specific name of the datatype.</td>
</tr>
<tr>
<td>Common Type</td>
<td>Specify the common type, the generic name of the datatype; for example, the Java <code>boolean</code> datatype has a common datatype <code>Boolean</code>.</td>
</tr>
<tr>
<td>New</td>
<td>Create a new data type.</td>
</tr>
<tr>
<td>Save</td>
<td>Save the newly created datatype.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete the selected datatype.</td>
</tr>
</tbody>
</table>

**Note:**
You can transport these data types between models, using the [Export Reference Data](#) and [Import Reference Data](#) options on the **Tools** menu.
6.16.7 Import and Export Reference Data

Reference data (including Glossary and Issue information) can be imported and exported to XML files for convenient update of models.

Examples of where this can be useful include:

- 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 and so on from one model to another.

6.16.7.1 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.

Procedure

To export data, follow the steps below:

1. Select the Tools | Export Reference Data menu option. The Data Exporter dialog displays.

2. From the Name list, select the table to export. You can select one or more tables for a single file.

3. Click on the Export button.

4. When prompted to do so, enter a valid file name with a .XML extension. This exports the data to the file. You can use any text or XML viewer to examine this file.

The data exported includes all instances of the data type in the project; for example, all defined cardinality values, or all RTF Style Templates.

Note:

If there are no instances of a selected data type in the project, the export does not generate any output for that data type in the XML file.

For information on each category of data you can export, refer to the following topics:

- Cardinality Types
- Constraint Status Types
- Defined Metric Types
- Defined Problem Types
6.16.7.2 Import Reference Data

When you import data into Enterprise Architect, the system merges the incoming data with the existing data. If a record already exists it is updated to the new values. If the record does not exist, Enterprise Architect adds a new record. Enterprise Architect never deletes records.

For a list and explanation of the data types you could import, see the Export Reference Data topic.

Import Data

To import data, follow the steps below:

1. Select the Tools | Import Reference Data menu option. The Import Reference data dialog displays.
2. Click on the **Select File** button and select the filename to import data from. This must be an XML file produced by the Enterprise Architect **Data Exporter**.

3. If you have entered the name of a valid file, a list of available tables to import displays.

4. Select one or more of the tables to import.

5. Click on the **Import** button to start the process. A message displays when the import is complete. Generally the process is quite fast.
Part VII
7 License Management

The License Management dialog in Enterprise Architect enables you to upgrade Enterprise Architect and to register Add-Ins.

**Note:**
From Enterprise Architect release 7.0, support for Add-Ins created before 2004 is no longer available. If an Add-In subscribes to the `Addn_Tmpl.tlb` interface (2003 style), it will fail on load.

To access License Management from within Enterprise Architect, select the Help | Register and Manage License Key(s) menu option. The License Management dialog displays, listing the currently-registered keys, their expiration date and the product each key applies to.

Use the Add Key button to enter a new registration key.

Currently Registered Keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Expires</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADP8D888-00</td>
<td>Never</td>
<td>EA Corporate Edition</td>
</tr>
<tr>
<td>V21-HWC-9X</td>
<td>Never</td>
<td>MDG Technology for Zachman Framework</td>
</tr>
<tr>
<td>QSH1-1CIO-4V</td>
<td>Never</td>
<td>MDG Integration for Eclipse</td>
</tr>
</tbody>
</table>

Use the buttons on the dialog as required:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Key</td>
<td>Display the Add Registration Key dialog, which enables you to:</td>
</tr>
<tr>
<td></td>
<td>- Add a new key to update to a higher version of Enterprise Architect or to register an Add-In.</td>
</tr>
<tr>
<td></td>
<td>- Obtain a key from the Enterprise Key Store (available for version 4.51 and above). For more information on adding keys see the Add License Key topic.</td>
</tr>
<tr>
<td>Remove Key</td>
<td>Make the Add-In or current version of Enterprise Architect inoperable.</td>
</tr>
<tr>
<td>Copy</td>
<td>Place the highlighted key into the clipboard.</td>
</tr>
<tr>
<td>Close</td>
<td>Close the dialog.</td>
</tr>
<tr>
<td>Help</td>
<td>Display the help for this topic.</td>
</tr>
</tbody>
</table>

You can also run the following tasks from the License Management dialog:

- Register a Full License
- Upgrade an Existing License
Register an Add-In
7.1 Finding Your License Information

You can find information on your Enterprise Architect license in the About Enterprise Architect dialog; select the Help | About EA menu option.
7.2 Add License Key

Two types of key can be used in conjunction with Enterprise Architect:

- Private keys allow you to register an Enterprise Architect license (Desktop, Professional or Corporate) or an Add-In product key (such as MDG Link for Eclipse or MDG Link for Visual Studio.NET) to the machine and user account that you are currently using.
- Shared keys allow you to temporarily obtain a product key from a central shared key store on your site. Shared Keys are only available with the purchase of a floating license edition and require Enterprise Architect version 4.51 or higher. For more information, see Enterprise Architect Corporate Floating License.

Add a Private Key

To add a private key, follow the steps below:

1. Select the Help | Register and Manage License Key(s) menu option. The License Management dialog displays.
2. Click on the Add Key button. The Add Registration Key dialog displays.
3. Click on the Enter Private Key tab.
4. In the Name and Company fields, type your user name and company name. Into the registration key field, copy the registration key.
5. Click on the OK button to confirm the key selection.

Add a Shared Key

Shared Keys require a shared license file to be configured on your network by your license administrator. Only the Key Administrator is required to install the Sparx Enterprise Key Store application. End users simply connect to the configured key file using Enterprise Architect as described below. No additional software is required to be installed.

Note:
If any error messages are displayed while attempting to obtain a shared key, see Keystore Troubleshooting.

To add a shared key, follow the steps below:

1. Select the Help | Register and Manage License Key(s) menu option. The License Management dialog displays.
2. Click on the Add Key button. The Add Registration Key dialog displays.
3. Click on the Get Shared Key tab.
4. In the **Name** and **Company** fields, type your user name and company name.

5. In the **Shared key store** field, click on the `[ ... ]` (Browse) button to locate and select the shared key store.

6. In the **Select a Product** field, click on the appropriate product name.

7. Click on the **OK** button.
### 7.3 Keystore Troubleshooting

<table>
<thead>
<tr>
<th>Message Displayed:</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error reading Key Store file: (Access is denied)</strong></td>
<td>All users who are to use the shared key facility require <strong>Read, Write and Modify</strong> access to the <code>sskeys.dat</code> file containing the shared keys. Please verify that all required users have sufficient permissions to the file and try again. If the problem continues, contact Sparx Support. <strong>Tip:</strong> Review the effective permissions calculated at the location of the key file for the user account reporting the problem. You should closely examine the permissions for both the Network Share and the File System. It is possible that these permissions have been overwritten at some point.</td>
</tr>
<tr>
<td><strong>Error reading Key Store file: (Key File has been moved)</strong></td>
<td>As a security measure in the key store, the hard drive serial number is recorded when the file is created. The file then cannot be moved from the original location in which it was created. If the key store has to be re-located for any reason, the administrator should re-create the key store in the new location using the original license keys. This issue is commonly seen after a file server has undergone a hardware upgrade in which the physical hard drives have been replaced. Problems could also occur if the drive used is part of a RAID configuration. This message could also appear where the key store exists on a Novell-based file system. When creating the key store, the administrator is prompted to confirm that the key store is to be located on a Novell Netware file server. If the administrator clicks on the <strong>Yes</strong> button, the key store instead records the logical path used to create it, and all users must connect to the key store using this same path. The recorded path is case-sensitive and must be an exact match.</td>
</tr>
</tbody>
</table>
### 7.4 Upgrade an Existing License

Enterprise Architect comes in three editions: Desktop, Professional and Corporate. If you are using the Desktop or Professional edition, you can 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 upgrades an existing license to a higher edition. Once you have purchased and received the appropriate key, use the following procedures to unlock additional features. The procedure for Enterprise Architect version 7.0 and later releases differs from the procedure for earlier releases.

**Note:**

The Lite version and the Trial version cannot be registered or upgraded. If you have purchased Enterprise Architect, you must download the registered version from [www.sparxsystems.com/securedownloads/easetupfull.exe](http://www.sparxsystems.com/securedownloads/easetupfull.exe) before you can enter your registration key.

**Tip:**

Once you have successfully completed the upgrade, select the Help | About EA menu option. 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 have to reinstall Enterprise Architect, you can register it with this key, so you won't have to go through the upgrade process again.

#### Upgrade Enterprise Architect Version 6.5 and Earlier

To upgrade from one license edition to another, follow the steps below:

1. Make sure you have a valid upgrade key purchased from Sparx Systems; you typically receive this in an email or PDF format.
2. Open Enterprise Architect.
3. Select the Help | Register and Manage License Key(s) menu option. The License Management dialog displays.
4. Click on the Add Key button or the Upgrade button to enter a new license key.
5. If you selected the Add key option, the Add Registration Key dialog displays. Enter the key you received for the upgraded edition of Enterprise Architect, including the { and } bracket characters (use copy and paste from an email to avoid typing mistakes).
6. If you selected the Upgrade option, the Upgrade Key dialog displays. Enter the key you received for the upgraded edition of Enterprise Architect, including the { and } bracket characters (use copy and paste from an email to avoid typing mistakes).
7. Click on the OK button. If the key is valid, Enterprise Architect modifies the Current Version field to reflect the upgrade.

8. Close Enterprise Architect and restart to enable the unlocked features.

**Upgrade Enterprise Architect Version 7.0 and Later**

To upgrade from one license edition to another, follow the steps below:

1. Make sure you have a valid upgrade key purchased from Sparx Systems; you typically receive this in an email or PDF format.
2. Open Enterprise Architect.
3. Select the Help | Register and Manage License Key(s) menu option. The License Management dialog displays.

4. Click on the Add Key button; the Add Registration Key dialog displays.

5. In the Name and Company fields, type your name and company name.
6. To avoid typing mistakes, copy the key you received for the upgraded edition of Enterprise Architect - including the { and } bracket characters - from the email and paste the key into the **Copy registration key** field.


8. Click on the **OK** button, and then on the **Close** button to continue working in Enterprise Architect.
7.5 Register Add-In

To register Add-Ins for Enterprise Architect, follow the steps below:

Register an Add-In for Enterprise Architect

1. Purchase one or more licenses for the Add-In from your Add-In provider. Once you have paid for a licensed version of the Add-In, you receive (via email or other suitable means) a license key for the product.
2. Save the license key and the latest full version of the Add-In.
3. Run the Add-In's setup program to install the Add-In.
4. In Enterprise Architect, select the Help | Register and Manage License Key(s) menu option, or the Add-Ins | Enter License Key for <Add-In name> menu option.

   ![License Management dialog](image)

   The License Management dialog displays.

5. Click on the Add Key button. The Enter Registration Key dialog displays.

   ![Enter Registration Key dialog](image)

   Name: John Field
   Company: Spass Systems

   Copy registration key into space below, then press Register button

   ABDE123:0385-4d1:AA60-I

   Register | Cancel | Help

6. Type in the key you received with the Add-In, including the { and } characters.
7. When the Add-In has been added successfully, close down Enterprise Architect and restart it to apply the changes.
8 Project Management

Enterprise Architect provides strong support for managing your project. Project Managers can use Enterprise Architect to estimate project size, measure risk and effort, and assign resources to elements. Enterprise Architect also provides support for testing, change control and maintenance.

Metrics and Estimation

Project estimation is working out how much time and effort is required to build and deploy a solution. Enterprise Architect provides the Use Case metrics facility as a means of measuring the complexity of a system and getting an indication of the effort required to implement the model, and the project timescale. You base these estimates on carefully-calibrated metrics.

Resource Management

Resources are the people who work on a project. You can assign roles to them and allocate tasks on specific model elements, which enables tracking of effort and estimation of time to complete.

Project Maintenance

During a project you monitor and manage the development and progress of individual model elements. You can record problems, changes, issues and tasks that affect these individual elements as they arise, and document the solution and associated details.

Similarly, Enterprise Architect helps you to manage changes and issues that apply to the whole system.

Project Tasks and Issues

In the course of a project, there are various non-technical tasks that are vital to the successful management and completion of the project, such as meetings. Enterprise Architect helps you to record and monitor these, and to manage non-technical project issues as they arise.

Testing

Enterprise Architect enables you to create and manage test scripts for model elements, covering unit, integration, scenario, system and acceptance tests. You can import tests from other elements, generate them from scenarios, and generate test documentation and reports. You can also indicate the presence of tests on an element by displaying test information on the element in a diagram.

See Also

- Project Glossary
- Update Package Status
- Manage Bookmarks
8.1 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 Enterprise Architect 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/UCMetrics.htm.

Calibrating

The following values must be carefully calibrated in order to gain the best possible estimates:

- **Technical Complexity Factors** which are values that attempt to quantify the difficulty and complexity of the work in hand
- **Environment Complexity Factors** which are values that attempt to quantify non-technical complexities such as team experience and knowledge
- **Default Hour Rate** which sets the number of hours per Use Case point.

Estimating

Once you have entered all the calibration values, you can estimate the project timescale through the Use Case Metrics dialog.
8.1.1 Technical Complexity Factors

Technical Complexity Factors are used in the Use Case Metrics estimation technique. You can add or modify these factors using the Estimation Factors dialog.

To open this dialog, select the Settings | Estimation Factors menu option. Click on the Technical Complexity Factors tab.

Defined Technical Types

This 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 value, work together with the Environment Complexity Factors to skew the overall complexity up or down, depending on the level of technical complexity and the corresponding level of environmental support.

**Note:**

You can transport the Technical Complexity Factors between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

**Weight**

The TCF Weight indicates how much technical complexity you assign to a factor. For example, ‘the system is to be developed in ADA’ might warrant a higher weight than ‘the system is to be a shell script’. A weight evaluates its respective factor, but is irrelevant 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 can be adjusted to suit a project's specific requirements.

**Value**

For most purposes, the only table column requiring adjustment is Value, which indicates the degree of influence a particular factor has on the project. As a suggested gauge, a value of 0 indicates no influence, 3 indicates average influence and 5 indicates strong influence.
8.1.2 Environment Complexity Factors

Environment Complexity Factors are used in the Use Case Metrics estimation technique. You can add or modify these using the Estimation Factors dialog.

To open this dialog, select the Settings | Estimation Factors menu option. Click on the Environment Complexity Factors tab.

Defined Environment Types

This 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 value, work together with the Technical Complexity Factors (TCFs) to skew the overall complexity up or down, depending on the level of technical complexity and the corresponding level of environmental support.

Note:

You can transport the Environment Complexity Factors between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

Weight

A Weight evaluates its respective factor's complexity in comparison to other factors, but is irrelevant 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 can be adjusted to suit a project's specific requirements.

**Value**

For most purposes, the only table column requiring adjustment is **Value**, which indicates the degree of influence a particular factor has over the project. As a suggested gauge, a value of 0 indicates no influence, 3 indicates average influence and 5 indicates strong influence.
8.1.3 Estimating Project Size

**Note:**
This technique is of value only once you have developed a 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.

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 the **Project | Use Case Metrics** menu option, or
- Right-clicking on the package of interest in the **Project Browser** and selecting the **Documentation | Package Metrics** menu option.

### Use Cases

<table>
<thead>
<tr>
<th>Use Case Model</th>
<th>Root Package</th>
</tr>
</thead>
</table>

- **Reload**
  - Re-run the search, usually after you change the filter criteria.

- **Phase like**
  - Include Use Cases with a phase that matches the wildcard value in the field (use * to match any characters, for example 1.* for 1.1 and 1.2).

- **Keyword like**
  - Include Use Cases with a keyword that matches the wildcard value in the field (use * to match any characters).

- **Use Cases**
  - Check the total count of Use Cases in estimate.

### Technical Complexity Factor

- **Unadjusted TCF Value (UTV):** 47
- **TCF Weight Factor (TwF):** 0.01
- **TCF Constant (TC):** 0.6
- **TCF = TC x (TwF x UTV):** 1.07

### Environment Complexity Factor

- **Unadjusted ECF Value (UEV):** 21.5
- **ECF Weight Factor (EWF):** 0.03
- **ECF Constant (EC):** 1.4
- **ECF = EC x (EWF x UEV):** 0.735

### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Package</td>
<td>Confirm the root package in the hierarchy. All Use Cases under here could potentially be included in the report.</td>
</tr>
<tr>
<td>Reload</td>
<td>Re-run the search, usually after you change the filter criteria.</td>
</tr>
<tr>
<td>Phase like</td>
<td>Include Use Cases with a phase that matches the wildcard value in the field (use * to match any characters, for example 1.* for 1.1 and 1.2).</td>
</tr>
<tr>
<td>Keyword like</td>
<td>Include Use Cases with a keyword that matches the wildcard value in the field (use * to match any characters).</td>
</tr>
<tr>
<td>Use Cases</td>
<td>Check the total count of Use Cases in estimate.</td>
</tr>
<tr>
<td>Technical Complexity Factor</td>
<td>Review the parameters that describe the degree of technical complexity of the project. While the unadjusted TCF value comes from the Technical Complexity Factor tab of the Metrics and Estimation Types dialog, the other values compose the Use Case Points Method formula. Modify these fields with caution. The final project estimate is directly proportional to the TCF.</td>
</tr>
<tr>
<td>Environment</td>
<td>Review the parameters that calculate the degree of environmental complexity of the project.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Complexity Factor</td>
<td>project, 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 Environment Complexity Factors tab of the Metrics and Estimation Types dialog. The final project estimate is directly proportional to the ECF.</td>
</tr>
<tr>
<td>Unadjusted Use Case Points (UUCP)</td>
<td>Check the sum of the Use Case complexity numbers.</td>
</tr>
<tr>
<td>Ave Hours per Use Case</td>
<td>Check the average of the number of hours assigned to easy, medium and difficult Use Cases; for information purposes only.</td>
</tr>
<tr>
<td>Total Estimate</td>
<td>Review the detailed breakdown of the final figure. Note that you must 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.</td>
</tr>
<tr>
<td>Default Rate</td>
<td>Set the default hours fed into the final calculation.</td>
</tr>
<tr>
<td>Re-Calculate</td>
<td>Re-run the estimate, usually after you change the hours or Use Case point number.</td>
</tr>
<tr>
<td>Report</td>
<td>Produce a rich text formatted report from the current estimate.</td>
</tr>
</tbody>
</table>
8.1.4 Default Hours

Set the default hour rate per adjusted Use Case point using the Default Hour Rate tab of the Estimation Factors dialog. To access this tab:

- Click on the Default Rate button on the Use Case Metrics (displays the tab as the only tab of the Settings dialog), or
- Select the Settings | Estimation Factors menu option and click on the Default Hour Rate tab.

Type values in the Duration and Hourly Rate fields; click on the OK button to save the current values.

Notes:

- The values you enter are stored as local settings on your computer only.
- This option is also active in the 'Lite', read-only version of Enterprise Architect.

Setting an hourly rate is the most difficult factor in an accurate estimation. Typical ranges can vary from 10 to 30 hours per Use Case point. Studying the Use Case Points Method, from which this variable is defined, can help you to understand its role in the estimation and facilitate selection of a suitable initial value. The best way to estimate this value is through analysis of previous completed projects. By calculating the project estimation on a completed project for which the Use Cases and environment are configured within Enterprise Architect, you can adjust the hour rate to render an appropriate value for your unique work environment.
8.2 Resource Management

What is a Resource?
Resources are the people who work on a project. They can be assigned roles and allocated tasks, which enables 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 this window, select the View | Project Management menu option, or press [Ctrl]+[Shift]+[7]. (If the window does not display as shown, click on the Show/Hide Properties button in the toolbar.)

What to Do?
To find out more information about Project Resource Management tasks, use the following guide:

- To allocate a resource to an element, see the Resource Allocation topic
- To record additional project management information for an element, see:
  - the Effort Management topic (record effort expended on the element)
  - the Risk Management topic (record risk associated with the element)
  - the Metrics topic (record metrics measured for an element)
- To obtain a report of resource allocation details, see the Resource Report topic
- To configure Project Management data and populate the drop-down lists used on the Project Management dialog tabs, see the following topics:
  - Roles
  - Clients
  - Effort Types
  - Metric Types
  - Risk Types
- To find out about the functions of the Project Management toolbar, see the Project Management Window topic.

Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Manage Project Information permission to update and manage project resources, effort, metrics and risks.
8.2.1 Resource Allocation

Enterprise Architect enables you to connect 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).

To enter Resource Allocation details for an element, select the element and select the View | Project Management menu option. The Project Management window displays, showing the Resource Allocation tab. Click on the New button on the Project Management window toolbar. (If the window does not display as shown, click on the Show/Hide Properties button on the toolbar.)

This tab enables you to enter the following data:
- The name of the resource (click on the drop-down arrow and select, or type the name in)
- The role of the resource (click on the drop-down arrow and select, or type the name in)
- The start and end date for the resource availability
- The time allocated to the resource
- The percentage of the task the resource has completed
- The expected time allocated to the resource
- The actual time expended by the resource
- A description of the work being done by the resource
- Notes on the activity history of the resource.

To edit existing items, click on the required item in the list on the left of the window.
8.2.2 Effort Management

To enter effort details for an element, follow the steps below:

1. Select the element.
2. Select the View | Project Management menu option. The Project Management window displays, showing the Resource Allocation tab.
3. Click on the Effort tab.
4. Click on the New button on the Project Management window toolbar. (If the window does not display as shown, click on the Show/Hide Properties button on the toolbar.)

The Effort tab enables you to enter the following data:

- A name for the effort (short description)
- The type of effort (click on the drop-down arrow and select, or type the name in; typed names are not added to the global effort type list)
- The time the effort will expend
- Some notes on the effort.

To edit an existing item, click on the required item in the list on the left of the window.

Notes:

- The drop-down arrow on the Type field displays a list of effort types as defined on the Effort tab of the Metric and Estimation Types dialog. If required, you can type in alternative effort types, but these are not added to the drop-down list of defined types.

- Although Enterprise Architect 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;
8.2.3 Risk Management

To enter risk details for an element, follow the steps below:

1. Select the element.
2. Select the View | Project Management menu option. The Project Management window displays, showing the Resource Allocation tab.
3. Click on the Risks tab.
4. Click on the New button on the Project Management window toolbar. (If the window does not display as shown, click on the Show/Hide Properties button on the toolbar.)

The Risks tab enables you to enter the following data:

- A name for the risk (short description)
- The type of risk (click on the drop-down arrow and select, or type the name in; typed names are not added to the global risk type list)
- A weighting for the risk
- Some notes on the risk.

To edit an existing item, click on the required item in the list on the left of the window.

**Note:** Although Enterprise Architect does not currently provide detailed reports on risks within a model, you can use the Automation Interface or similar tools to create your own custom reports based on risk information you enter;
8.2.4 Metrics

To enter metrics for an element, follow the steps below:

1. Select the element.
2. Select the View | Project Management menu option. The Project Management window displays, showing the Resource Allocation tab.
3. Click on the Metrics tab.
4. Click on the New button on the Project Management window toolbar. (If the window does not display as shown, click on the Show/Hide Properties button on the toolbar.)

The Metrics tab enables you to enter the following data:

- A name for the metric (short description)
- The type of metric (click on the drop-down arrow and select, or type the name in; type names are not added to the global metric type list)
- A weighting for the metric
- Some notes on the metric.

To edit an existing item, click on the required item in the list on the left of the window.

**Note:**
Although Enterprise Architect does not currently provide detailed reports on metrics within a model, you can use the Automation Interface or similar tools to create your own custom reports based on metric information you enter.
8.2.5 Resource Report

To generate a resource report on a package, either:

- In the Project Browser, right-click on the package to create a report for and, from the context menu, select the Documentation | Resource Allocation option, or
- If the diagram currently active belongs to the package to create a report for, select the Project | Documentation | Resource and Tasking Details menu option.

The Resource and Tasking Details dialog displays a list of all elements that have resources allocated to them. The result list includes the resource allocated, the start and end dates, the percentage complete and other relevant information. You can print out the results if required.

### Option

<table>
<thead>
<tr>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Package</td>
</tr>
<tr>
<td>Resource</td>
</tr>
<tr>
<td>As At Date</td>
</tr>
<tr>
<td>Cut Off</td>
</tr>
<tr>
<td>Show Where</td>
</tr>
<tr>
<td>Refresh</td>
</tr>
<tr>
<td>Locate Object</td>
</tr>
<tr>
<td>Print</td>
</tr>
<tr>
<td>Resourcing Details</td>
</tr>
</tbody>
</table>
8.2.6 Effort Types

You can specify the effort types used when assigning effort to an element in Enterprise Architect, using the Effort tab of the Project Indicators dialog. Creating an effort type using this dialog adds to a global list of effort types that can be added to any element in the model. This list of types displays in the Type field drop-down list on the Effort tab of the Project Management window.

To open the Project Indicators dialog, select the Settings | Project Indicators menu option. Click on the Effort tab.

To create a new effort type, click on the New button, or to edit an existing effort type, click on the effort type name in the Defined Effort Types list. Complete the fields as follows:

- In the Effort field type the name of the effort type
- In the Description field type a short description of the effort type
- In the Weight field type the weighting to apply to the effort type
- In the Note field, type any additional information on the effort type
- Click on the Save button.

Notes:

- Although Enterprise Architect 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;
- You can transport effort types between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
8.2.7 Metric Types

You can specify the metric types used when assigning metrics to an element in Enterprise Architect, using the Metric tab of the Project Indicators dialog. Creating a metric using this dialog creates a global list of metrics that can be added to any element in the model. You can define a metric on other screens, such as the Metrics tab of the Project Management window, but such metrics are not added to the global list.

Select the Settings | Project Indicators menu option. On the Project Indicators dialog, click on the Metric tab.

To create a new metric type, click on the New button, or to edit an existing metric type, click on the metric typename in the Defined Metrics list. Complete the fields as follows:

- In the Metric Type field type the name of the metric type
- In the Description field type a short description of the metric type
- In the Weight field type the weighting to apply to the metric type
- In the Note field, type any additional information on the metric type
- Click on the Save button.

Notes:

- Although Enterprise Architect does not currently provide detailed reports on metrics within a model, you can use the Automation Interface or similar tools to create your own custom reports based on metrics information you enter;
- You can transport metric types between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

Enterprise Architect User Guide
8.2.8 Risk Types

You can specify the risk types used when assigning risk to an element in Enterprise Architect, using the Risk tab of the Project Indicators dialog. Creating a risk type using this dialog creates a global list of risk types that can be added to any element in the model. You can define a risk type on other screens, such as the Risks tab of the Project Management window, but such risks are not added to the global list.

Select the Settings | Project Indicators menu option. On the Project Indicators dialog, click on the Risk tab.

To create a new risk type, click on the New button, or to edit an existing risk type, click on the risk type name in the Defined Risks list. Complete the fields as follows:

- In the Risk Type field, type the name of the risk type
- In the Description field, type a short description of the risk type
- In the Weight field, type the weighting to apply to the risk type
- In the Note field, type any additional information on the risk type
- Click on the Save button.

Notes:

- Although Enterprise Architect does not currently provide detailed reports on risks within a model, you can use the Automation Interface or similar tools to create your own custom reports based on risk information you enter;
- You can transport risk types between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
8.3 Testing

Introduction to Testing

In addition to the integrated JUnit and NUnit testing capabilities, Enterprise Architect enables you to attach arbitrarily complex tests to any model element. Keeping the model elements and the testing documentation in one integrated model significantly improves the communication between the test-team and the software developers and architects. The detailed search facilities make it easy to find failing test cases, test cases not run and tests cases that have been passed. Using the testing and search capabilities, it is easy to navigate through the model and quickly locate problem spots, design flaws and other critical issues. Enterprise Architect is not only a UML Modeling environment, it is also a complete Test Management environment.

Basic Tasks

Simple tasks that you might perform include:

- Open the Testing Workspace
- Use the Test Details Dialog

Categories

Typically you 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 requirements
- Acceptance tests to test user satisfaction, and
- Scenario tests to test the end-to-end suitability and functionality of the application.

Using Tests

Other tasks that you might perform when working with tests include:

- Import Scenario as Test
- Import Test from Other Elements
- Generate Test Details Report
- Show Test Script Compartments
- Create Test Documentation

Note:

Most of the tasks identified above relate to a tests for a single element. You can make a set of tests available to a number of elements by performing the above tasks on a Test Case element and then associating that Test Case with each of the other elements. The Test Case element also helps to make tests more visible in diagrams, the Project Browser, windows and searches.
8.3.1 The Testing Workspace

The Testing window, or Workspace, provides a quick and convenient method of working with element tests. When you select an element in a diagram or in the Project Browser window, if the Testing window is visible the lists of tests for that element are loaded ready for modification or addition.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have Manage Tests permission to update and delete test records.

To open the Testing window, select the View | Testing menu option. Alternatively, press [Alt]+[3]. (If the window does not display as shown, click on the Show/Hide Properties button in the toolbar.)

This window can be docked to the application workspace.

Click on an existing item to edit the details, or click on the New icon in the Testing window toolbar to add further items. Alternatively, use the Test details dialog.

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**
8.3.2 The Test Details Dialog

The Test details dialog opens from the Testing window in list mode. (The Testing window displays as shown below in list mode. If it does not display like this, click on the Show/Hide Properties icon in the window toolbar.)

Double-click on an existing test case or click on the New icon in the window toolbar. The Test Details dialog displays.

Notes:
- Add multiple test cases in one batch by using the New and Apply buttons.
- You can format the text in the Description field using the Rich Text Notes toolbar at the top of the field.
8.3.3 **Unit Testing**

Use Unit Testing to test Classes, Components and other elements as programmers build them.

The Unit testing tab displays in the Testing window by default. To open the Testing window, select the View | Testing menu option. Open a diagram and select the required element; all of the test scripts for that element display in the Testing window.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Specify the name of the test.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify the current status of the test.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of test.</td>
</tr>
<tr>
<td>Run By</td>
<td>Specify the name of the person who ran the test.</td>
</tr>
<tr>
<td>Checked By</td>
<td>Specify the name of the person who checked the test run.</td>
</tr>
<tr>
<td>Last Run</td>
<td>Select the date on which the test was last run.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a description of the test. You can format the text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Input</td>
<td>Type in the input data.</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Type the acceptance or test success conditions.</td>
</tr>
<tr>
<td>Results</td>
<td>Type the results of the last test.</td>
</tr>
</tbody>
</table>
### 8.3.4 Integration Testing

Use Integration Testing to test how the constructed components work together.

To display Integration Testing details select the View | Testing menu option to display the Testing window. Open a diagram and select the required element; all of the test scripts for that element display in the Testing window. Click on the Integration tab.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Specify the name of the test.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify the current status of the test.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of test.</td>
</tr>
<tr>
<td>Run By</td>
<td>Specify the name of the person who ran the test.</td>
</tr>
<tr>
<td>Checked By</td>
<td>Specify the name of the person who checked the test run.</td>
</tr>
<tr>
<td>Last Run</td>
<td>Select the date on which the test was last run.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a description of the test. You can format the text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Input</td>
<td>Type in the input data.</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Type the acceptance or test success conditions.</td>
</tr>
<tr>
<td>Results</td>
<td>Type the results of the last test.</td>
</tr>
</tbody>
</table>
### 8.3.5 System Testing

Use System Testing to test that the system performs the right business functions correctly.

To display System Testing details select the View | Testing menu option to display the Testing window. Open a diagram and select the required element; all of the test scripts for that element display in the Testing window. Click on the System tab.

#### Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Specify the name of the test.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify the current status of the test.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of test.</td>
</tr>
<tr>
<td>Run By</td>
<td>Specify the name of the person who ran the test.</td>
</tr>
<tr>
<td>Checked By</td>
<td>Specify the name of the person who checked the test run.</td>
</tr>
<tr>
<td>Last Run</td>
<td>Select the date on which the test was last run.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a description of the test. You can format the text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Input</td>
<td>Type in the input data.</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Type the acceptance or test success conditions.</td>
</tr>
<tr>
<td>Results</td>
<td>Type the results of the last test.</td>
</tr>
</tbody>
</table>

---

Operator can access customer information.
## 8.3.6 Acceptance Testing

Use Acceptance Testing to ensure that users are satisfied with the system.

To display Acceptance Testing details select the **View | Testing** menu option to display the **Testing** window. Open a diagram and select the required element; all of the test scripts for that element display in the **Testing** window. Click on the **Acceptance** tab.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Specify the name of the test.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify the current status of the test.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of test.</td>
</tr>
<tr>
<td>Run By</td>
<td>Specify the name of the person who ran the test.</td>
</tr>
<tr>
<td>Checked By</td>
<td>Specify the name of the person who checked the test run.</td>
</tr>
<tr>
<td>Last Run</td>
<td>Select the date on which the test was last run.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a description of the test. You can format the text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Input</td>
<td>Type in the input data.</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Type the acceptance or test success conditions.</td>
</tr>
<tr>
<td>Results</td>
<td>Type the results of the last test.</td>
</tr>
</tbody>
</table>

Users can access customer data as and when required under busy business conditions.
8.3.7 **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 Scenario Testing details select the View | Testing menu option to display the Testing window. Open a diagram and select the required element; all of the test scripts for that element display in the Testing window. Click on the Scenario tab.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Specify the name of the test.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify the current status of the test.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of test.</td>
</tr>
<tr>
<td>Run By</td>
<td>Specify the name of the person who ran the test.</td>
</tr>
<tr>
<td>Checked By</td>
<td>Specify the name of the person who checked the test run.</td>
</tr>
<tr>
<td>Last Run</td>
<td>Select the date on which the test was last run.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a description of the test. You can format the text using the Rich Text Notes toolbar at the top of the field.</td>
</tr>
<tr>
<td>Input</td>
<td>Type in the input data.</td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td>Type the acceptance or test success conditions.</td>
</tr>
<tr>
<td>Results</td>
<td>Type the results of the last test.</td>
</tr>
</tbody>
</table>
8.3.8 Import Scenario as Test

You can import a scenario from a Use Case or other element into the Test Scenarios list, or from all elements in a package. This avoids having to duplicate the scenario information manually.

Import Element Scenarios

To import one or more scenarios from a specific element, follow the steps below:

1. Select the View | Testing menu option to display the Testing window. Open a diagram and select the required element; all of the test scripts for that element display in the Testing window. Click on the Scenario tab, at the bottom of the window.
2. Right-click on the list of tests to display the context menu, and select the Import element scenario(s) menu option. The Import Scenario dialog displays.

3. Select the scenarios to import from the Select items to import list. You can import scenarios from any element in the model by clicking on the Select element drop-down arrow and selecting the required element.
4. Click on the OK button to import the selected scenario(s).

The Import Scenario dialog has the following additional options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show related elements only</td>
<td>Filter selection to apply only to related elements.</td>
</tr>
<tr>
<td>Limit selection to these Object Types only</td>
<td>Type in specific element types, separated by commas, to filter for only those element types.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Refresh the list of available scenarios.</td>
</tr>
</tbody>
</table>

Import Package Scenarios

To import scenarios from all elements in a package, follow the steps below:

1. Select the View | Testing menu option to display the Testing window. Open a diagram and select the parent package element or an element in the package; all of the test scripts for that element display
in the Testing window. Click on the Scenario tab, at the bottom of the window.

2. Right-click on the list of tests to display the context menu, and select the Import Package Scenario(s) menu option. The Import Scenario dialog displays.

   This version of the Import Scenario dialog lists all scenarios against all elements in the package. It does not enable you to select a specific element, but does enable you to filter the list of scenarios to those from specific types of element.

3. In the Limit selection to these Object Types only field, type a comma-separated list of the object types for which to show scenarios. Click on the Refresh button.

4. Click on the OK button to import the scenarios from each element as test scenarios for that element.
8.3.9 Import Test From Other Elements

You can import any test from a Use Case or other element into the Testing window. This avoids having to duplicate the test information manually.

Import a Test

To import a test, follow the steps below:

1. Select the View | Testing menu option to display the Testing window. Open a diagram and select the required element; all of the test scripts for that element display in the Testing window.

2. Right-click on the list of test cases to display the context menu, and select the Import Tests from Other Element menu option. The Import Element Tests dialog displays.

3. Select the test to import from the Select items to import list. You can import tests from any element in the model by clicking on the Select element drop-down arrow and selecting the required element.

4. Click on the OK button to import the selected test(s).

The Import Element Tests dialog has the following additional options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show related elements only</td>
<td>Filter selection to apply only to related elements.</td>
</tr>
<tr>
<td>Limit Selection to these Object Types only</td>
<td>Type in specific element types, separated by commas, to filter for only those element types.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Refresh available options.</td>
</tr>
</tbody>
</table>
8.3.10 Testing Details Report

You can view the Testing Details dialog for a package, which enables 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 Testing Details dialog, right-click on a package in the Project Browser to display the context menu, and select the Documentation | Testing Details menu option.

The Testing Details dialog includes the following options:

<table>
<thead>
<tr>
<th>Options</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run By</td>
<td>Select a name to filter for tests run by that person. Click on the x button to clear the field.</td>
</tr>
<tr>
<td>Checked By</td>
<td>Select a name to filter for tests checked by that person. Click on the x button to clear the field.</td>
</tr>
<tr>
<td>Test Type</td>
<td>Select the radio button for the required test type.</td>
</tr>
<tr>
<td>Status</td>
<td>Select the radio button for the required status.</td>
</tr>
<tr>
<td>Locate Object</td>
<td>(After clicking on an element in the Test Details list) locate the element in the Project Browser.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Re-run the report query.</td>
</tr>
<tr>
<td>Print</td>
<td>Print a summary of the test results.</td>
</tr>
</tbody>
</table>
8.3.11 Show Test Script Compartments

Any element that is capable of displaying a compartment can be used to show test scripts in a diagram. To make use of the feature the element must have an attached test. To use this feature follow the steps below:

1. Open a diagram containing the element with the attached test items.
2. Double-click on the diagram background to display the Diagram Properties dialog. Click on the Elements tab.

3. In the Show Compartments panel, select the Testing checkbox.
4. Click on the OK button to save the setting.

The tests now appear as an item in the test scripts compartment of the diagram element.
8.3.12 Test Documentation

Enterprise Architect enables you to output the test scripts and results you have entered against elements in the model, in Rich Text format. For more information on entering test scripts and details see the previous sections of the Testing topic.

To create the documentation, right-click on a package in the Project Browser and select the Documentation | Testing Report context menu option. The Create Test Documentation dialog displays.

You can also access the Create Test Documentation dialog by selecting the Project | Documentation | Testing Report menu option.

The Create Test Documentation dialog enables 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.
8.4 Maintenance

Maintenance Elements

Maintenance elements are defects, changes, issues and tasks. They all apply to individual model elements and can 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** records a risk 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, defects and issues that apply to the whole system, see the Changes and Defects topic; for tasks that apply to the whole system, see the Project Tasks topic.

The following topics explain how to create and edit Maintenance elements:

- **The Maintenance Workspace** - describes the Maintenance window
- **Maintenance Element Properties** - describes how to complete the Properties dialog for the various maintenance elements
- **Show Maintenance Script in Diagram** - describes how to display maintenance elements on diagrams.
### 8.4.1 The Maintenance Workspace

Enterprise Architect 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 changes, issues defects and 'to do' items associated with a particular model element. Access this window by selecting the View | Maintenance menu option, or by pressing [Alt]+[4].

Click on the required tab - Element Defects, Element Changes, Element Issues and Element Tasks - and 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 Enterprise Architect. The RTF Setup dialog has checkboxes to show or hide element defects, changes, issues and tasks.

<table>
<thead>
<tr>
<th>Defect</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current thread model is broken</td>
<td>High</td>
</tr>
</tbody>
</table>

Using the toolbar, you can add or delete items and show or hide the Properties window to enable you to edit each item in the list. You can also display the Maintenance items in a compartment of each appropriate element in a diagram.
8.4.2 Maintenance Element Properties

Notes:

- This topic describes the Element Defects tab of the Maintenance window. The Element Changes, Element Issues and Element Tasks tabs differ only in minor details.
- For information on element-level Defects, Issues, Changes and Tasks, see the Maintenance topic. For information on the Maintenance window, see the Maintenance Workspace topic.

Element defect details are recorded via the Element Defects tab. To access this tab, follow the steps below:

1. Select the View | Maintenance menu option. The Maintenance window displays. (If the window does not have the format shown below, click on the Show/Hide Properties button in the Maintenance window toolbar.)

2. Open a diagram and select an element. All of the maintenance entries for that element are shown in the Maintenance window, under the various tabbed sections.

3. Click on the Element Defects tab.

4. To:
   - Add a new item, click on a blank line in the Defect list and complete the fields as described in the table below
   - Modify an existing item, click on that item in the Defect List and edit the fields as described in the table below
   - Delete an existing item, right-click on the item in the Defect list and select the Delete context menu option.

5. Click on the Save button in the window toolbar.

Complete or edit the following fields on the Maintenance window:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect</td>
<td>Display a list of recorded defects associated with the element.</td>
</tr>
<tr>
<td>Name</td>
<td>Type the name or a short description of the defect.</td>
</tr>
<tr>
<td>Reported by</td>
<td>Select the name of the person who reported the defect.</td>
</tr>
<tr>
<td>Reported</td>
<td>Select the date on which the defect was reported.</td>
</tr>
<tr>
<td>Status</td>
<td>Select the defect status, such as Complete or Approved.</td>
</tr>
<tr>
<td>Resolved by</td>
<td>Select the name of the person who fixed the defect.</td>
</tr>
<tr>
<td>Resolved</td>
<td>Select the date on which the defect was resolved.</td>
</tr>
<tr>
<td>Priority</td>
<td>Select the priority assigned to resolving the defect.</td>
</tr>
<tr>
<td>Version</td>
<td>Type the version number associated with this fix.</td>
</tr>
<tr>
<td>Description</td>
<td>Type a longer description of the defect.</td>
</tr>
<tr>
<td>History</td>
<td>Enter any notes or references to previous occurrences of this defect.</td>
</tr>
</tbody>
</table>
8.4.3 Show Maintenance Script in Diagram

Any element that is capable of displaying a compartment can show maintenance scripts in a diagram. To make use of the feature the element must have an attached maintenance item.

To use this feature follow the steps below:

1. Open a diagram containing the element with the attached maintenance items.
2. Double-click on the diagram background to display the Diagram Properties dialog. Click on the Elements tab.

3. In the Show Compartments panel, select the Maintenance checkbox.
4. Click on the OK button to save the setting.

The Maintenance Items now appear as items in the maintenance scripts compartment of the diagram element.
8.5 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 Defect elements can be created in UML diagrams and connected using Realization, Dependency, Aggregation and other relationships to show what model element each affects and how each is resolved. You can edit the element properties and assign people (as Actor elements) to changes and defects.
8.5.1 Defects (Issues)

A Defect (or Issue) element is a structured comment that contains information about defects and issues that relate to the system or model. This corresponds in some sense to a failure to meet defined requirements for the current system. An Issue element looks the same as a Requirement element:

Enterprise Architect enables you to generate and handle issues in much the same way as you can handle and color code Requirements. See the Requirements Management topic 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.

Note:

Issue elements can be created with or without an identifying I in the top right corner of the element. To toggle the display of this letter, select or deselect the Show stereotype icon for requirements checkbox on the Options dialog, Objects page.

Add an Issue Using the Enterprise Architect UML Toolbox

To add an Issue to the model using the Enterprise Architect UML Toolbox:

1. Open a Custom diagram.
2. From the Custom pages or Common page of the Enterprise Architect UML Toolbox, drag the Issue icon onto the diagram.
3. Enter the 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, follow the steps below:

1. Right-click on a package in the Project Browser.
2. Select the Insert | New Element menu option. The New Element dialog displays.
3. In the Type field, click on the drop-down arrow and select Issue.
4. In the Name field, type a name for the element.
5. Click on the OK button.
8.5.2 Changes

A Change element is a structured comment that contains information about requested changes to the system/model. This corresponds in some sense to a change in requirements for the current system. A Change element looks the same as a Requirement element:

Enterprise Architect enables you to generate and handle Changes in much the same way as you can handle and color code Requirements. See the Requirements Management topic for more information.

You can connect Changes using Realization connectors to model elements that implement the Change, and you can structure a hierarchy of changes using Aggregation.

**Note:**
Change elements can be created with or without an identifying C in the top right corner of the element. To toggle the display of this letter, select or deselect the Show stereotype icon for requirements checkbox on the Options dialog, Objects page.

Add a Change Using the Enterprise Architect UML Toolbox

To add a Change to the model using the Enterprise Architect UML Toolbox:

1. Open a Custom diagram.
2. From the Custom pages or Common page of the Enterprise Architect UML Toolbox, drag the Change icon onto the diagram.
3. Enter the details as required.

Add a Change Using the Insert New Element Dialog

To add a Change to the model using the Insert New Element dialog, follow the steps below:

1. Right-click on a package in the Project Browser.
2. Select the Insert | New Element menu option. The New Element dialog displays.
3. In the Type field, click on the drop-down arrow and select Change.
4. In the Name field, type a name for the element.
5. Click on the OK button.
8.5.3 Element Properties

The Properties 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 (such as owner and dates). You can also associate files with the issue and add Tagged Values.
8.5.4 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 of a staff member to work on or correct a named issue. This same approach can be used for any mix of model elements.
8.6 Project Tasks List

The Project Tasks List is a convenient 'To Do' list of major project work items that are not recorded elsewhere. It can also be used to track things like requests or meetings.

The Project Tasks List is available as a tab on the System window. To open the System window, select the View | System menu option, or press [Alt]+[2]. Select the Project Tasks tab.

Right-click on the list to view the context menu, and select to add, modify or delete list items, or to set a status filter. To set the sort order, click the title-bar of the column on which to index the tasks.

For more information see the Add, Modify and Delete Tasks topic.

Tip:
Select the Print List menu option to print out the currently displayed items.

Note:
You can transport these task definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
8.6.1 Add, Modify and Delete Tasks

From the Project Tasks tab on the System window, display 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 a blank area of the Project Tasks tab, or right-click and select the Add New context menu option. The Task Detail dialog displays.

2. Enter the details for the task. You can define the following:
   - The task name
   - Auto counters - if you have configured these, click on the Auto button
   - 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 and the actual time expended
   - The percent complete
   - The phase associated with this task.

3. Click on the Apply button.

4. To create another entry, click on the New button, or to close, click on the OK button.

Modify a Task
To modify a task, on the Project Tasks tab, either:

- Double-click on the task to modify, or
• Right-click on the task to modify and, from the context menu, select the Modify Selected menu option. The Task Detail dialog displays, and you can edit the task data.

Delete a Task
To delete a task, follow the steps below:
1. On the Project Tasks tab, right-click on the task to delete. The context menu displays.
2. Select the Delete menu option.
8.7 Project and Model Issues

Any identified issues can be recorded against the current project. Issues are raised with a description, date, owner and status.

Note:
You can transport these issue definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

You can add, delete and modify issues using either the Project Issues dialog, or the Issue Detail dialog from the Project Issues tab of the System window. You can also generate and view an RTF report of your issue list, using either the Project Issues dialog or the Project Issues tab.

Tip:
You can view sample report output in the Report Output Sample topic.
8.7.1 Project Issues Dialog

The Project Issues dialog is accessed from the Project | Documentation | Issues menu option. This dialog enables you to record a description, date, owner and status of any identified issues against the current project. You can add, modify and delete issues, and generate a report of your project issues in Rich Text Format.

### Details

- **Issue:**
- **Priority:** Low
- **Date:** 20/03/2007
- **Status:** Open
- **Owner:**
- **Desc.:**

### Resolution

- **Resolver:**
- **Date:** 20/03/2007
- **Close Issue**
- **Comments:**

### Project Issues & Discussion

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Owner</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>The test servers will be delayed</td>
<td>13/10/2005</td>
<td>Frank Mclver</td>
<td>Under Review</td>
</tr>
<tr>
<td>Pre-production Environment Model</td>
<td>12/08/2005</td>
<td>Frank Mclver</td>
<td>Open</td>
</tr>
<tr>
<td>Missing Training material</td>
<td>13/06/2005</td>
<td>Frank Mclver</td>
<td>Open</td>
</tr>
<tr>
<td>Compiler Version disparity</td>
<td>13/06/2005</td>
<td>Frank Mclver</td>
<td>Under Review</td>
</tr>
<tr>
<td>Public Holidays</td>
<td>13/06/2005</td>
<td>Frank Mclver</td>
<td>Open</td>
</tr>
</tbody>
</table>
8.7.2  Project Issues Tab

The Project Issues tab in the System window enables any identified issues to be recorded against the current project. Issues are raised with a description, date, owner and status.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have Manage Issues permission to update and delete Issues records.

To access this tab, select the View | System menu option or press [Alt]+[2] to display the System window, and click on the Project Issues tab.

**Tip:**

You can right-click on the list and select the Print List context menu option to print out the currently displayed items.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Issue</th>
<th>Date</th>
<th>Status</th>
<th>Owner</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Missing Training material</td>
<td>13/06/2005</td>
<td>Open</td>
<td></td>
<td>A number of the train...</td>
</tr>
<tr>
<td>Medium</td>
<td>Compiler Version disparity</td>
<td>13/06/2005</td>
<td>Under Res...</td>
<td>A number of the de...</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Public Holidays</td>
<td>13/06/2005</td>
<td>Open</td>
<td>Frank M...</td>
<td>The schedule inclu...</td>
</tr>
<tr>
<td>High</td>
<td>The test servers will be...</td>
<td>13/10/2005</td>
<td>Under Res...</td>
<td>Frank M...</td>
<td>The test server bul...</td>
</tr>
<tr>
<td>Low</td>
<td>Pre-production Environment</td>
<td>12/08/2005</td>
<td>Open</td>
<td>Frank M...</td>
<td>This model is yet to...</td>
</tr>
</tbody>
</table>

To add a new issue, double-click on an empty row of the Project Issues tab. To modify an issue, double-click on the required item in the list. In each case, the Issue Detail dialog displays.
You can also delete an issue and generate a report of your issues in Rich Text Format.
8.7.3 Add, Delete and Modify Issues

Issues can be added, deleted and modified using either the Project Issues dialog, or the Issue Detail dialog from the Project Issues tab of the System window.

To add an issue, click on the New button and complete the following fields:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue</td>
<td>The name of the issue.</td>
</tr>
<tr>
<td>Auto</td>
<td>Click on the Auto button if you have auto counters configured.</td>
</tr>
<tr>
<td>Priority</td>
<td>The priority of this issue: low, medium or high.</td>
</tr>
<tr>
<td>Date</td>
<td>The date the issue arose.</td>
</tr>
<tr>
<td>Status</td>
<td>The issue's current status.</td>
</tr>
<tr>
<td>Owner</td>
<td>The person owning the issue.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the issue.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Notes on the resolution of the issue.</td>
</tr>
<tr>
<td>Date</td>
<td>The date the issue was resolved.</td>
</tr>
<tr>
<td>Resolved By</td>
<td>Person who resolved the issue.</td>
</tr>
<tr>
<td>Comments</td>
<td>Any comments regarding the resolution of the issue.</td>
</tr>
<tr>
<td>Close Issue</td>
<td>Click on this button to close the issue.</td>
</tr>
<tr>
<td>Apply</td>
<td>Save and apply the issue.</td>
</tr>
</tbody>
</table>

To modify an issue, double-click on it in the Project Issues tab or Project Issues & Discussion list, then edit the fields as indicated in the above table.

To delete an issue, click on it in the Project Issues tab or Project Issues & Discussion list, then:

- Click on the Delete button (Project Issues dialog) or
- Right-click on the issue and select the Delete option from the context menu.
8.7.4 Report From Project Issues Dialog

To generate an RTF document of your issue log using the Project Issues dialog, follow the steps below:

1. Select the Project | Documentation | Issues menu option. The Project issues dialog displays.
2. Click on the Report button. The Save As dialog displays.
3. Browse for the appropriate file location and, in the File name field, type the file name for the report.
4. Click on the Save button.
5. To view the report, click on the View RTF button.

Tip:
For information on viewing sample report output, see the Report Output Sample topic.
8.7.5 Report From Project Issues Tab

To generate an RTF document of your issue log using the Project Issues tab of the System window, follow the steps below:

1. Select the View | System menu option, or press [Alt]+[2]. The System window displays.
2. Click on the Project Issues tab.
3. Right-click on a blank line of the Project Issues tab and select the Create RTF Report context menu option. The Save As dialog displays.
4. Enter the directory location and file name to save your report to and click on the Save button. Enterprise Architect generates the report. This should only take a few moments to complete.

Tip:
For information on viewing sample report output, see the Report Output Sample topic.
### 8.7.6 Report Output Sample

An example of the output from an Issues report is shown below:

**List of Project Issues: 24-Jul-2007 9:47:00 AM**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date/Owner</th>
<th>Description</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test servers will be delayed</td>
<td>24/07/2007 Elosie Norman</td>
<td>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.</td>
<td>Closed: 24/07/2007 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.</td>
</tr>
<tr>
<td>Public Holidays</td>
<td>24/07/2007 Joanna Stoat</td>
<td>The schedule includes staff working on public holidays. A number of staff have indicated that contrary to what they stated earlier they are not available.</td>
<td>Open: 24/07/2007</td>
</tr>
<tr>
<td>Compiler Version disparity</td>
<td>24/07/2007 Elosie Norman</td>
<td>A number of the developers have downloaded different versions of a number of the compilers. This has lead to unpredictable builds impacting on testing.</td>
<td>Under Review: 24/07/2007</td>
</tr>
</tbody>
</table>
8.8 Project Glossary

The glossary enables you to set up a list of defined terms for your project. You can further divide the items by category; for example, Business terms and Technical terms. The glossary can be saved in Rich Text format for inclusion as part of a larger project document.

You can add, delete and modify the project glossary entries through the Glossary dialog or through the Project Glossary tab on the System window.

**Tip:**
Include a Glossary Report in your project requirements or functional specifications documents.

**Note:**
You can transport these glossary definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
8.8.1 The Glossary Dialog

To open the Glossary dialog, select the Project | Documentation | Glossary menu option. Use this dialog to add, modify, and delete glossary entries. You can also limit the display to show only technical or business related entries.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glossary Term</td>
<td>Type the term to include in the glossary.</td>
</tr>
<tr>
<td>Glossary Type</td>
<td>Select either Technical or Business.</td>
</tr>
<tr>
<td>Description</td>
<td>Type the definition or description of the term.</td>
</tr>
<tr>
<td>Limit Display To</td>
<td>Select the appropriate radio button to filter the list of entries on the dialog to show Technical or Functional entries, or both.</td>
</tr>
<tr>
<td>Type</td>
<td>Term</td>
</tr>
<tr>
<td>Business</td>
<td>Accounting Periods</td>
</tr>
<tr>
<td>Technical</td>
<td>Association</td>
</tr>
<tr>
<td>Technical</td>
<td>Class</td>
</tr>
<tr>
<td>Technical</td>
<td>Component Model</td>
</tr>
<tr>
<td>Business</td>
<td>Customer</td>
</tr>
<tr>
<td>Technical</td>
<td>Deployment Architecture</td>
</tr>
<tr>
<td>Technical</td>
<td>Deployment Model</td>
</tr>
<tr>
<td>Technical</td>
<td>Extends Relationships</td>
</tr>
<tr>
<td>Technical</td>
<td>Includes Relationship</td>
</tr>
<tr>
<td>Technical</td>
<td>Use Case</td>
</tr>
</tbody>
</table>

Add a Glossary Entry

To add an entry to the glossary, follow the steps below:

1. Enter the details for the glossary item: the Glossary Term, the Glossary Type and the Description.
2. Click on the Save button.
3. To enter another item, click on the New button.

Modify a Glossary Entry

To modify a glossary entry, follow the steps below:

1. Select the entry to modify from the bottom panel of the dialog. The details of the entry display in the fields in the top half of the window.
2. Change the details as required.
3. Click on the **Save** button.

**Delete a Glossary Entry**
To delete a glossary entry, follow the steps below:
1. Select the entry to delete from the bottom panel of the dialog. The details of the entry display in the fields in the top half of the window.
2. Click on the **Delete** button.

**Limit the Display**
You can select which entry categories are displayed in the list. To:
- 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.
8.8.2 Project Glossary Tab

The Project 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 the View | System menu option or press [Alt]+[2]. Select the Project Glossary tab.

Tip:
To print out the currently displayed items, right-click on the list and select the Print List menu option.

Double-click on an entry and use the Glossary Detail dialog to add, modify and delete glossary entries (as below). Alternatively, select the Project | Documentation | Glossary menu option and use the Glossary dialog.

Tip:
Include a Glossary Report in your project requirements or functional specifications document(s).

Add a Glossary Entry

To add an entry to the glossary, follow the steps below:

1. Double-click on the Project Glossary tab, or right-click on the tab and select the Add New context menu option. The Glossary Detail dialog displays.

2. Enter the details for the glossary item: the Term, Type and Meaning.

3. Click on the Apply button.
4. To create another entry, click on the New button.
5. To close, click on the OK button.

**Modify a Glossary Entry**

To modify a glossary entry, either:

1. Double-click on the entry to modify in the list on the Project Glossary tab, or
2. Right-click on the entry to modify in the list on the Project Glossary tab and select the **Modify Selected** context menu option.

The Glossary Detail window displays; edit the fields as required.

**Delete a Glossary Entry**

To delete a glossary entry, follow the steps below:

1. Right-click on the entry to modify in the list on the Project Glossary tab. The context menu displays.
2. Select the **Delete** menu option.
8.8.3 Generate a Report

To generate a report of your model's glossary, follow the steps below:

3. Enter a filename and a heading for the glossary.
4. Select whether to include all sections, or only the Technical or Business sections.
5. To include page breaks, select the Page break between sections checkbox.
6. Click on the OK button to generate the report.
7. Click on the View button to open the report.

Note:
You can view sample report output in the Glossary Report Output Sample topic.
8.8.4 Glossary Report Output Sample

An example of the output from a Glossary report is shown below:

**Glossary**

**Business Terms**

*Accounting Periods*
A defined period of time whereby performance reports can 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 is 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 might optionally request a list or report at some point in a performing a business Use Case.
### 8.9 Update Package Status

Elements in Enterprise Architect can be assigned a current status, such as *Proposed*, *Validated* or *Mandatory*. Often a complete package structure is updated from one status to another (or released) at the same time. To help facilitate this, Enterprise Architect 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 to update. The context menu displays.
2. Select the **Package Control | Update Package Status** menu option. The *Status Update* dialog displays.

   ![Status Update Dialog](image)

3. 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. Click on the **OK** button. Enterprise Architect updates all required elements to the new status.
8.10 Manage Bookmarks

Bookmarks are small red triangles that display above elements in diagrams when 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 enables searching based on bookmarked elements.

You can bookmark an element manually by pressing [Shift]+[Spacebar].

Bookmark Multiple Elements
You can also bookmark all elements in a folder (and their children) using the Manage Bookmarks dialog. Right-click on the parent package in the Project Browser and select the Bookmarks menu option.

This dialog enables you to automatically bookmark elements that have new changes or defects defined in the Maintenance window, or test scripts defined in the Testing window. This is useful to highlight elements that have additional project information.

You can click on the Clear All button to clear all elements in the current tree of bookmarks. You should reload the project to show the new or cleared bookmarks ([Ctrl]+[Shift]+[F11]).
9 Code Engineering

Code Engineering is a process that includes automated code generation, reverse engineering of source code and synchronization between the source code and model.

Enterprise Architect also enables you to rapidly model, generate - or forward engineer - and reverse engineer:

- **XML Technologies** namely XML Schema (XSD) and Web Service Definition Language (WSDL)
- **Database schema** keys, triggers, constraints, RI and other relational database features, for and from a range of database products.

Code Engineering is available in the Professional and Corporate editions of Enterprise Architect.

**Code Generation**

Enterprise Architect enables you to generate source code from UML model elements, creating a source code equivalent of the Class or Interface element for future elaboration and compilation. In particular you can generate C, C++, C#, Delphi, Java, PHP, Python, ActionScript, 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. You can use the Source Code Viewer to view any source code you are opening.

The Code Template Framework (CTF) enables you to customize the way Enterprise Architect generates source code. It also enables you to generate languages that Enterprise Architect does not specifically support, by helping you define the appropriate code generation templates for that language (this is discussed in SDK for Enterprise Architect).

You can link the facilities of Enterprise Architect to other development environments. The MDG Link for Eclipse and MDG Link for Visual Studio.NET are standalone products that provide an enhanced code engineering functionality between Enterprise Architect and the development environments.

**Reverse Engineering**

Reverse Engineering is the import of existing source code into model elements, mapping the source code structures onto their UML representations. This enables you to examine legacy code and the functionality of code libraries for reuse, or to bring the UML model up to date with the code. You can reverse engineer in the same languages as you perform code generation with Enterprise Architect.

Enterprise Architect is also able to reverse engineer binary files, namely Java .jar files and .NET PE files.

**Note:**

Reverse Engineering of other languages including CORBA IDL is also currently available through the use of the MDG Technologies. See www.sparxsystems.com/resources/MDG tech/.

**Synchronization**

Synchronization is when changes in the model are exported to the source code and changes to source code are imported into the model. This enables you to keep your model and source up to date as the project develops.

**Round-Trip Engineering**

Round trip engineering is a combination of reverse and forward generation of code and includes synchronization between the source code and the model in all but the most trivial of code engineering projects. In order to get the most out of round trip engineering in Enterprise Architect, you should be familiar with the modeling conventions used when generating and reverse engineering the languages you use.
9.1 Reverse Engineering

Reverse Engineering in Enterprise Architect enables you to import existing source code from a variety of code languages into a UML model. Existing source code structures are mapped into their UML representations, for example a Java Class is 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 enables 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 enables user to identify the critical modules contained the code, enabling 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 Enterprise Architect, an existing source of code must be imported into Enterprise Architect which can be a single directory or a directory structure. Several options are available when performing the reverse engineering process. The Source Code Engineering Options topic contains several options that affect 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, simply importing the source into Enterprise Architect does not create an easily understood UML model. When working with a legacy system that is poorly designed it is useful to break down the code into manageable components by examining the code elements individually. 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 this point it is possible to create Use Cases that identify the interaction between the legacy Classes, enabling an overview of the legacy system's operation.

Copyright ownership is an important issue to take into account when undertaking the process of reverse engineering. In some cases, software might have specific limitations that prohibit the process of reverse engineering. It is important that a user address the issue of copyright before beginning the process of reverse engineering code. Situations that typically lend themselves to reverse engineering source code include source code that:

- You have already developed
- Is part of a third-party library that you have obtained permission to use
- Is part of a framework that your organization uses
- Is being developed on a daily basis by your developers.

Enterprise Architect currently supports reverse engineering in the following programming languages:

- ActionScript
- C
- 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.
Notes:

- Reverse Engineering of other languages, including CORBA IDL, is currently available through the use of MDG Technologies from [www.sparxsystmes.co m/resources/mdg_tech/](http://www.sparxsystmes.com/resources/mdg_tech/).

- In the Corporate edition of Enterprise Architect, if security is enabled you must have **Reverse Engineer From DDL And Source Code** permission to reverse engineer source code and synchronize model elements against code.
9.1.1 Import Source Code

To import source code (reverse engineer) follow the steps below:

1. In the Project Browser, select (or add) a diagram into which to import the Classes.
2. Right-click on the diagram background to open the context menu and either:
   - Select the language to import from the Import from source file(s) submenu, or
   - Click on the Import Language drop-down arrow in the Code Generation toolbar and select the Import | Import xxx files menu option, where xxx represents the language to import.
3. From the file browser that appears, select one or more source code files to import.
4. Click on the Open button to start the import process.

As the import proceeds, Enterprise Architect provides progress information. When all files are imported, Enterprise Architect makes a second pass to resolve associations and inheritance relationships between the imported Classes.
Current Action:

Reverse engineer classes:
--importing C:\Temp\Class2.java
Resolve relationships:
--relationships for Class2
---resolve generalizations
Complete
9.1.2 Notes on Source Code Import

Enterprise Architect enables you to import code into your project, in the following programming languages:

- ActionScript
- C
- C#
- C++
- Delphi
- Java
- PHP
- Python
- Visual Basic
- Visual Basic .NET

Enterprise Architect supports most constructs and keywords for each coding language. If there is a particular feature you require support for that you feel is missing, please contact Sparx Systems.

You must select the appropriate type of source file for the language, as the source code to import.

**ActionScript**

Appropriate type of source file: `.as`.

**C**

Appropriate type of source file: `.h` header files and/or `.c` files.

When you select a header file Enterprise Architect automatically searches for the corresponding `.c` implementation file to import based on the options for extension and search path specified in the C options.

Enterprise Architect does not expand macros that have been used; these must be added into the internal list of Language Macros.

**C++**

Appropriate type of source file: `.h` header file.

Enterprise Architect automatically searches for the `.cpp` implementation file based on the extension and search path set in the C++ options. When it finds the implementation file it can use it to resolve parameter names and method notes as necessary.

When importing C++ source code, Enterprise Architect ignores function pointer declarations. To import them into your model you could create a `typedef` to define a function pointer type, then declare function pointers using that type. Function pointers declared in this way are imported as attributes of the function pointer type.

Enterprise Architect does not expand macros that have been used; these must be added into the internal list of Language Macros.

**C#**

Appropriate type of source file: `.cs`.

**Delphi**

Appropriate type of source file: `.pas`.

**Java**

Appropriate type of source file: `.java`.

Enterprise Architect supports the AspectJ language extensions.
Aspects are modeled using Classes with the stereotype `aspect`. These aspects can then contain attributes and methods as for a normal Class. If an `intertype` attribute or operation is required, you can add a tag `className` with the value being the name of the Class it belongs to.

Pointcuts are defined as operations with the stereotype of `pointcut`. These can occur in any Java Class, Interface or aspect. The details of the pointcut are included in the `behavior` field of the method.

Advice is defined as an operation with the stereotype `advice`. The pointcut this advice operates on is in the `behavior` field and acts as part of the method’s unique signature. After advice can also have one of the Tagged Values `returning` or `throwing`.

**PHP**
Appropriate type of source file: `.php`, `.php4`, or `.inc`.

**Python**
Appropriate type of source file: `.py`.

**Visual Basic**
Appropriate type of source file: `.cls` Class file.

**Visual Basic .NET**
Appropriate type of source file: `.vb` Class file.
9.1.3 Import a Directory Structure

You can import from all source files in a complete directory structure. This process enables you to import or synchronize multiple files in a directory tree in one pass. Enterprise Architect creates 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 the target package for the import.
2. From the context menu, select the Code Engineering | Import Source Directory menu option. The Import Directory Structure dialog displays.

3. Select the options you require. You can 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
   - 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
   - Whether to create a package for every directory, namespace or file; this might be restricted depending on the source type selected
   - Whether to Synchronize or Overwrite existing Classes when found (if a model Class is found matching the one in code, Synchronize updates the model Class to include the details from the one in code, which preserves information not represented in code such as the location of Classes in diagrams; Overwrite deletes the model Class and generates a new one from code, which deletes and does not replace the additional information)
   - How to handle Classes not found during the import (Prompt for action enables you to review Classes individually)
   - What is shown on diagrams created by the import.

4. Click on the OK button to start.
9.1.4 Import Binary Module

Enterprise Architect enables you to reverse-engineer certain types of binary modules. To import a binary module, right-click on the target package in the Project Browser and select the Code Engineering | Import Binary Module menu option.

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 creates the necessary packages and diagrams during the import process. Selecting the Do not import private members checkbox excludes private members from libraries from being imported into the model.

When importing .Net files, you can import via reflection or via disassembly, or let Enterprise Architect decide the best method - this might 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 can 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 can only be opened using the disassembler. However, the reflection-based importer is generally much faster.

You can also configure:

- Whether to Synchronize or Overwrite existing Classes when found (if a model Class is found matching the one in the file, Synchronize updates the model Class to include the details from the one in the file, which preserves information not represented in the file such as the location of Classes in diagrams; Overwrite deletes the model Class and generates a new one from the file, which deletes and does not replace the additional information)
- Whether to create a diagram for each package
- What is shown on diagrams created by the import.
9.1.5 **MDG Link and Code Engineering**

MDG Link for Eclipse and MDG Link for Visual Studio.NET are standalone products that provide an enhanced code engineering functionality between Enterprise Architect and the development environments.

The MDG Link programs provide a lightweight bridge between Enterprise Architect 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 can be downloaded from [www.sparxsystems.com/products/mdg_eclipse.html](http://www.sparxsystems.com/products/mdg_eclipse.html) and MDG Link for Visual Studio.NET can be downloaded from [www.sparxsystems.com/products/mdg_vs.html](http://www.sparxsystems.com/products/mdg_vs.html).
9.1.6 Classes Not Found During Import

When reverse synchronizing from your code, there are times when some Classes might be deliberately removed from your source code. Enterprise Architect’s import source directory functionality keeps track of the Classes it expects to synchronize with and, on the Import Directory Structure dialog, provides options for how to handle the Classes that weren’t found. You can select the appropriate action so that, at the end of the import, Enterprise Architect either ignores the missing Classes, automatically deletes them or prompts you to handle them.

If you select the Prompt For Action radio button on the Import Directory Structure dialog, to manually review missing Classes, the following dialog displays:

![Dialog showing options to handle missing Classes]

By default, all Classes are marked for deletion. To keep one or more Classes, select them and click on the Ignore button.
9.1.7 Synchronize Model and Code

In addition to generating and importing code, Enterprise Architect provides the option to synchronize the model and source code, creating a model that represents the latest changes in the source code and vice versa. You can use either the model as the source, or the code as the source.

For example: you generated some source code, but made subsequent changes to the model. When you generate code again, Enterprise Architect adds any new attributes or methods to the existing source code, leaving intact what already exists. This means developers can work on the source code and then generate additional methods as required from the model, without having their code overwritten or destroyed.

Similarly, you might have made changes to a source code file, but the model has detailed notes and characteristics you do not want to lose. By synchronizing from the source code into the model, you import additional attributes and methods but do not change other model elements.

Using the two synchronization methods above, it is simple to keep source code and model elements up to date and synchronized.

**Note:**

In the Corporate edition of Enterprise Architect, if security is enabled you must have Generate Source Code and DDL permission to synchronize source code with model elements.

Synchronize Classes on Forward Generation

When there are features present in the code but not in the model you can use the following buttons during forward synchronization:

**Note:**

These buttons are only available when the On forward synch, prompt to delete code features not in model checkbox is selected in the Options - Attributes and Operations dialog.

- **Delete:** when you click on this button the selected code features are removed from the code.
- **Reassign:** when you click on this button the code elements are reassigned to elements in the model (this is only possible when an appropriate model element is present that is not already defined in the code).
- **Ignore:** when you click on this button the code elements not present in the model are ignored completely.
- **Reset to Default:** when you click on this button the settings for synchronizing during forward generation are set to Ignore, meaning that the elements present in the code but not in the model are ignored completely.

<table>
<thead>
<tr>
<th>Code features not in model</th>
<th>Select All</th>
<th>Clear All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>Action</td>
<td>Type</td>
</tr>
<tr>
<td>ClassA() finalizer() init</td>
<td>&lt;none&gt;</td>
<td>operation</td>
</tr>
<tr>
<td></td>
<td>&lt;none&gt;</td>
<td>operation</td>
</tr>
<tr>
<td></td>
<td>&lt;none&gt;</td>
<td>attribute</td>
</tr>
</tbody>
</table>
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 avoided, 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 Source Code Engineering page of the Options dialog (select the Tools | Options | Source Code Engineering menu option). Set up the defaults to match your required language and preferences. Preferences that you can define include default constructors and destructors, methods for interfaces and the Unicode options for created languages. Languages such as Java support namespaces and can be configured to specify a namespace root. In addition to the default settings for generating code, Enterprise Architect supports the following code languages with their own specific code generation options:

- ActionScript
- C
- C# (for both .NET 1.1 and .NET 2.0)
- C++ (standard, plus .NET managed C++ extensions)
- Delphi
- Java (including Java 1.5, Aspects and Generics)
- PHP
- Python
- Visual Basic
- Visual Basic .NET

The Code Template Framework (CTF) enables you to customize the way Enterprise Architect generates source code and also enables generation of languages that are not specifically supported by Enterprise Architect.

Before generating code, you should also familiarize yourself with the way Enterprise Architect handles local path names. Local path names enable you to substitute tags for directory names (e.g. %SRC% = C:\Source).

When you have completed the design of your Classes, you can generate source code.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have Generate Source Code and DDL permission to generate source code.

Use Live Code Generation

On the Package Build Scripts dialog, you have the option to update your source code instantly as you make changes to your model.

Tasks

When you generate code, you perform one or more of the following tasks:

- Generate a Single Class
- Generate a Group of Classes
- Generate a Package
• Update Package Contents
9.2.1 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 connectors to parents and associations to other Classes that are used. Also add Inheritance connectors 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.

**Generate Code for a Single Class**

1. Open the diagram containing the Class or Interface for which to generate code.
2. Right-click on the required Class or Interface to display the context menu and select the **Generate Code** menu option, or press [F11]. The Generate Code dialog displays, which enables you to control how and where your source code is generated.

3. On the Path field, click on [ ... ] (Browse) and select a path name for your source code to be generated to.
4. In the **Target Language** field, click on the drop-down arrow and select the language to generate; this becomes the permanent option for that Class, so change it back if you are only doing one pass in another language.
5. Click on the Advanced button. The **Source Code Engineering** page of the Options dialog displays.
6. Set any custom options (for this Class alone), then click on the Close button to return to the Generate Code dialog.

7. In the Import(s) / Header(s) fields, type any import statements, #includes or other header information. (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.)

8. Click on the Generate button to create the source code.

9. When complete, click on the View button to see what has been generated. Note that you should set up your default viewer/editor for each language type first. You can also set up the default editor on the DDL page of the Options dialog (Tools | Options | Source Code Engineering | Code Editors).
9.2.2 Generate a Group of Classes

In addition to being able to generate code for an individual Class, you can also select a group of Classes for batch code generation. When you do this, you 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 display the context menu.
3. Select the Code Generation | Generate Selected elements menu option. The Save As dialog displays, on which you specify the file path and name for each code file. Enter this information and click on the Save button.
4. The Batch Generation dialog displays, showing the status of the process as it executes (the process might be too fast to see this dialog).

Note:

If any of the elements selected are not Classes or interfaces the option to generate code is not available.
9.2.3 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 enables you to generate a whole branch of your project model in one step.

**Generate a Package**

To generate a package, follow the steps below:

1. In the Project Browser, right-click on the package to generate code for. The context menu displays.

   ![Generate Package Source Code Dialog]

   - **Root Package**: Package
   - **Synchronize**: Synchronize model and code
   - **Generate**:
     - **Auto Generate Files**: Root Directory
     - **Retain Existing File Paths**
   - **Select Objects to Generate**
     - **Object**: Class4
     - **Type**: Class
     - **Target File**

3. In the **Synchronize** field, click on the drop-down arrow and select the appropriate synchronize option:
   - **Synchronize model and code**: Classes with existing files are forward synchronized with that file; Classes with no existing file are generated to the displayed target file
   - **Overwrite code**: All selected target files are overwritten (forward generated)
   - **Do not generate**: Only selected Classes that do not have an existing file are generated; all other Classes are ignored.

4. Highlight the Classes to generate. Leave unselected any to not generate.
5. To make Enterprise Architect automatically generate directories and filenames based on the package hierarchy, select the **Auto Generate Files** checkbox. This then enables the **Root Directory** field, in which you select a root directory under which the source directories are to be generated. By default, the **Auto Generate Files** feature ignores any file paths that are already associated with a Class. You can change this behavior by also selecting the **Retain Existing File Paths** checkbox.
6. To include all sub-packages in the output, select the **Include Child Packages** checkbox.
7. Click on the **Generate** button to start generating code.

As code generation proceeds Enterprise Architect displays progress messages. If a Class requires an output filename Enterprise Architect prompts you to enter one at the appropriate time (assuming **Auto Generate Files** is not selected). For example, if the selected Classes include partial Classes, a prompt displays to enter the filename for the second partial Class.

For additional information on the options on the **Generate Package Source Code** dialog, see the following table:
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Package</td>
<td>Check the name of the package to be generated.</td>
</tr>
<tr>
<td>Synchronize</td>
<td>Select options that specify how existing files should be generated.</td>
</tr>
<tr>
<td>Auto Generate Files</td>
<td>Specify whether Enterprise Architect should automatically generate filenames and directories, based on the package hierarchy.</td>
</tr>
<tr>
<td>Root Directory</td>
<td>If <strong>Auto Generate Files</strong> is selected, display the path under which the generated directory structures are created.</td>
</tr>
<tr>
<td>Retain Existing File Paths</td>
<td>If <strong>Auto Generate Files</strong> is selected, specify whether to use existing file paths associated with Classes. If unselected, Enterprise Architect generates Classes to automatically determined paths, regardless of whether source files are already associated with Classes.</td>
</tr>
<tr>
<td>Include all Child Packages</td>
<td>Include all Classes from all sub-packages of the target package in the list. This option facilitates recursive generation of a given package and its sub-packages.</td>
</tr>
<tr>
<td>Select Objects to Generate</td>
<td>List all Classes that are available for generation under the target packages. Only selected (highlighted) Classes are generated. Classes are listed with their target source file.</td>
</tr>
<tr>
<td>Select All</td>
<td>Mark all Classes in the list as selected.</td>
</tr>
<tr>
<td>Select None</td>
<td>Mark all Classes in the list as unselected.</td>
</tr>
<tr>
<td>Generate</td>
<td>Start the generation of all selected Classes.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Exit the Generate Package Source Code dialog. No Classes are generated.</td>
</tr>
</tbody>
</table>
9.2.4 Update Package Contents

Enterprise Architect enables you to synchronize a directory tree. Follow the steps below:

1. In the Project Browser, right-click on the root package of the tree to synchronize. The context menu displays.
3. In the Update Type panel, select the radio button to Forward Engineer or Reverse Engineer the package Classes.
4. To include child packages in the synchronization, select the Include child packages in generation checkbox.
5. Click on the OK button to start.

Enterprise Architect uses the directory names specified when the project source was first imported/generated and updates either the model or the source code depending on the option chosen.
9.2.5 Namespaces

Languages such as Java support package structures or namespaces. Enterprise Architect lets you specify a package as a namespace root, which denotes where the namespace structure starts; all subordinate packages below this point are generated as namespaces to code.

To define a package as a namespace root, right-click on the package in the Project Browser and select the Code Engineering | Set as Namespace Root menu option. The package icon in the Project Browser changes to include a colored corner (.setImageResource(258, 658)). When you have set the namespace root, the menu option changes to Clear Namespace Root; click on this option to take the namespace root status off the package. (Also, see the context menu described below.)

Once you have set a namespace root, Java code generated beneath this root automatically adds a package declaration at the head of the generated file indicating the current package location.

To view a list of namespaces, select the Settings | Namespaces menu option. The Namespaces dialog displays.

If you double-click on a namespace in the list, the package is highlighted in the Project Browser. Alternatively, right-click on the namespace to display a context menu, and select the Locate Package in Browser menu option.

You can also clear the selected namespace, by selecting the Clear Namespace Attribute option.
9.3 Code Engineering Settings

You can set the default code options such as the editors for each of the programming languages available for Enterprise Architect and special options for how source code is generated.

See Also
- General Options
- Local Paths
- Local Path Dialog
- Language Macros
- Setting Collection Classes
9.3.1 Source Code Engineering

The following topics describe general options that apply to all languages when generating code from Enterprise Architect. These options are all available under the Source Code Engineering section of the Options dialog (select the Tools | Options | Source Code Engineering menu option).

- Source Code Options
- Options - Code Editors
- Options - Object Lifetimes
- Options - Attribute/Operations
- Synchronize Model and Code
- Code Page for Source Editing

9.3.1.1 Source Code Options

When you generate code for a particular language, you can set certain options. These include:

- 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 the Source Code Engineering page of the Options dialog (select the Tools | Options | Source Code Engineering menu option).

Most of the settings are self-explanatory. The Remove prefixes when generating Get/Set properties field enables you to specify prefixes used in your variable naming conventions, if those prefixes should be removed in the variables' corresponding get/set functions.

Click on the Component Types button to configure what elements you would like to be created for files of any extension found while importing a source code directory.
Note:

It is worthwhile to configure these settings, as they serve as the defaults for all Classes in the model. You can override these on a per-Class basis using the custom settings (from the Code Generation dialog).

9.3.1.1 Import Component Types

The Import Component Types dialog enables 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 select the Tools | Options | Source Code Engineering menu option to display the Source Code Engineering page of the Options dialog, and click on the Component Types button.

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Stereotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>.sln</td>
<td>Artifact</td>
<td>solution</td>
</tr>
<tr>
<td>.binp</td>
<td>Component</td>
<td>bitmap</td>
</tr>
<tr>
<td>.sln</td>
<td>Artifact</td>
<td>solution</td>
</tr>
<tr>
<td>.rc</td>
<td>Artifact</td>
<td>resource</td>
</tr>
</tbody>
</table>

For each extension you can specify:

- The element type to be created
- The stereotype to apply to these objects.

Note:

You can transport these import component types between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

9.3.1.2 Options - Code Editors

You access the source code editor options via the DDL page of the Options dialog (select the Tools | Options | Source Code Engineering | Code Editors menu option). They enable you to configure options for Enterprise Architect’s internal editor, as well as the default editor for DDL scripts. You can configure external editors for code languages on each language options page.
The options for the inbuilt editor are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use inbuilt editor if no external editor set</td>
<td>Specify the editor for code in a language if no external editor is defined for that language.</td>
</tr>
<tr>
<td>Show Line Numbers</td>
<td>Show line numbers in the editor.</td>
</tr>
<tr>
<td>Enable Outlining</td>
<td>Enable collapsible regions for standard languages.</td>
</tr>
<tr>
<td>Show Structure Tree</td>
<td>Show a tree with the results of parsing the open file (requires that the file is parsed successfully).</td>
</tr>
<tr>
<td>Don't parse files larger than</td>
<td>Specify an upper limit on file size for parsing. Used to prevent performance decrease due to parsing very large files.</td>
</tr>
</tbody>
</table>

**9.3.1.3 Options - Object Lifetimes**

This set of options enables you to configure:

- Constructor details when generating code
- Whether to create a copy constructor
- Destructor details.

These options are accessed via the Constructor page of the Options dialog (select the Tools | Options | Source Code Engineering | Object Lifetimes menu option).
9.3.1.4 Options - Attribute/Operations

This set of options enables 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
- Define whether or not the bodies of methods are included and saved in the Enterprise Architect model when reverse engineering
- Create attributes in quick succession, clearing the dialog when you click on Save so that you can enter another attribute name.

These options are accessed via the Attribute Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | Attribute/Operations menu option).
9.3.1.5 Code Page for Source Editing

Enterprise Architect enables you to define the Unicode character set for code generation. To set the Unicode character set follow the steps below:

2. In the **Code page for source editing** field, click on the drop-down arrow and select the appropriate Unicode character set.

3. Click on the **Close** button.
9.3.2 Local Paths

Sometimes a team of developers could be working on the same Enterprise Architect model. Each developer might 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 worth while.

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 Enterprise Architect model located on a shared (or replicated) network drive.

To handle this scenario, Enterprise Architect enables you to define local paths for each Enterprise Architect user, using the Local Paths dialog (select the Settings | Local Paths menu option).

In our example, Developer A might define a local path of:

```
JAVA_SOURCE = "C:\Java\Source"
```

All Classes generated and stored in the Enterprise Architect project are stored as:

```
%JAVA_SOURCE%<xxx.java>.
```

Developer B now defines local path as:

```
JAVA_SOURCE = "D:\Source".
```

Now, Enterprise Architect stores all java files in these directories as:

```
%JAVA_SOURCE%<filename>
```

On each developer's machine, the filename is expanded to the correct local version.
9.3.3 Local Paths Dialog

The Local Paths dialog enables 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 the Local Paths topic. To open the Local Paths dialog, select the Settings | Local Paths option.

The Local Paths dialog enables you to define:

- **Path** - the local directory in the file system (e.g. d:\java\source)
- **ID** - the shared ID that is substituted for the Local Path (e.g. JAVA_SRC)
- **Type** - the language type (e.g. Java).

And also to:

- **Apply Path** - Select a path and click on this button to update any existing paths in the model (with full path names) to the shared relative path name (so d:\java\source\main.java might become %JAVA_SRC%\main.java)
- **Expand Path** - The opposite of Apply Path. This enables you to remove a relative path and substitute the full path name.

Using the two options you can update and change existing paths.
9.3.4 Language Macros

When reverse engineering a language such as C++, you might find preprocessor directives scattered throughout the code. This can make code management easier, but can hamper parsing of the underlying C++ language.

To help remedy this, you can include any number of macro definitions, which are ignored during the parsing phase of the reverse engineering. It is still preferable, if you have the facility, to preprocess the code using the appropriate compiler first; this way, complex macro definitions and defines are expanded out and can be readily parsed. If you don't have this facility, then this option provides a convenient substitute.

**Note:**

You can transport these language macro (or preprocessor macro) definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu. The macros are exported as a Macro List.

Define a Macro

1. Select the Settings | Preprocessor Macros menu option. The Language Macros dialog displays.

2. Click on the Add New button.
3. Enter details for your macro.
4. Click on the OK button.

**Macros Embedded Within Declarations**

Macros are sometimes used within the declaration of Classes and operations, as in the following examples:

```cpp
class __declspec Foo{int __declspec Bar(int p);};
```

If `declspec` is defined as a C++ macro, as outlined above, the imported Class and operation contain a Tagged Value called `DeclMacro1` with value `declspec`. (Subsequent macros would be defined as `DeclMacro2, DeclMacro3` and so on.) During forward engineering, these Tagged Values are used to regenerate the macros in code.

Define Complex Macros

It is sometimes useful to define rules for complex macros that can span multiple lines. Enterprise Architect ignores the entire code section defined by the rule. Such macros can be defined in Enterprise Architect as in the following two examples. Both types can be combined in one definition.

**Block Macros**

```
BEGIN_INTERFACE_PART ^ END_INTERFACE_PART
```

where the `^` symbol represents the body of the macro. This enables skipping from one macro to another.
**Note:**
The spaces surrounding the ^ symbol are required.

*Function Macros*

`RTTI_EMULATION()`
where Enterprise Architect skips over the token including everything inside the parentheses.
9.3.5 Set Collection Classes

Enterprise Architect enables you to define Collection Classes for generating code from Association connectors where the target role has a multiplicity setting greater than 1. There are two options for doing this:

1. On the Source Code Engineering section of the Options dialog (select the Tools | Options | Source Code Engineering option), on each language page click on the Collection Classes button.

The Collection Classes for Association Roles dialog displays.

On this dialog, 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. On the Detail tab of the Class Properties dialog (accessible from the right-click context menu of any Class), click on the Collection Classes button.

The Collection Classes for Association Roles dialog again displays, but here you define for when only this Class is used:
- 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.

When Enterprise Architect generates code for a connector 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; Enterprise Architect replaces this with the name of the Class being collected at source generation time (e.g. Vector<\#TYPE\#> would become Vector<foo>).

Additionally, on both the Source Role and Target Role tabs of the Association Property dialog (accessible from the right-click context menu of any Association) there is a Member Type field. If you set this, the value you enter overrides all the above options. The example below shows a defined PersonList; when code is generated, because this has a Multiplicity greater than 1 and the Member Type is defined, the variable created is of type PersonList.
9.3.6 Language Options

You can set up various options for how Enterprise Architect handles a particular language when generating code. These options are accessible on the Options dialog (select the Tools | Options menu option).

Under the Source Code Engineering option, select the required language. The following topics outline the options available for each language.

- ActionScript
- ANSI C
- C#
- C++
- Delphi
- Delphi Properties
- Java
- PHP
- Python
- Visual Basic
- VB.Net
- MDG Technology Languages
- Reset Options

9.3.6.1 ActionScript Options

Configure options for ActionScript code generation using the ActionScript Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | ActionScript menu option). The options you can specify include the:

- Default ActionScript version to generate (AS2.0 or AS3.0)
- Default file extensions (header and source)
- Default source directory
- Editor for ActionScript code.
9.3.6.2 C Options

Configure options for C code generation using the C Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | C menu option). The options you can specify include:

- Support for Object Oriented coding
- Default file extensions (header and source)
- Default source directory
- Editor for C code
- Path that Enterprise Architect uses to search for the implementation file; the first path in the list is the default path when generating.

9.3.6.3 C# Options

Configure options for C# code generation using the C# Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | C# menu option). The options you can specify include the default:

- File extension
- 'Get' prefix
- 'Set' prefix
- Directory for opening and saving C# source code.
9.3.6.4 C++ Options

Configure options for C++ code generation using the C++ Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | C++ menu option). 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 by reference
- The default file extensions
- Default Get/Set prefixes
- Default source directory
- The path that Enterprise Architect uses to search for the implementation file. The first path in the list is the default path when generating new implementation files and parsing existing files; if you add further directories, Enterprise Architect also searches these when parsing existing files.

For example, you have a directory inc that contains all of your headers, while the source code is mixed through directories src, srca, and srcb. You therefore set the Source Path option to ../../../srca/;../../../srcb/;../src/. This ensures that new implementation files are generated into src, but when parsing existing files Enterprise Architect looks in all three source directories (but never in the inc directory). You must still ensure that the implementation file name matches the header file name, and that the file extension matches the extension specified in the options. If these conditions are not met, Enterprise Architect cannot handle that code.


9.3.6.5 Delphi Options

Configure options for Delphi code generation using the Delphi Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | Delphi menu option). The options you can specify include the:

- Default file extension
- Default source directory

You can also set a default directory for opening and saving Delphi source code.
You should also set Delphi properties within each Class element.

9.3.6.5.1 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 Feature Visibility element context menu option, you can display the "tags" compartment that contains the properties. Imported Delphi Classes with properties have this feature automatically made visible for your convenience.

Note:

When you use the Create Property dialog from the Attribute screen, Enterprise Architect generates a pair of Get and Set functions, together with the required property definition as Tagged Values. You can manually edit these Tagged Values if required.
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 the Delphi Properties context menu option to open the editor.

The Delphi Properties editor enables 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 (the drop-down list includes all defined Classes in the project)
- Set the Read and Write information (the drop-down lists have all the attributes and operations from the current Class; you can also enter free text)
- Set Stored to True or False
- Set the Implements information
- Set the Default value, if one exists.
Notes:

- Public properties are displayed with a '+' symbol prefix and published with a '^'.
- When creating a property in the Property Wizard (accessed through the Attributes dialog), you can set the scope to **Published** if the property type is Delphi - see the example below.

Limitations

- Only **Public** and **Published** are 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.
9.3.6.6 Java Options

Configure options for Java code generation using the Java Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | Java menu option). The options you can specify include:

- Default file extension
- Default 'Get' prefix
- Default 'Set' prefix

You can also set a default directory for opening and saving Java source code.

9.3.6.7 PHP Options

Configure options for PHP code generation using the PHP Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | PHP menu option). The options you can specify include:

- Default source extension - specify the extension to be used when creating files for PHP source
- Default import extension - a semi-colon separated list of extensions to look at when doing a directory code import for PHP
- Default PHP version - the version of PHP to generate.

You can also set a default directory for opening and saving PHP source code.
9.3.6.8 **Python Options**

Configure options for Python code generation using the Python Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | Python menu option). The options you can specify include the:

- Default file extension(s)
- Default source directory.

You can also set the editor for Python code.
### 9.3.6.9 Visual Basic Options

Configure options for Visual Basic code generation using the VB Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | Visual Basic menu option). 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.
9.3.6.10 VB .Net Options

Configure options for VB.Net code generation using the VB.Net Specifications page of the Options dialog (select the Tools | Options | Source Code Engineering | VB.Net menu option). The options you can specify include the:

- Default file extension
- Default source directory.
9.3.6.11 MDG Technology Language Options

If you have loaded an MDG Technology that specifies a code module into your Sparx Systems > EA > MDG Technologies folder, the language is included in the Source Code Engineering list on the Options dialog. The language is only listed on the Options dialog if an MDG Technology file actually uses it in your model.

The options for each language are based on what is defined in the technology code module, but are limited to the following:

- **Default Extension**
  - Default extension for generated source files
  - Shown if the option is in the technology
  - Saved per project.

- **Import File Extensions**
  - Default folder to import source files from
  - Shown if there is a grammar set in the technology
  - Saved once for all projects.

- **Generate Namespaces**
  - Option to generate namespaces or not
  - Shown if the technology supports namespaces
  - Saved once for all projects.

- **Default Source Directory**
  - The default directory to save generated source files
  - Always shown
  - Saved once for all projects.

- **Editor**
  - The editor that is loaded to edit the source files
  - Always shown
  - Saved once for all projects.

- **Att Type**
  - Default type for attributes
  - Always shown
  - Saved once for all projects.

These options are set in the technology inside the `<CodeOptions>` tag of a code module, as follows:

```
<CodeOption name="DefaultExtension">.rb</CodeOption>
```
9.3.6.12 Reset Options

Enterprise Architect stores some of the options for a Class when it is first created. 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 dialog in existing Classes. You must modify the options for the existing Class.

Modify Options for Single Class
To modify options for a single Class, follow the steps below:

1. Right-click on the Class to change, and select the Generate Code menu option from the context menu. The Generate Code dialog displays.
2. Click on the Advanced button. The Object Options dialog displays.
3. Click on the Attributes/Operations button.
4. Change the options, and click on the Close button to apply changes.

Modify Options for All Classes
To modify options for all Classes within a package, follow the steps below:

1. Right-click on the package in the Project Browser. The context menu displays.
3. Reset the required defaults for each existing Class.
4. Click on the **OK** button to apply changes.
9.4 Code Template Framework

The Code Template Framework (CTF) is used during forward engineering of UML models. The CTF enables you to:

- Generate source code from UML models
- Customize the way in which Enterprise Architect generates source code
- Forward engineer languages not specifically supported by Enterprise Architect.

The CTF consists of:

- Default Code Templates which are built into Enterprise Architect for forward engineering supported languages
- A Code Template Editor for creating and maintaining user-defined Code Templates (also see SDK for Enterprise Architect)
- Code templates to synchronize code
9.4.1 Code Templates

Code templates enable you to customize code generation of existing languages. For example:

- Modify the file headers created when generating new files
- Change the style of the generated code (such as indenting or brace position) to match the required coding standards
- Handle particular stereotypes to generate things like specialized method bodies and extra methods.

They also enable you to add code generation of entirely new languages that Enterprise Architect would otherwise not be able to handle. In this situation it is most useful to combine code templates with an MDG technology file that includes the datatypes, and options for default file extensions.

Enterprise Architect's base 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 Enterprise Architect 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{
...
}
```

## Execution of Code Templates

A reference to a template (such as the %ClassNotes% macro, from our example above) 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 Enterprise Architect looks for a template that has been defined for that stereotype. If a match is found, the specialized 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.

### 9.4.1.1 Base Templates

The Code Template Framework 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 table lists and briefly describes the base templates used in the CTF.

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>A top-level template to generate member variables from UML attributes.</td>
</tr>
<tr>
<td>Attribute Declaration</td>
<td>Used by the Attribute template to generate a member variable declaration.</td>
</tr>
<tr>
<td>Attribute Notes</td>
<td>Used by the Attribute template to generate member variable notes.</td>
</tr>
<tr>
<td>Class</td>
<td>A top-level template for generating Classes from UML Classes.</td>
</tr>
<tr>
<td>Class Base</td>
<td>Used by the 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.</td>
</tr>
<tr>
<td>Class Body</td>
<td>Used by the Class template to generate the body of a Class.</td>
</tr>
<tr>
<td>Class Declaration</td>
<td>Used by the Class template to generate the declaration of a Class.</td>
</tr>
<tr>
<td>Template</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Class Interface</td>
<td>Used by the Class template to generate an interface name in the inheritance list of a derived Class, where the interface doesn’t exist in the model.</td>
</tr>
<tr>
<td>Class Notes</td>
<td>Used by the Class template to generate the Class notes.</td>
</tr>
<tr>
<td>File</td>
<td>A top-level template for generating the source file. For languages such as C++, this corresponds to the header file.</td>
</tr>
<tr>
<td>Import Section</td>
<td>Used in the File template to generate external dependencies.</td>
</tr>
<tr>
<td>Linked Attribute</td>
<td>A top-level template for generating attributes derived from UML Associations.</td>
</tr>
<tr>
<td>Linked Attribute Notes</td>
<td>Used by the Linked Attribute template to generate the attribute notes.</td>
</tr>
<tr>
<td>Linked Class Base</td>
<td>Used by the 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.</td>
</tr>
<tr>
<td>Linked Class Interface</td>
<td>Used by the 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.</td>
</tr>
<tr>
<td>Namespace</td>
<td>A 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.)</td>
</tr>
<tr>
<td>Namespace Body</td>
<td>Used by the Namespace template to generate the body of a namespace.</td>
</tr>
<tr>
<td>Namespace Declaration</td>
<td>Used by the Namespace template to generate the namespace declaration.</td>
</tr>
<tr>
<td>Operation</td>
<td>A top-level template for generating operations from a UML Class's operations.</td>
</tr>
<tr>
<td>Operation Body</td>
<td>Used by the Operation template to generate the body of a UML operation.</td>
</tr>
<tr>
<td>Operation Declaration</td>
<td>Used by the Operation template to generate the operation declaration.</td>
</tr>
<tr>
<td>Operation Notes</td>
<td>Used by the Operation template to generate documentation for an operation.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Used by the Operation Declaration template to generate parameters.</td>
</tr>
</tbody>
</table>

The second table lists templates used for generating code for languages that have separate interface and implementation sections.

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Impl</td>
<td>A top-level template for generating the implementation of a Class.</td>
</tr>
<tr>
<td>Class Body Impl</td>
<td>Used by the Class Impl template to generate the implementation of Class members.</td>
</tr>
<tr>
<td>File Impl</td>
<td>A top-level template for generating the implementation file.</td>
</tr>
<tr>
<td>File Notes Impl</td>
<td>Used by the File Impl template to generate notes in the source file.</td>
</tr>
<tr>
<td>Import Section Impl</td>
<td>Used by the File Impl template to generate external dependencies.</td>
</tr>
<tr>
<td>Operation Impl</td>
<td>A top-level template for generating operations from a UML Class's operations.</td>
</tr>
<tr>
<td>Operation Body Impl</td>
<td>Used by the Operation template to generate the body of a UML operation.</td>
</tr>
<tr>
<td>Operation Declaration Impl</td>
<td>Used by the Operation template to generate the operation declaration.</td>
</tr>
<tr>
<td>Operation Notes Impl</td>
<td>Used by the Operation template to generate documentation for an operation.</td>
</tr>
</tbody>
</table>

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 in the example diagram below. In this diagram the templates are modeled 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 must be specialized to be of use in code engineering. In particular, each template is specialized for the supported languages (or ‘products’). For example, there is a ClassBody template defined for C++, another for C#, another for Java, and so on. By specializing the templates, you can tailor the code generated for the corresponding UML entity.

Once the base templates are specialized for a given language, they can be further specialized based on:

- A Class's stereotype
- A feature's stereotype (where the feature can be an operation or attribute)

This type of specialization enables, for example, a C# operation that is stereotyped as «property» to have a different Operation Body template from an ordinary operation. The Operation Body template can then be specialized further, based on the Class stereotype.

**Note:**

The above Class Model shows the hierarchy of Code Generation templates for a language such as C# or Java. The Aggregation connectors denote references between templates.
### 9.4.2 The Code Template Editor

The Code Template Editor window is accessed by selecting the **Settings | Code Generation Templates** menu option.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td>Select the programming language.</td>
</tr>
<tr>
<td><strong>New Language</strong></td>
<td>Display the Programming Languages Datatypes dialog, which enables you to include programming languages other than those supported for Enterprise Architect, for which to create or edit code templates.</td>
</tr>
<tr>
<td><strong>Template</strong></td>
<td>Display the contents of the active template, and provides the editor for modifying templates.</td>
</tr>
<tr>
<td><strong>Templates</strong></td>
<td>List the base code templates. The active template is highlighted. The <strong>Modified</strong> field indicates whether you have changed the default template for the current language.</td>
</tr>
<tr>
<td><strong>Stereotype Overrides</strong></td>
<td>List the stereotyped templates, for the active base template. The <strong>Modified</strong> field indicates whether you have modified a default stereotyped template.</td>
</tr>
<tr>
<td><strong>Add New Custom Template</strong></td>
<td>Invoke a dialog for creating a custom stereotyped template.</td>
</tr>
<tr>
<td><strong>Add New Stereotyped Override</strong></td>
<td>Invoke a dialog for adding a stereotyped template, for the currently selected base template.</td>
</tr>
<tr>
<td><strong>Get Default Template</strong></td>
<td>Update the editor display with the default version of the active template.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Save</td>
<td>Overwrite the active templates with the contents of the editor.</td>
</tr>
<tr>
<td>Delete</td>
<td>If you have overridden the active template, the override is deleted and replaced by the corresponding default code template.</td>
</tr>
</tbody>
</table>

For information on creating and editing code templates using the Code Template Editor window, see [SDK for Enterprise Architect](https://www.sparxsystems.com/).
9.4.3 Synchronize Code

Enterprise Architect uses code templates during the forward synchronization of the following programming languages:

- ActionScript
- C
- C++
- C#
- Delphi
- Java
- PHP
- Python
- VB
- VB.Net

Only a subset of the code templates are used during synchronization. This subset corresponds to the distinct sections that Enterprise Architect recognizes in the source code. The following table lists the code templates and their corresponding code sections, which can be synchronized.

<table>
<thead>
<tr>
<th>Code Template</th>
<th>Code Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Notes</td>
<td>Comments preceding Class declaration.</td>
</tr>
<tr>
<td>Class Declaration</td>
<td>Up to and including Class parents.</td>
</tr>
<tr>
<td>Attribute Notes</td>
<td>Comments preceding Attribute declaration.</td>
</tr>
<tr>
<td>Attribute Declaration</td>
<td>Up to and including terminating character.</td>
</tr>
<tr>
<td>Operation Notes</td>
<td>Comments preceding operation declaration.</td>
</tr>
<tr>
<td>Operation Notes Impl</td>
<td>As for Operation Notes.</td>
</tr>
<tr>
<td>Operation Declaration</td>
<td>Up to and including terminating character.</td>
</tr>
<tr>
<td>Operation Declaration Impl</td>
<td>Up to and including terminating character.</td>
</tr>
<tr>
<td>Operation Body</td>
<td>Everything between and including the braces.</td>
</tr>
<tr>
<td>Operation Body Impl</td>
<td>As for Operation Body.</td>
</tr>
</tbody>
</table>

Three types of change can occur in the source when it is synchronized with the UML model:

- **Synchronize Existing Sections** for example, changing the return type in an operation declaration
- **Add New Sections to Existing Features** for example, adding notes to a Class declaration, where there were previously none
- **Add New Features and Elements** for example, adding a new operation to a Class.

Each of these changes must be handled differently by Enterprise Architect; their effect on the CTF is described in the linked topics above.

9.4.3.1 Synchronize 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 you 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
```

9.4.3.2 Add New Sections

The following can be added as new sections, to existing features in the source code:

- Class Notes
- Attribute Notes
- Operation Notes
Assume Class A from the previous example had no note when you originally generated the code. Now assume that you specify a note in the model for Class A. Enterprise Architect attempts to add the new note from the model during synchronization. It does this by executing the Class Notes template.

To make room for the new section to be inserted, you can specify how much white space to append to the section via synchronization macros. These macros are described in SDK for Enterprise Architect.

### 9.4.3.3 Add 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. Enterprise Architect 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 are indented according to the value set for this macro, during synchronization. This value is ignored for Classes defined within a package setup as a root namespace, or if the Generate Namespace option is set to False in the appropriate language page (C#, C++ or VB.Net) on the Options dialog (Tools | Options | Source Code Engineering | &lt;language&gt;).
9.5 Modeling Conventions

In order to get the most out of the round trip engineering in Enterprise Architect, you must be familiar with the modeling conventions used when generating and reverse engineering the languages you use. This topic describes the stereotypes, Tagged Values and other conventions used in code engineering in Enterprise Architect for the following supported languages:

- ActionScript
- C
- C#
- C++
- Delphi
- Java
- PHP
- Python
- VB.Net
- Visual Basic

Note:

Enterprise Architect incorporates a number of visibility indicators or scope values for its supported languages. These include, for:

- All languages - Public (+), Protected (#) and Private (-)
- Java - Package (-)
- Delphi - Published (*)
- C# - Internal (-), Protected Internal (*)
- ActionScript - Internal (-)
- VB.NET - Friend (-), Protected Friend (*)
- PHP - Package (-)
- Python - Package (-)
- C - Package (-)
- C++ - Package (-).
9.5.1 ActionScript Conventions

Enterprise Architect supports round trip engineering of ActionScript 2 and 3, where the following conventions are used.

**Stereotypes**

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>literal</td>
<td>Operation</td>
<td>A literal method referred to by a variable.</td>
</tr>
<tr>
<td>property get</td>
<td>Operation</td>
<td>A read property.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>A write property.</td>
</tr>
</tbody>
</table>

**Tagged Values**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype property get or property set</td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>dynamic</td>
<td>Class or Interface</td>
<td>The <code>dynamic</code> keyword.</td>
</tr>
<tr>
<td>final</td>
<td>ActionScript 3: Operation</td>
<td>The <code>final</code> keyword.</td>
</tr>
<tr>
<td>intrinsic</td>
<td>ActionScript 2: Class</td>
<td>The <code>intrinsic</code> keyword</td>
</tr>
<tr>
<td>namespace</td>
<td>ActionScript 3: Class, Interface, Attribute, Operation</td>
<td>The namespace of the current element.</td>
</tr>
<tr>
<td>override</td>
<td>ActionScript 3: Operation</td>
<td>The <code>override</code> keyword.</td>
</tr>
<tr>
<td>prototype</td>
<td>ActionScript 3: Attribute</td>
<td>The <code>prototype</code> keyword.</td>
</tr>
<tr>
<td>rest</td>
<td>ActionScript 3: Parameter</td>
<td>The <code>rest</code> parameter (...).</td>
</tr>
</tbody>
</table>

**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
9.5.2 C Conventions

**Note:** Separate conventions apply to Object Oriented programming in C.

Enterprise Architect supports round trip engineering of C, where the following conventions are used:

### Stereotype

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumeration</td>
<td>Inner Class</td>
<td>An enum type.</td>
</tr>
<tr>
<td>struct</td>
<td>Inner Class</td>
<td>A struct type.</td>
</tr>
<tr>
<td></td>
<td>Attribute</td>
<td>A keyword struct in variable definition.</td>
</tr>
<tr>
<td>typedef</td>
<td>Inner Class</td>
<td>A typedef statement, where the parent is the original type name.</td>
</tr>
<tr>
<td>union</td>
<td>Inner Class</td>
<td>A union type.</td>
</tr>
<tr>
<td></td>
<td>Attribute</td>
<td>A keyword union in variable definition.</td>
</tr>
</tbody>
</table>

### Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous</td>
<td>Class also containing the Tagged Value typedef</td>
<td>The name of this class being defined only by the typedef statement.</td>
</tr>
<tr>
<td>bodyLocation</td>
<td>Operation</td>
<td>The location the method body is generated to. Expected values are header, classDec or classBody.</td>
</tr>
<tr>
<td>typedef</td>
<td>Class with stereotype other than typedef</td>
<td>This Class being defined in a typedef statement.</td>
</tr>
</tbody>
</table>

### C Code Generation for UML Model

<table>
<thead>
<tr>
<th>UML</th>
<th>C Code</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Class</td>
<td>A pair of C files (.h + .c)</td>
<td>File name is the same as Class name.</td>
</tr>
<tr>
<td>Operation (public &amp; protected)</td>
<td>Function declaration in .h file and definition in .c file</td>
<td></td>
</tr>
<tr>
<td>Operation (private)</td>
<td>Function definition in .c file only</td>
<td></td>
</tr>
<tr>
<td>Attribute (public &amp; protected)</td>
<td>Variable definition in .h file</td>
<td></td>
</tr>
<tr>
<td>Attribute (private)</td>
<td>Variable definition in .c file</td>
<td></td>
</tr>
<tr>
<td>Inner Class (without stereotype)</td>
<td>(N/A)</td>
<td>This inner Class would be ignored</td>
</tr>
</tbody>
</table>

**See Also**
- Import Source Code
- Generate Source Code
- C Options
9.5.2.1 Object Oriented Programming In C

The following conventions are used for Object-Oriented programming in C.

To configure Enterprise Architect to support Object-Oriented programming using C, you must set the Object Oriented Support option to True on the C Specifications page of the Options dialog.

### Stereotype

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumeration</td>
<td>Class</td>
<td>An enum type.</td>
</tr>
<tr>
<td>struct</td>
<td>Class</td>
<td>A struct type.</td>
</tr>
<tr>
<td></td>
<td>Attribute</td>
<td>A keyword struct in variable definition.</td>
</tr>
<tr>
<td>typedef</td>
<td>Class</td>
<td>A typedef statement, where the parent is the original type name.</td>
</tr>
<tr>
<td></td>
<td>Attribute</td>
<td>A keyword union in variable definition.</td>
</tr>
</tbody>
</table>

### Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous</td>
<td>Class with stereotype of enumeration, struct or union.</td>
<td>The name of this Class being defined only by the typedef statement.</td>
</tr>
<tr>
<td>bodyLocation</td>
<td>Operation</td>
<td>The location the method body is generated to. Expected values are header, classDec or classBody.</td>
</tr>
<tr>
<td>define</td>
<td>Attribute</td>
<td>#define statement.</td>
</tr>
<tr>
<td>typedef</td>
<td>Class with stereotype of enumeration, struct or union.</td>
<td>This Class being defined in a typedef statement.</td>
</tr>
</tbody>
</table>

### Object-Oriented C Code Generation for UML Model

The basic idea of implementing a UML Class in C code is to group the data variable (UML attributes) into a structure type. This structure is defined in a .h file so that it can be shared by other classes and by the client that referred to it.

An operation in a UML Class is implemented in C code as a function. The name of the function must be a fully qualified name that consists of the operation name, as well as the Class name to indicate that the operation is for that Class. A delimiter (specified in the Namespace Delimiter option on the C Specifications page) is used to join the Class name and function (operation) name.

The function in C code must also have a reference parameter to the Class object. You can modify the Reference as Operation Parameter, Reference Parameter Style and Reference Parameter Name options on the C Specifications page to support this reference parameter.

### Limitations of Object-Oriented Programming in C

1. No scope mapping for an attribute: an attribute in a UML Class is mapped to a structure variable in C code, and its scope (private, protected or public) is ignored.
2. Currently an inner Class is ignored: if a UML Class is the inner Class of another UML Class, it is ignored when generating C code.
3. Initial value is ignored: the initial value of an attribute in a UML Class is ignored in generated C code.

### See Also

- Import Source Code
- Generate Source Code
- C Options
9.5.3 C# Conventions

Enterprise Architect supports the round trip engineering of C#, where the following conventions are used.

**Stereotypes**

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumeration</td>
<td>Class</td>
<td>An enum type.</td>
</tr>
<tr>
<td>event</td>
<td>Operation</td>
<td>An event.</td>
</tr>
<tr>
<td>indexer</td>
<td>Operation</td>
<td>A property acting as an index for this Class.</td>
</tr>
<tr>
<td>property</td>
<td>Operation</td>
<td>A property possibly containing both read and write code.</td>
</tr>
<tr>
<td>struct</td>
<td>Class</td>
<td>A struct type.</td>
</tr>
</tbody>
</table>

**Tagged Values**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype</td>
<td>The name of the variable behind this property or event.</td>
</tr>
<tr>
<td></td>
<td>property or event</td>
<td></td>
</tr>
<tr>
<td>const</td>
<td>Attribute</td>
<td>The const keyword.</td>
</tr>
<tr>
<td>delegate</td>
<td>Operation</td>
<td>The delegate keyword.</td>
</tr>
<tr>
<td>enumType</td>
<td>Operation with stereotype</td>
<td>The datatype that the property is represented as.</td>
</tr>
<tr>
<td></td>
<td>property</td>
<td></td>
</tr>
<tr>
<td>extern</td>
<td>Operation</td>
<td>The extern keyword.</td>
</tr>
<tr>
<td>generic</td>
<td>Operation</td>
<td>The generic parameters for this Operation.</td>
</tr>
<tr>
<td>genericConstraints</td>
<td>Templated Class or Interface,</td>
<td>The constraints on the generic parameters of this type or operation.</td>
</tr>
<tr>
<td></td>
<td>Operation with tag generic</td>
<td></td>
</tr>
<tr>
<td>Implements</td>
<td>Operation</td>
<td>The name of the method this implements, including the interface name.</td>
</tr>
<tr>
<td>ImplementsExplicit</td>
<td>Operation</td>
<td>The presence of the source interface name in this method declaration.</td>
</tr>
<tr>
<td>initializer</td>
<td>Operation</td>
<td>A constructor initialization list.</td>
</tr>
<tr>
<td>new</td>
<td>Class, Interface, Operation</td>
<td>The new keyword.</td>
</tr>
<tr>
<td>override</td>
<td>Operation</td>
<td>The override keyword.</td>
</tr>
<tr>
<td>params</td>
<td>Parameter</td>
<td>A parameter list using the params keyword.</td>
</tr>
<tr>
<td>partial</td>
<td>Class, Interface</td>
<td>The partial keyword.</td>
</tr>
<tr>
<td>readonly</td>
<td>Operation with stereotype</td>
<td>This property only defining read code.</td>
</tr>
<tr>
<td></td>
<td>property</td>
<td></td>
</tr>
<tr>
<td>sealed</td>
<td>Operation</td>
<td>The sealed keyword.</td>
</tr>
<tr>
<td>static</td>
<td>Class</td>
<td>The static keyword.</td>
</tr>
<tr>
<td>unsafe</td>
<td>Class, Interface, Operation</td>
<td>The unsafe keyword.</td>
</tr>
<tr>
<td>virtual</td>
<td>Operation</td>
<td>The virtual keyword.</td>
</tr>
<tr>
<td>writeonly</td>
<td>Operation with stereotype</td>
<td>This property only defining write code.</td>
</tr>
<tr>
<td></td>
<td>property</td>
<td></td>
</tr>
</tbody>
</table>

**Other Conventions**

- Namespaces are generated for each package below a namespace root.

Enterprise Architect User Guide
- 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](#)
9.5.4 C++ Conventions

Enterprise Architect supports round trip engineering of C++, including the Managed C++ and C++/CLI extensions, where the following conventions are used.

### Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumeration</td>
<td>Class</td>
<td>An <code>enum</code> type.</td>
</tr>
<tr>
<td>friend</td>
<td>Operation</td>
<td>The <code>friend</code> keyword.</td>
</tr>
<tr>
<td>property get</td>
<td>Operation</td>
<td>A read property.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>A write property.</td>
</tr>
<tr>
<td>struct</td>
<td>Class</td>
<td>A <code>struct</code> type.</td>
</tr>
<tr>
<td>typedef</td>
<td>Class</td>
<td>A <code>typedef</code> statement, where the parent is the original type name.</td>
</tr>
<tr>
<td>union</td>
<td>Class</td>
<td>A <code>union</code> type.</td>
</tr>
</tbody>
</table>

### Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>afx_msg</td>
<td>Operation</td>
<td>The <code>afx_msg</code> keyword.</td>
</tr>
<tr>
<td>anonymous</td>
<td>Class also containing the Tagged Value typedef</td>
<td>The name of this class being only defined by the typedef statement.</td>
</tr>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype property get or property set</td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>bitfield</td>
<td>Attribute</td>
<td>The size, in bits, allowed for storage of this attribute.</td>
</tr>
<tr>
<td>bodyLocation</td>
<td>Operation</td>
<td>The exceptions that are thrown by this method.</td>
</tr>
<tr>
<td>callback</td>
<td>Operation</td>
<td>A reference to the <code>CALLBACK</code> macro.</td>
</tr>
<tr>
<td>explicit</td>
<td>Operation</td>
<td>The <code>explicit</code> keyword.</td>
</tr>
<tr>
<td>initializer</td>
<td>Operation</td>
<td>A constructor initialization list.</td>
</tr>
<tr>
<td>inline</td>
<td>Operation</td>
<td>The <code>inline</code> keyword and inline generation of the method body.</td>
</tr>
<tr>
<td>mutable</td>
<td>Attribute</td>
<td>The location the method body is generated to. Expected values are <code>header</code>, <code>classDec</code> or <code>classBody</code>.</td>
</tr>
<tr>
<td>throws</td>
<td>Operation</td>
<td>The <code>mutable</code> keyword.</td>
</tr>
<tr>
<td>typedef</td>
<td>Class with stereotype other than typedef</td>
<td>This Class being defined in a <code>typedef</code> statement.</td>
</tr>
<tr>
<td>typeSynonyms</td>
<td>Class</td>
<td>The <code>typedef</code> name and/or fields of this type.</td>
</tr>
<tr>
<td>volatile</td>
<td>Operation</td>
<td>The <code>volatile</code> keyword.</td>
</tr>
</tbody>
</table>

### 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 \textit{Fixed} property of a parameter corresponds to the \textit{const} keyword.

\textbf{See Also}
• Import Source Code
• Generate Source Code
• C++ Options

\subsection*{9.5.4.1 Managed C++ Conventions}

The following conventions are used for managed extensions to C++ prior to C++/CLI. In order to set Enterprise Architect to generate managed C++ you must modify the C++ version in the C++ Options.

\textbf{Stereotypes}

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>Operation</td>
<td>The __property keyword.</td>
</tr>
<tr>
<td>property get</td>
<td>Operation</td>
<td>The __property keyword and a read property.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>The __property keyword and a write property.</td>
</tr>
<tr>
<td>reference</td>
<td>Class</td>
<td>The __gc keyword.</td>
</tr>
<tr>
<td>value</td>
<td>Class</td>
<td>The __value keyword.</td>
</tr>
</tbody>
</table>

\textbf{Tagged Values}

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>managedType</td>
<td>Class with stereotype reference, value or enumeration; Interface</td>
<td>The keyword used in declaration of this type. Expected values are class or struct.</td>
</tr>
</tbody>
</table>

\textbf{Other Conventions}

• The typedef and anonymous tags from native C++ are not supported
• The Pure property of an operation corresponds to the keyword __abstract.

\textbf{See Also}
• Import Source Code
• Generate Source Code

\subsection*{9.5.4.2 C++/CLI Conventions}

The following conventions are used for modeling C++/CLI extensions to C++. In order to set Enterprise Architect to generate managed C++/CLI you must modify the C++ version in the C++ Options.

\textbf{Stereotypes}

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>Operation</td>
<td>Defines an event to provide access to the event handler for this Class.</td>
</tr>
<tr>
<td>property</td>
<td>Operation, Attribute</td>
<td>This is a property possibly containing both read and write code.</td>
</tr>
<tr>
<td>reference</td>
<td>Class</td>
<td>Corresponds to the ref class or ref struct keyword.</td>
</tr>
<tr>
<td>value</td>
<td>Class</td>
<td>Corresponds to the value class or value struct keyword.</td>
</tr>
</tbody>
</table>
Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype property or event</td>
<td>The name of the variable behind this property or event.</td>
</tr>
<tr>
<td>generic</td>
<td>Operation</td>
<td>Defines the generic parameters for this Operation.</td>
</tr>
<tr>
<td>genericConstraints</td>
<td>Templated Class or Interface, Operation with tag generic</td>
<td>Defines the constraints on the generic parameters for this Operation.</td>
</tr>
<tr>
<td>initonly</td>
<td>Attribute</td>
<td>Corresponds to the initonly keyword.</td>
</tr>
<tr>
<td>literal</td>
<td>Attribute</td>
<td>Corresponds to the literal keyword.</td>
</tr>
<tr>
<td>managedType</td>
<td>Class with stereotype reference, value or enumeration; Interface</td>
<td>Corresponds to either the class or struct keyword.</td>
</tr>
</tbody>
</table>

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
9.5.5 Delphi Conventions

Enterprise Architect supports round trip engineering of Delphi, where the following conventions are used.

**Stereotypes**

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructor</td>
<td>Operation</td>
<td>A constructor.</td>
</tr>
<tr>
<td>destructor</td>
<td>Operation</td>
<td>A destructor.</td>
</tr>
<tr>
<td>dispinterface</td>
<td>Class, Interface</td>
<td>A dispatch interface.</td>
</tr>
<tr>
<td>enumeration</td>
<td>Class</td>
<td>An enumerated type.</td>
</tr>
<tr>
<td>operator</td>
<td>Operation</td>
<td>An operator.</td>
</tr>
<tr>
<td>property get</td>
<td>Operation</td>
<td>A read property.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>A write property.</td>
</tr>
<tr>
<td>struct</td>
<td>Class</td>
<td>A record type.</td>
</tr>
</tbody>
</table>

**Tagged Values**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype</td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>property get</td>
<td>property get or property set</td>
<td></td>
</tr>
<tr>
<td>overload</td>
<td>Operation</td>
<td>The <code>overload</code> keyword.</td>
</tr>
<tr>
<td>override</td>
<td>Operation</td>
<td>The <code>override</code> keyword.</td>
</tr>
<tr>
<td>packed</td>
<td>Class</td>
<td>The <code>packed</code> keyword.</td>
</tr>
<tr>
<td>property</td>
<td>Class</td>
<td>A property. See Delphi Properties for more information.</td>
</tr>
<tr>
<td>reintroduce</td>
<td>Operation</td>
<td>The <code>reintroduce</code> keyword.</td>
</tr>
</tbody>
</table>

**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
9.5.6 Java Conventions

Enterprise Architect supports round trip engineering of Java - including AspectJ extensions - where the following conventions are used.

### Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotation</td>
<td>Interface</td>
<td>An annotation type.</td>
</tr>
<tr>
<td>enum</td>
<td>Attributes within a Class stereotyped enumeration</td>
<td>An enumerated option, distinguished from other attributes that have no stereotype.</td>
</tr>
<tr>
<td>enumeration</td>
<td>Class</td>
<td>An enum type.</td>
</tr>
<tr>
<td>operator</td>
<td>Operation</td>
<td>An operator.</td>
</tr>
<tr>
<td>property get</td>
<td>Operation</td>
<td>A read property.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>A write property.</td>
</tr>
<tr>
<td>static</td>
<td>Class or Interface</td>
<td>The static keyword.</td>
</tr>
</tbody>
</table>

### Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotations</td>
<td>Anything</td>
<td>The annotations on the current code feature.</td>
</tr>
<tr>
<td>arguments</td>
<td>Attribute with stereotype enum</td>
<td>The arguments that apply to this enumerated value.</td>
</tr>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype property get or property set</td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>dynamic</td>
<td>Class or Interface</td>
<td>The dynamic keyword.</td>
</tr>
<tr>
<td>generic</td>
<td>Operation</td>
<td>The generic parameters to this operation.</td>
</tr>
<tr>
<td>parameterList</td>
<td>Parameter</td>
<td>A parameter list with the ... syntax.</td>
</tr>
<tr>
<td>throws</td>
<td>Operation</td>
<td>The exceptions that are thrown by this method.</td>
</tr>
</tbody>
</table>

### 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
- Java Options

#### 9.5.6.1 AspectJ Conventions

The following are the conventions used for supporting AspectJ extensions to Java.

### Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>advice</td>
<td>Operation</td>
<td>A piece of advice in an AspectJ aspect.</td>
</tr>
</tbody>
</table>
Stereotype | Applies to | Corresponds To
--- | --- | ---
aspect | Class | An AspectJ aspect.
pointcut | Operation | A pointcut in an AspectJ aspect.

Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>className</td>
<td>Attribute or operation within a Class stereotyped aspect</td>
<td>The Classes this AspectJ intertype member belongs to.</td>
</tr>
</tbody>
</table>

Other Conventions

- The specifications of a pointcut are included in the **Behavior** field of the method.

See Also

- [Import Source Code](#)
- [Generate Source Code](#)
9.5.7 **PHP Conventions**

Enterprise Architect supports the round trip engineering of PHP 4 and 5, where the following conventions are used.

### Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>property get</td>
<td>Operation</td>
<td>A read property.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>A write property.</td>
</tr>
</tbody>
</table>

### Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype property get or property set</td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>final</td>
<td>Operations in PHP 5.</td>
<td>The final keyword.</td>
</tr>
</tbody>
</table>

### 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 corresponds to the *Is Leaf* property
- The `abstract` Class modifier 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*](#)
9.5.8 Python Conventions

Enterprise Architect supports the round trip engineering of Python, where the following conventions are used.

Tagged values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Class, Operation</td>
<td>The decorators applied to this element in the source.</td>
</tr>
</tbody>
</table>

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
9.5.9 VB.Net Conventions

Enterprise Architect supports 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**

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>Operation</td>
<td>An event declaration.</td>
</tr>
<tr>
<td>import</td>
<td>Operation</td>
<td>An operation to be imported from another library.</td>
</tr>
<tr>
<td>module</td>
<td>Class</td>
<td>A module.</td>
</tr>
<tr>
<td>operator</td>
<td>Operation</td>
<td>An operator overload definition.</td>
</tr>
<tr>
<td>property</td>
<td>Operation</td>
<td>A property possibly containing both read and write code.</td>
</tr>
</tbody>
</table>

**Tagged Values**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>Operation with stereotype import</td>
<td>The alias for this imported operation.</td>
</tr>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype property</td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>CharSet</td>
<td>Operation with stereotype import</td>
<td>The character set clause for this import. One of the values Ansi, Unicode or Auto.</td>
</tr>
<tr>
<td>delegate</td>
<td>Operation</td>
<td>The Delegate keyword.</td>
</tr>
<tr>
<td>enumTag</td>
<td>Operation with stereotype property</td>
<td>The datatype that this property is represented as.</td>
</tr>
<tr>
<td>Handles</td>
<td>Operation</td>
<td>The handles clause on this operation.</td>
</tr>
<tr>
<td>Implements</td>
<td>Operation</td>
<td>The implements clause on this operation.</td>
</tr>
<tr>
<td>Lib</td>
<td>Operation with stereotype import</td>
<td>The library this import comes from.</td>
</tr>
<tr>
<td>MustOverride</td>
<td>Operation</td>
<td>The MustOverride keyword.</td>
</tr>
<tr>
<td>Narrowing</td>
<td>Operation with stereotype operator</td>
<td>The Narrowing keyword.</td>
</tr>
<tr>
<td>NotOverrideable</td>
<td>Operation</td>
<td>The NotOverrideable keyword.</td>
</tr>
<tr>
<td>Overloads</td>
<td>Operation</td>
<td>The Overloads keyword.</td>
</tr>
<tr>
<td>Overrides</td>
<td>Operation</td>
<td>The Overrides keyword.</td>
</tr>
<tr>
<td>parameterArray</td>
<td>Parameter</td>
<td>A parameter list using the ParamArray keyword.</td>
</tr>
<tr>
<td>partial</td>
<td>Class, Interface</td>
<td>The Partial keyword.</td>
</tr>
<tr>
<td>readonly</td>
<td>Operation with stereotype property</td>
<td>This property only defining read code.</td>
</tr>
<tr>
<td>shadows</td>
<td>Class, Interface, Operation</td>
<td>The Shadows keyword.</td>
</tr>
<tr>
<td>Shared</td>
<td>Attribute</td>
<td>The Shared keyword.</td>
</tr>
<tr>
<td>Widening</td>
<td>Operation with stereotype operator</td>
<td>The Widening keyword.</td>
</tr>
<tr>
<td>writeonly</td>
<td>Operation with stereotype property</td>
<td>This property only defining write code.</td>
</tr>
</tbody>
</table>

**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 \textit{in} for the \textit{Kind} property of a parameter corresponds to the \textit{ByVal} keyword.
- The value of \textit{inout} or \textit{out} for the \textit{Kind} property of a parameter corresponds to the \textit{ByRef} keyword.

\textbf{See Also}
- \texttt{Import Source Code}
- \texttt{Generate Source Code}
- \texttt{VB.Net Options}
9.5.10 **Visual Basic Conventions**

Enterprise Architect 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

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>global</td>
<td>Attribute</td>
<td>The <em>Global</em> keyword.</td>
</tr>
<tr>
<td>import</td>
<td>Operation</td>
<td>An operation to be imported from another library.</td>
</tr>
<tr>
<td>property get</td>
<td>Operation</td>
<td>A property get.</td>
</tr>
<tr>
<td>property set</td>
<td>Operation</td>
<td>A property set.</td>
</tr>
<tr>
<td>property let</td>
<td>Operation</td>
<td>A property let.</td>
</tr>
<tr>
<td>with events</td>
<td>Attribute</td>
<td>The <em>WithEvents</em> keyword.</td>
</tr>
</tbody>
</table>

### Tagged Values

<table>
<thead>
<tr>
<th>Tag</th>
<th>Applies to</th>
<th>Corresponds To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>Operation with stereotype <em>import</em></td>
<td>The alias for this imported operation.</td>
</tr>
<tr>
<td>attribute_name</td>
<td>Operation with stereotype <em>property get</em>, <em>property set</em> or <em>property let</em></td>
<td>The name of the variable behind this property.</td>
</tr>
<tr>
<td>Lib</td>
<td>Operation with stereotype <em>import</em></td>
<td>The library this import comes from.</td>
</tr>
<tr>
<td>New</td>
<td>Attribute</td>
<td>The <em>New</em> keyword.</td>
</tr>
</tbody>
</table>

### 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
- Visual Basic Options
Part X
The visual execution analyzer, or debug capability, is available in the Enterprise Architect Professional and Corporate editions.

Enterprise Architect enables you to build, test, debug, run and execute deployment scripts, all from within the Enterprise Architect development environment. Enterprise Architect 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 to integrate the test process directly into the Enterprise Architect IDE, you can now integrate UML and modeling into the build/test/execute/deploy process.

In addition to build/test and execute functionality, Enterprise Architect includes debugging capabilities for .NET, Java and Microsoft Native (C, C++ and Visual Basic) applications. The debuggers built into Enterprise Architect are specifically designed to enable the capture of stack trace information as a developer or tester ‘walks through’ the executing code, performing runtime inspection of suspended threads. The final stack trace history can then be used to generate Sequence diagrams within Enterprise Architect, converting the actual code execution and calls into visual diagrams. This capability provides an excellent means of managing complexity within a project, and 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 Enterprise Architect **Project** menu or from the context menu of a package in the **Project Browser**. It enables you to create and store custom scripts that specify how to build, test, run and deploy code associated with a package.

<table>
<thead>
<tr>
<th>Package Build Scripts...</th>
<th>Shift+F12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Ctrl+Shift+F12</td>
</tr>
<tr>
<td>Test</td>
<td>Ctrl+Alt+T</td>
</tr>
<tr>
<td>Run</td>
<td>Ctrl+Alt+N</td>
</tr>
<tr>
<td>Deploy</td>
<td>Ctrl+Shift+Alt+F12</td>
</tr>
</tbody>
</table>

Debug Run: F6

Debug Pause

Step Into: Shift+F6

Step Over: Alt+F6

Step Out: Ctrl+F6

Debug Stop: Ctrl+Alt+F6

Start Debug Recording

Stop Debug Recording

Auto Record Thread

Show/Hide Execution

Create Sequence Diagram

With the appropriate scripts set up Enterprise Architect can:

- Call a compiler to build your application, parse the compiler output 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.
10.1 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 can have multiple scripts, but only one is active at any one time. The Package Build Scripts dialog enables you to create and manage those scripts.

To access the Package Build Scripts dialog, either:
- On the Debug Workbench Toolbar, select the Build Scripts icon (the last icon on the right) and select the appropriate menu option, such as Build, or
- Select the Project | Build and Run | Package Build Scripts menu option, or
- Right-click on a package in the Project Browser, and select the Build and Run | Package Build Scripts menu option.

### Defined Scripts

<table>
<thead>
<tr>
<th>Active</th>
<th>Name</th>
<th>Build</th>
<th>Test</th>
<th>Run</th>
<th>Debug</th>
<th>Deploy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>Example 4</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>✔️</td>
<td>Example 5</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

![Defined Scripts Table]

Scripts are assigned to packages, and although a package might have only one active script at any time, you can assign multiple scripts and select from them as required. The Package Build Scripts dialog 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; the Build Script dialog displays.
- To modify an existing script, highlight the script name in the list and click on the Edit button.
- To copy a script with a new name, highlight the script name to copy and click on the Copy button; Enterprise Architect prompts you to enter a name for the new copy. Enter the new name in the dialog and click on the OK button. The new copy appears in the list and can be modified as usual.
- To delete a script, highlight the script name to delete, click on the Delete button, and click on the OK.
button.

- To export your scripts, click on the Export button to choose the scripts to export for this package.
- To import build scripts, click on the Import button to choose a .xml file of the scripts to import.

The Default Language field enables you to set the default language for generating source code for all new elements within this package and its descendents.

Select the Use Live Code Generation checkbox to update your source code instantly as you make changes to your model.

Select the Package is Namespace Root checkbox to set the source code namespace root (i.e. Java/C# namespace) to be the current package. Once you have set a namespace root, code generated beneath this root adds a package declaration at the head of the generated file indicating the current package location.

Click on the All Package Scripts button to open a new window that displays all scripts in the current project (see the next topic, Managing Scripts).

Once you have created new scripts or made any changes to existing ones, click on the OK button to confirm the changes, otherwise click on the Cancel button to quit the Package Build Scripts dialog without saving any changes.
10.1.1 Managing Scripts

The All Package Build Scripts dialog lists every script in the current project.

<table>
<thead>
<tr>
<th>Active</th>
<th>Package</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User Interface</td>
<td>Build</td>
</tr>
<tr>
<td></td>
<td>User Interface</td>
<td>Test with NUnit</td>
</tr>
<tr>
<td></td>
<td>Access Control</td>
<td>Build with output</td>
</tr>
</tbody>
</table>

To edit a script, double-click on its name or highlight the script and click on the Edit button. The Build Script dialog displays.

To delete a script, highlight the script and click on the Delete button. Enterprise Architect prompts you to confirm the deletion. Click on the Yes button to continue and delete the script.
10.1.2 Build Script Dialog

The Build Script dialog enables 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 the package should be deployed. You access this dialog by clicking on the:

- **Add** button on the Package Build Scripts dialog, or
- **Edit** button on the All Package Build Scripts dialog.

Each script requires a name and a working directory.

To use the tabs on this dialog, see the following topics:

- **Build Command**
- **Test Command**
- **Run Command**
- **Deploy Command**
- **Debug Command**
- **Sequence Tab Options**
10.1.3 Build Command

The Build tab enables you to enter a command for building the current package. This command is executed when you select the Project | Build and Run | Build menu option.

Write your script in the large text box using the standard Windows Command Line commands. You can specify, for example, compiler and linker options, and the names of output files. The format and content of this section depends on the actual compiler, make system, linker and so on that 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 you select the Capture Output checkbox, output from the script is logged in Enterprise Architect’s Output window. This can be activated by selecting the View | Output menu option.

The Output Parser field enables you to define a method for automatically parsing the compiler output. If you have selected the Capture Output checkbox, Enterprise Architect parses the output of the compiler so that by clicking on an error message in the Output window, you directly access the corresponding line of code.

Note:

The command listed in this field is executed as if from the command prompt. Therefore, if the executable path or any arguments contain spaces, they must be surrounded by quotes.

When you run the compile command inside Enterprise Architect, output from the compiler is piped back to the Output window and displayed as illustrated below:

```
Running build script - .Net 2.0 CPP Example 2
Microsoft (R) Visual Studio Version 8.0.50727.42.
Copyright (C) Microsoft Corp 1981-2005. All rights reserved.
========== Build: 0 succeeded, 0 failed, 1 up-to-date, 0 skipped ==========
Build completed with exit code 0
```

If you double-click on an error line, Enterprise Architect loads the appropriate source file and positions the cursor on the line where the error has been reported.
10.1.4 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 the Project | Build and Run | Test menu option.

Testing could be integrated with any test tool using the command line provided, but in these examples you can see how to integrate NUnit and JUnit testing with your source code. Enterprise Architect 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. Enterprise Architect does not include these products in the base installer.

The Capture Output checkbox enables Enterprise Architect to show the output of the program in the Output window, while the Output Parser field specifies what format output is expected. When parsing is enabled, double-clicking on a result in the Output window opens the corresponding code segment in Enterprise Architect's code window.

Selecting the Build before Test checkbox 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 that shows the Build before Test checkbox selected. As a result, every time the test command is given it runs the build script first.

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 the Build before Test checkbox selected, so the build script won't be executed before every test, but as a result it could test out of date code. This also shows the use of \%N, which is replaced by the fully namespace-qualified name of the currently selected Class when the script is executed.
Enter your test script below and select the appropriate output parser for the type of testing required.

`java junit.junit.TestRunner Main`
### 10.1.5 Run Command

Here you can enter a command for running your executable. This is the command that is executed when you select the **Project | Build and Run | Run** menu option. At its simplest, the script would contain the location and name of the file to be run.

**Note:**

Enterprise Architect provides the ability to start your application normally OR with debugging from the same script. The **Build and Run** menu has separate options for starting a normal run and a debug run.

The following two examples show scripts configured to run a .Net and a Java application in Enterprise Architect.

#### Build Script

**Name:** .Net 1.1 C#P Example 1  
**Directory:** C:\Benchmark\DotNet\cpplexample_net_1

Enter the path to the compiled application

```
c:\\benchmark\\cpplexample_net_1\\release\\example.exe
```

#### Build Script

**Name:** Java - Example 2  
**Directory:** C:\eclipse\Java\customer

Enter the path to the compiled application

```
customer
```

**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.
10.1.6 Debug Command

The Debug tab of the Build and Run dialog enables you configure how to debug your application within Enterprise Architect. Details on how to complete the fields for a variety of debugging scenarios are provided in the following platform-specific topics:

- Java
- .NET
- Microsoft Native
10.1.6.1 Java

**Option**

<table>
<thead>
<tr>
<th><strong>Use to</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application (Enter path)</strong></td>
</tr>
<tr>
<td>Identify the fully qualified Class name to debug, followed by any arguments. The Class must have a method declared with the following signature: public static void main(String[]); The debugger calls this method on the Class you name. In the example above, the parameters 1, 2 and 3 are passed to the method. You can also debug a Java application by attaching to an existing Java process.</td>
</tr>
<tr>
<td><strong>Enter any run time variables below</strong></td>
</tr>
<tr>
<td>Type any required 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 Debugging).</td>
</tr>
<tr>
<td>In the example above, a virtual machine is created with a new Class path property that comprises any paths named in the environment variable classpath 1603 plus the single path &quot;C:benchmark\java\examplel&quot;. If no Class path is specified, the debugger always creates the virtual machine 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.</td>
</tr>
<tr>
<td><strong>Show Console</strong></td>
</tr>
<tr>
<td>Create a console window for Java. If no console window is required, leave blank.</td>
</tr>
<tr>
<td><strong>Use Debugger</strong></td>
</tr>
<tr>
<td>Select Java.</td>
</tr>
</tbody>
</table>

### 10.1.6.1.1 Attach to Virtual Machine

You can debug a Java application by attaching to an existing Java process. However, the Java process requires a specific startup option specifying the Sparx Systems Java Agent. The format of the command line option is:

- `-agentlib:SSJavaProfiler71`
The example below is for attaching to the Tomcat Webserver. Select the **Attach to process** radio button, and then the keyword **Attach** is all that you have to enter. This keyword causes the debugger to prompt you for a process at runtime.

**Note:**
The **Show Console** checkbox has no effect when attaching to an existing virtual machine.

No run time variables are necessary when attaching.
10.1.6.2 .NET

**Table of Options:**

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application (Enter path)</td>
<td>Select and enter either the full or the relative path to the application executable, followed by any command line arguments.</td>
</tr>
<tr>
<td>Enter any runtime variables</td>
<td>Type any required command line options, if debugging a single .NET Assembly.</td>
</tr>
<tr>
<td>below</td>
<td></td>
</tr>
<tr>
<td>Show Console</td>
<td>Create a console window for the debugger. Not applicable for attaching to a process.</td>
</tr>
<tr>
<td>Use Debugger</td>
<td>Select the debugger to suit the .NET Framework under which your application runs.</td>
</tr>
</tbody>
</table>

**Note:**

If you intend to debug managed code using an unmanaged application, please see the Debug - CLR Versions topic.

10.1.6.2.1 Debug Assemblies

Enterprise Architect permits debugging of individual assemblies. The assembly is 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 following image is of a Build Script configured for debugging a .NET assembly.
Notice the **Enter any run time variables below** 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
- `method_name` is the unqualified name of the method belonging to the type that is invoked
- the `argument list` is optional depending on the method invoked.

The information in this field is passed to the debugger.

### 10.1.6.2.2 Debug - CLR Versions

Please note that if you are debugging managed code using an unmanaged application, the debugger might fail to detect the correct version of the Common Language Runtime (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` is the 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` is the version of the CLR targeted by your plugin or COM code.

10.1.6.3 Microsoft Native

You can also configure how to debug applications built in a Microsoft Native code. The example script below is configured to enable debugging of a C++ project built in Microsoft Visual Studio 2005.

You can debug native code only if there is a corresponding PDB file for the executable. You normally create this as a result of building the application with debug information.

The script must specify two things to support debugging:

- The path to the executable
- Microsoft Native as the debugging platform.

10.1.6.4 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 Enterprise Architect iteratively replaces the token with any subpath found under the root and executes the command again.
The output from this Java example is shown below:

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running build script - Java - Example 1</td>
</tr>
<tr>
<td>C:\Benchmark\Java\Example1 Build completed with exit code 0</td>
</tr>
<tr>
<td>C:\Benchmark\Java\Example1\common\draw build completed with exit code 0</td>
</tr>
<tr>
<td>C:\Benchmark\Java\Example1\common\toolbars Build completed with exit code 0</td>
</tr>
<tr>
<td>Note: source\Collection.java uses unchecked or unsafe operations.</td>
</tr>
<tr>
<td>Note: Recompile with -Xlint:unchecked for details.</td>
</tr>
<tr>
<td>C:\Benchmark\Java\Example1\source Build completed with exit code 0</td>
</tr>
</tbody>
</table>

**Note:**

The path being built is displayed along with the exit code.
10.1.7 **Deploy Command**

This section enables you to create a command for deploying the current package. These are the commands that are executed when you select the **Project | Build and Run | Deploy** menu option.

Write your script in the large text box using the standard *Windows Command Line* commands.

```
deploy.bat
```
10.1.8 Sequence Options

On the Build Script dialog for any model, the Sequence tab contains the following options:

- **Enable Filter**
- **Record arguments to function calls**
- **Record calls to external modules**
- **Capture state transitions using**

The options are not all available for each platform, as indicated in the following table:

<table>
<thead>
<tr>
<th>Option</th>
<th>.NET</th>
<th>Java</th>
<th>Native</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Filter</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record arguments</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record external calls</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Capture state transitions</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

10.1.8.1 Enable Filter

If the **Enable Filter** option is selected on the Sequence tab, the debugger excludes calls to matching methods from the generated sequence history and diagram. The comparison is case-sensitive.

To add a value, click on the **New** (Insert) icon in the right corner of the Filters box, and type in the comparison string. Each filter string takes the form:

```
class_name_token::method_name_token
```

The `class_name_token` excludes calls to all methods of a Class or Classes having a name that matches the token. The string can contain the wildcard character `*` (asterisk). The token is optional.

The `method_name_token` excludes calls to methods having a name that matches token. The string can contain the wildcard character `*`. The token is optional.

Where no Class token is present, the filter is applied only to global or public functions; that is, methods not belonging to any Class.

<table>
<thead>
<tr>
<th>To Filter</th>
<th>Use Filter Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>All public functions having a name beginning with <strong>Get</strong> from the recording session (GetClientRect for example in Windows API).</td>
<td>:::Get*</td>
</tr>
<tr>
<td>All methods beginning with <strong>Get</strong> for every Class member method.</td>
<td><em>::Get</em></td>
</tr>
<tr>
<td>All methods beginning with <strong>Get</strong> from the Class <strong>CClass</strong>.</td>
<td>CClass::Get*</td>
</tr>
<tr>
<td>All methods for Class <strong>CClass</strong>.</td>
<td>CClass:*</td>
</tr>
<tr>
<td>All methods for Classes belonging to Standard Template and Active Template Libraries.</td>
<td>• ATL*</td>
</tr>
<tr>
<td>All methods for Classes belonging to Standard Template and Active Template Libraries.</td>
<td>• std*</td>
</tr>
<tr>
<td>The specific method <strong>GetName</strong> for Class <strong>CClass</strong>.</td>
<td>CClass::GetName</td>
</tr>
</tbody>
</table>

In the Java example in the screen below, the debugger would exclude:

- Calls to **OnDraw** method for Class **Example.common.draw.DrawPane**
- Calls to any method of any Class having a name beginning with **Example.source.Collection**
- Calls to any constructor for any Class (i.e: `<clint>` and `<init>`).
In the Native code example below, the debugger would exclude:

- Calls made to Standard Template Library namespace
- Calls to any Class beginning with `TOb`
- Calls to any method of Class `CLock`
- Calls to any Global or Public Function with a name beginning with `Get`
- Calls to the method `GetLocation` for Class `Ctrain`
### 10.1.8.2 Record Arguments To Function Calls

When recording the sequence history, Enterprise Architect can record the arguments passed to method calls.

When the **Record Arguments to function calls** option is selected on the Debugger Sequence tab, the resulting Sequence diagram shows the values of elemental and string types passed to the method. See the following Java example.

Where the argument is not an elemental type, the type name is recorded instead.

### 10.1.8.3 Record Calls To External Modules

On the Debugger Sequence tab, the **Record calls to external modules** option causes function calls to external modules outside the model to be included in the sequence history and generated diagram.

Only calls originating within the model to functions external to the model are recorded.

**Note:**

External calls are displayed with a blue connector, as shown below.
This example shows an external call to the Microsoft .NET framework assembly function `System.Single.ToString`. 
10.2 Debugging

Enterprise Architect enables debugging of source code within Enterprise Architect, using a debug interface appropriate to the source language.

Debugging is configured by creating a debug script for the packages to be tested. One of the primary objectives of this feature is to enable 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.
10.2.1 System Requirements

Important:
Please read this information.

Supported Platforms
Enterprise Architect supports debugging on these platforms:

.Net
- Microsoft™ .NET Framework 1.1 and later
- Language support: C, C#, C++, J#, Visual Basic

Note:
Debugging under Windows Vista (x64) - If you encounter problems debugging with Enterprise Architect on a 64-bit platform, you should build a Win32 platform configuration in Visual Studio; that is, do not specify ANY-CPU, specify WIN32.

Java
- Java 2 Platform Standard edition (J2SE) version 5.0
- 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.

Windows for Native Applications
Enterprise Architect supports debugging native code (C, C++ and Visual Basic) compiled with the Microsoft™ compiler where an associated PDB file is available. Select Microsoft Native from the list of debugging platforms in your package script.

You can import native code into your model, and record the execution history for any Classes and methods. You can also generate Sequence diagrams from the resulting execution path.

Note:
Enterprise Architect currently does not support remote debugging.

Prerequisites
Creation of a Package Build Script and configuration of the Debug command in that script.
10.2.2 Debug 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 Enterprise Architect 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.

The following image shows the project tree of a web service imported into Enterprise Architect.

It is necessary to launch the client first, as the ASP .NET service process might 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, as follows:
When you start the debugger (click on the **Debug Workbench** Run button) the Attach To Process dialog displays.
Note 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 click on the OK button.

The Debugger Toolbar Stop button should be enabled and any breakpoints should be red, indicating they have been bound.

**Note:**

Some breakpoints might not 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 re-importing source code.

You can 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.
10.2.3 Debug COM interop

Enterprise Architect enables you to debug .NET managed code executed using COM in either a Local or an In-Process server. This feature is useful for debugging Plugins and ActiveX components.

1. Create a package in Enterprise Architect and import the code to debug. See Code Engineering.
2. Ensure the COM component is built with debug information.
3. Create a Script for the Package.
4. In the Debug tab, you can elect to either attach to an unmanaged process or specify the path to an unmanaged application to call your managed code.

5. Add breakpoints in the source code to debug.

Attach to an Unmanaged Process

- If an In-Process COM server, attach to the client process or
- If a Local COM Server, attach to the server process.

Click on the Debug Run button (or press [F6]) to display a list of processes from which you can choose.

Important:

Detaching from a COM interop process you have been debugging terminates the process. This is a known issue for Microsoft .NET Framework, and information on it can be found on many of the MSDN .NET blogs.
10.2.4 Debug Another Process

If the **Build Script Debug** command is set to the **attach** keyword, when you click on the **Debug Run** button or press **[F6]** the **Attach to Process** dialog displays, from which you can select the process to attach to.

Once Enterprise Architect is attached to the process, any breakpoints encountered are detected by the debugger and the information is available in the **Debug** windows.

To detach from a process, click on the **Debug Stop** button.
10.2.5 **Debug Java Web Servers**

This topic describes the configuration requirements and procedure for debugging Java web servers such as JBOSS and Apache Tomcat (both Server configuration and Windows Service configuration) in Enterprise Architect.

The procedure involves attaching to the process hosting the Java Virtual Machine from Enterprise Architect, as summarized below:

1. Ensure binaries for the web server code to be debugged have been built with debug information.
2. Launch the server with the Virtual Machine startup option described in this topic.
3. Import source code into the Enterprise Architect Model, or synchronize existing code.
4. Create or modify the Package Build Script to specify the Debug option for attaching to the process.
5. Set breakpoints.
6. Launch the client.
7. Attach to the process from Enterprise Architect.

**Server Configuration**

The configuration necessary for the web servers to interact with Enterprise Architect must address the following two essential points:

- Any VM to be debugged, created or hosted by the server must have the Sparx Systems Agent SSJavaProfiler command line option specified in the VM startup option (that is: \-agentlib:SSJavaProfiler)
- The CLASSPATH, however it is passed to the VM, must specify the root path to the package source files. The Enterprise Architect debugger uses the `java.class.path` property in the VM being debugged, to locate the source file corresponding to a breakpoint occurring in a Class during execution. For example, a Class to be debugged is called:
  
  a.b.C

  This is located in physical directory:
  
  C:\source\a\b
  
  So, for debugging to be successful, the CLASSPATH must contain the root path:

  c:\source.

**Package Script Configuration**

Using the Debug tab of the Build Script dialog, create a script for the code you have imported and specify the following:

- Select the Attach to process radio button, and in the field below type attach.
- In the Use Debugger field, click on the drop-down arrow and select Java.
All other fields are unimportant. The **Directory** field is normally used in the absence of any Class path property.

**Debugging**

First ensure that the server is running, and that the server process has loaded the Sparx Systems Agent DLL SSJavaProfiler71.DLL (use *Process Explorer* or similar tools to prove this).

Launch the client and ensure the client executes. This must be done before attaching to the server process in Enterprise Architect.

After the client has been executed at least once, return to Enterprise Architect, open the source code you imported and set some breakpoints.

Click on the **Run Debugger** button in Enterprise Architect. The *Attach To Process* dialog displays.
Click on the **OK** button. A confirmation message displays in the Debug Toolbar Output Tab, stating that the process has been attached.

The breakpoints should remain enabled (bright red). If the breakpoints fail, and contain either an exclamation mark or a question mark, then either the process is not hosting the SSJavaprofiler Agent or the binaries being executed by the server are not based on the source code. If so, check your configuration.
10.2.5.1 JBOSS Server

Consider the JBoss example below. The source code for a simple servlet is located in the directory location:

```
| Address | C:\Benchmark\Java\JBOSS\Inventory\com\inventory\dto |
| Folders | × | Name |
| Desktop | My Computer |
| - Local Disk (C:) | - Benchmark |
| | - DotNet |
| | - Interop |
| | - Java |
| | - Customer |
| | - Eclipse |
| | - Example1 |
| | - FleetRentalThreadedServer |
| | - JBOSS |
| | - Inventory |
| | - com |
| | - inventory |
| | - dao |
| | - dto |
```

The binaries executed by JBOSS are located in the JAW.EAR file in this location:

```
| Address | C:\JBOSS\C3b-dae\build\distribution |
| Folders | × | Name |
| JBOSS | 03b-dao |
| | build |
| | distribution |
| | common |
```

The Enterprise Architect debugger has to be able to locate source files during debugging. To do this it also uses the CLASSPATH, searching in any listed path for a matching JAVA source file, so the CLASSPATH must include a path to the root of the package for Enterprise Architect to find the source during debugging.

The following is an excerpt from the command file that executes the JBOSS server. Since the Class to be debugged is at com/inventory/dto/carDTO, the root of this path is included in the JBOSS classpath.

```
RUN.BAT
----------------
set SOURCE=C:\Benchmark\Java\JBOSS\Inventory
set JAVAC_JAR=%JAVA_HOME%/lib/tools.jar
if "%JBOSS_CLASSPATH%" == ""
{
    set JBOSS_CLASSPATH=%SOURCE%;%JAVAC_JAR%;%RUNJAR%;
}
else
```

Enterprise Architect User Guide
(  
  set JBOSS_CLASSPATH=%SOURCE%;%JBOSS_CLASSPATH%;%JAVAC_JAR%;%RUNJAR%;
)
set JAVA_OPTS=%JAVA_OPTS% -agentpath:\"C:\program files\sparx systems\ssjavaprofiler71\"  

10.2.5.2 Apache Tomcat Server

This configuration is for the same application as outlined in the JBOSS server configuration topic. There are two things to notice of importance.

- The Java VM option: -agentpath:C:\program files\sparx systems\ea\ssjavaprofiler71
- The addition to the Class path property of the path to the source code: C:\JBOSS\03b-dao\common\src;
10.2.5.3 Apache Tomcat Windows Service

For users running Apache Tomcat as a Windows™ service, it is important to configure the service to enable interaction with the Desktop. Failure to do so causes debugging to fail within Enterprise Architect.

Select the Allow service to interact with desktop checkbox.
10.2.6 Debug Internet Browser Java Applets

This topic describes the configuration requirements and procedure for debugging Java Applets running in a browser from Enterprise Architect.

The procedure requires you to attach to the browser process hosting the Java Virtual Machine (JVM) from Enterprise Architect, as summarized below:

1. Ensure binaries for the applet code to be debugged have been built with debug information.
2. Configure the JVM using the Java Control Panel.

3. In the Java Applet Runtime Settings panel, click on the View button. The Java Runtime Settings dialog displays.

4. Click on the appropriate entry and click on the OK button to load the Sparx Systems Agent.

5. Import source code into the Enterprise Architect model, or synchronize existing code.

6. Create or modify the Package Build Script to specify the Debug option for attaching to the process.

5. Set breakpoints.
6. Launch the browser.
7. Attach to the browser process from Enterprise Architect.

Note that the class.path property specified for the JVM includes the root path to the applet source files. This is necessary for the Enterprise Architect debugger to match the execution to the imported source in the model.
10.2.7 Use the Debugger

The debugging components of Enterprise Architect comprise the Debug Toolbar and a tabbed Debug Workbench. To display these, either press [Alt]+[8] or select the View | Debug Workbench menu option.

If a Debug script has been configured for the currently selected package, the Run button is enabled. If no script has been created or is incomplete, all buttons are disabled and debugging remains unavailable.

As Build Scripts are linked to a package, selecting a package in the Project Browser also changes the active Build Script and, naturally, the target to be debugged.

Breakpoints are recorded against the package also, and these update depending on which package is selected.

Start The Debugger

To start debugging click on the Run button on the toolbar or press [F6].

Note:

The debugger only stops if it encounters breakpoints, so these should be set beforehand.

The figure below shows a view of Enterprise Architect where the debugger has been invoked for a .NET application written in C#.

Stop the Debugger

To stop debugging click on the Stop button or press [Ctrl]+[Alt]+[F6].
Note:
In most situations, debugging ends when the debug process terminates or the Java Class thread exits. However, due to the nature of the Java Virtual Machine, it is necessary at times for Java developers to stop the Debugger manually with the Stop button.

10.2.7.1 The Debug Toolbar

The Debug toolbar hosts all the important commands you use during a debug session.

Two of the buttons toggle: the Pause/Resume button (2) and the Enable/Disable Editor Update button (20).

From left to right the buttons are as follows:

1. Run Debugger [F6] Both initiates debugging and continues execution of a thread that has been halted by a breakpoint.
2. Pause/Resume debug execution Enables you to temporarily halt execution of every thread in the process being debugged, or resume execution after it has been paused. Clicking on any command button also resumes execution. The Pause button displays in red while execution is suspended.
   **Note:** Pausing execution undermines the validity of the stack trace and local variable information displayed in any of the debug windows; after pausing, this information should not be relied upon.
3. 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.
4. Step Over [Alt]+[F6] 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; i.e. the stack depth is greater than one.
5. Step Through Is most useful while recording a debug session. The command has the same behaviour as the StepOver command, except that it records intermediate method calls made between lines. If you are recording and you press the StepOver button, any method called between lines is not recorded. The Step Through command causes the debugger to step into any method executed at the current line, and any subsequent method called when the command is issued. The debugger continues to step until it arrives back at the next line.
6. Step In [Shift]+[F6] Instructs the debugger to try and step over every line of physical code that the thread is executing. However, the debugger ignores any call to a namespace, function class or method that does not exist in the model. That is, only code either generated within Enterprise Architect or imported into the model is executed.
7. Step Out [Ctrl]+[F6] Instructs the debugger to run until the current method exits and, if the method has a caller (stack depth > 1), to stop at the next line of code in the calling method.
8. Show Execution Point Only enabled when a thread has encountered a breakpoint. The command 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. If no thread is selected, the first suspended thread has the focus.)
9. Show Break Points Displays all current breakpoints. Breakpoints are linked to the Project Browser.
tree, so changing view by clicking in the tree changes the breakpoints displayed, if any. The only time the breakpoints do not reflect selection changes in the Project Browser is during a debug session.

10 Delete Break Points Delete all breakpoints for the currently active view. If debugging, the breakpoints are removed.

11 Disable All Break Points Disables all break points for the current view. They are not deleted, so you can re-enable them.

12 Enable All Break Points Re-enables any disabled break points.

13 Show Local Variables Selects the Locals tab, and shows 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.

14 Show Call Stack Selects the Stack tab and shows all currently running threads. The call stack for the any suspended thread is displayed.

15 Record Stack Trace Starts recording the trace history. Recording occurs for the currently suspended thread. The stack history is cleared ready to accept the new trace history.

Note:
This option is also available from the shortcut menu on the Stack tab.

16 Auto Record Stack Trace Automatically records a stack trace for a selected thread. All other threads are ignored, although entries appear for all thread creations and terminations. The Step In command is issued automatically until either a breakpoint is encountered or the thread terminates. If a breakpoint is encountered, the debugger halts. To continue automatic recording click on any continue command (Run, StepIn, StepOut, StepOver) and Stack Trace recording is resumed.

Note:
This option is also available from the shortcut menu on the Stack tab.

17 Stop Recording Stops recording into the stack trace history.

Note:
This option is also available from the shortcut menu on the Stack tab.

18 Create Sequence Diagram Creates a sequence diagram from a recorded Stack Trace history. Give your diagram a name and it is placed in the package for this debug session.

19 Save Saves recorded history to an HTML file for viewing in a browser, or to an XML file for opening later in Enterprise Architect.

20 Enables and updates display of the editor window with the current line of executing source code. During automatic recording of a thread, time is spent updating the editors with the current line of executing code. To speed up execution in debugging larger applications, you can disable this highlighting in any editors.

21 Accesses a menu that enables 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
10.2.7.2 Runtime Inspection

During debugging, whenever a thread is suspended at a line of execution, you can inspect member variables in the Editor window.

To evaluate a member variable, use the mouse to move the cursor over the variable in the Editor window, as shown in the following examples.

```java
public void Print()
{
    int n = 0;
    while (names[n].Length > 0)
    {
        Document d = new Document(names[n++]);
        d.Print();
    }
}
```

```java
public void Print()
{
    int n = 0;
    while (int 32-bit signed integer r=0)
    {
        Document d = new Document(names[n++]);
        d.Print();
    }
}
```

10.2.7.3 The Debug Workbench Window

The Debug Workbench window is available through the View | Debug Workbench menu option, or by pressing [Alt]+[8]. It contains seven tabs:

- Breakpoints
- Locals
- Stack
- Output
- Recording History
- Workbench
- Module

**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, select the View | Source Code menu option or press [Alt]+[7]. The Source Code window displays. Find the line on which to place the breakpoint in the source and click in the column to the left of the line-numbers. A round circle displays in that column to indicate that a breakpoint has been placed there (see the Breakpoints topic).

**Locals**
This display shows the local variables defined in the current code segment, their type and value. (See the Local Variables topic.)

**Stack**
This view shows the position of the debugger in the code. Clicking on the > button advances the stack through the code until the next breakpoint is reached. (See the Stack topic.)

**Output**
Displays output from the debugger including any messages output by debugged process, such as writes to standard output. (See the Output topic.)

**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 the Recording a Debug session topic.

**Modules**
This tab displays all the modules loaded during a .Net debug session.

**Workbench**
This display is a place to create your own variables and invoke methods. See the Workbench topic.

### 10.2.7.3.1 Breakpoints

Breakpoints work in Enterprise Architect much like in any other debugger. Adding 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 is displayed in an editor window, and the line of code where the breakpoint occurred is highlighted. You can enable and disable breakpoints, or delete them from the script altogether.

An Enterprise Architect model maintains breakpoints for every package having a Build Script - Debug command. Breakpoints are displayed in a tab of the Debugger Workbench window (press [Alt]+[8]). Selecting a different package in the tree updates the breakpoints displayed, depending on whether the node selected, or its parent, has a script attached.

**Note:**
The debugger does not stop automatically. It runs 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 Enterprise Architect user

### Breakpoint States

<table>
<thead>
<tr>
<th>Debugger State</th>
<th>Running</th>
<th>Not running</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active breakpoint</td>
<td>Enabled breakpoint</td>
<td></td>
</tr>
<tr>
<td>Unbound breakpoint</td>
<td>N/a</td>
<td></td>
</tr>
<tr>
<td>Failed breakpoint</td>
<td>N/a</td>
<td></td>
</tr>
<tr>
<td>Disabled breakpoint</td>
<td>Disabled breakpoint</td>
<td></td>
</tr>
</tbody>
</table>

### Add Breakpoints

To set a breakpoint for a code segment, open the model code 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.

If the code is currently halted at a breakpoint, that point is indicated by a yellow arrow within the red circle.

```csharp
67 [System::Web::Services::WebMethod]
68 float Add( float a, float b)  
69 {  
70     return a + b;  
71 }  
72 [System::Web::Services::WebMethod]
73 float Multiply( float a, float b)  
74 {  
75     return a * b;  
76 }  
77 [System::Web::Services::WebMethod]
78 float Subtract( float a, float b)  
79 {  
80     return a - b;  
81 }  
82 [System::Web::Services::WebMethod]
83 float Divide( float a, float b)  
84 {  
85     return a / b;  
86 }  
```

### Delete, Disable and Enable Breakpoints

To delete a specific breakpoint, either:

- If the breakpoint is enabled, click on the red breakpoint circle in the left margin of the Source Code Editor,
- or
- Select the breakpoint in the Breakpoints tab and press [Delete].
You can also delete all breakpoints by clicking on the **Delete all breakpoints** button on the **Debug toolbar**.

To disable a breakpoint, deselect its checkbox on the **Breakpoints** tab. It is then shown as an empty grey circle. Select the breakpoint again to enable it.

### 10.2.7.3.2 Local Variables

Whenever a thread encounters a breakpoint, this window displays 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, as shown in the following screen:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>args</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>args[0]</code></td>
<td>&quot;a #&quot;</td>
<td>String</td>
</tr>
<tr>
<td><code>args[1]</code></td>
<td>&quot;example&quot;</td>
<td>String</td>
</tr>
<tr>
<td><code>args[2]</code></td>
<td>&quot;using&quot;</td>
<td>String</td>
</tr>
<tr>
<td><code>args[3]</code></td>
<td>&quot;NET 1.1&quot;</td>
<td>String</td>
</tr>
<tr>
<td><code>P</code></td>
<td></td>
<td>Example.Program</td>
</tr>
<tr>
<td><code>pdoc</code></td>
<td></td>
<td>Document</td>
</tr>
<tr>
<td><code>cites</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>names</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local variables are 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

### 10.2.7.3.3 Stack

The stack view shows all currently running threads. Any thread that is at a breakpoint displays a stack trace. Clicking on the **Run Debugger** button continues the thread until 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 arrow indicates 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 on the thread entry to select that thread
- Selecting a thread results in that thread being displayed in orange; the **Source Code Editor** also changes to reflect the current line of code for that thread
- Double-clicking a frame takes you to that line of code in the **Source Code Editor**; local variables are also refreshed for the selected frame.
10.2.7.3.4 Output

During a debug session the Debugger emits messages detailing both startup and termination of session, to its Output tab. Details of exceptions and any errors are also output to this tab. Any trace messages such as those output using Java System.out or .NET System.Diagnostics.Debug are also captured and displayed here.

```
Stack

Thread 00000000
- Form1::InitializeComponent, c:\debugging\cpp\vs2003_threading\Form1.h, line 176
- Form1::ctor, c:\debugging\cpp\vs2003_threading\Form1.h, line 145
- WinMain, c:\debugging\cpp\vs2003_threading\Form1.cpp, line 15
```

10.2.7.3.5 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 Using Breakpoints.

Columns

- **Sequence** - unique sequence number

**Note:**

The checkbox against each number is used to control whether or not this call should be used to create a Sequence diagram from this history. In addition to enabling or disabling the call using the checkbox, you can use context menu options to enable or disable an entire call, all calls to a given method, or all calls to a given Class.

- **Thread** - operating system thread ID
- **Method** - Class and method names
- **Source** - filename and line number
- **Direction** - Stack Frame Movement, either Call, Return, Breakpoint or Escape (Escape is used internally when producing Sequence diagram to mark end of an iteration)
- **Depth** - stack depth at time of call; used in generation of sequence diagrams.
The Workbench is a tool in Enterprise Architect Debugging, enabling you to create your 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.

Platforms Supported

The Workbench supports the following workbench platforms:

- Microsoft .NET (version 2.0 or later)
- Java (JDK 1.4 or later)

Note:
The Workbench does not currently support the creation of Class instances written in native C++, C or VB.

Mode

The 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 change if the debugger encounters any breakpoints. If no breakpoints are set, then the variables do not change. The Workbench immediately returns to Idle mode.

Logging

The result of creating variables and the result of calls on their methods is displayed in the Output tab.

10.2.7.3.6.1 Workbench Variables

You can create and delete workbench variables from any Class in your model. When you do so, you are asked to name the variable. It then displays in the Workbench tab of the Debug Workbench 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 those 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 (see the Debug Tab topic).

Constraints (.NET)

- Members defined as struct in managed code are not supported.
- Classes defined as internal are not supported.

Delete Workbench Variables

You can delete variables using the Delete shortcut menu on any instance on the Workbench. If all instances are deleted the debugger is shut down, and the Workbench is closed.

10.2.7.3.6.2 Create Workbench Variables

Right-click on the required Class node in the Project Browser and select the Create Workbench Instance menu option, or press [Ctrl]+[Shift]+[J]. The menu option is also available from within a Class diagram.

Naming the Workbench

When you elect to create an instance of a type Enterprise Architect prompts you with the Workbench dialog 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 you do not define a constructor, or define a single constructor taking no arguments, the default constructor
or the defined constructor is automatically invoked.

Otherwise the following dialog displays. Select the constructor from the drop-down list and fill in any parameters required.

![Constructor Dialog](image)

**Arguments**

In the dialog above, type any parameters required by the constructor.

- **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 checks any name entered in an argument against its list of workbench instances, and substitutes 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; for example, the only time you should 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.
  
<table>
<thead>
<tr>
<th>Type</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>String[]</td>
<td>one,two,three,&quot;a book&quot;,&quot;a bigger book&quot;</td>
</tr>
<tr>
<td>CPerson[]</td>
<td>Tom,Dick,Harry</td>
</tr>
</tbody>
</table>

**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, click on the **Invoke** button to create the variable. Output confirming this action is displayed in the **Output** tab.

10.2.7.3.6.3 **Invoke Methods**

On the **Workbench** tab, right-click on the instance on which to execute a method.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rob</td>
<td>MyClassLibrary:CRobert</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>0</td>
<td>32-bit floating point</td>
</tr>
<tr>
<td>Reset workbench</td>
<td>0</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>Age</td>
<td>2</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>Friends</td>
<td>MyClassLibrary:CPerson</td>
<td></td>
</tr>
<tr>
<td>Town</td>
<td>&quot;Daylesford&quot;</td>
<td>String</td>
</tr>
<tr>
<td>Name</td>
<td>&quot;Robert&quot;</td>
<td>String</td>
</tr>
<tr>
<td>Fred</td>
<td>MyClassLibrary:CJohn</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>MyClassLibrary:CJohn</td>
<td></td>
</tr>
</tbody>
</table>

**Choose Method**

A list of methods for the type are presented in a dialog. Select a method from the list. Note that all methods listed are public; private methods are not available.
Supply Arguments

In this example, you have created an instance or variable named Rob of type MyClassLibrary.CRobert. You now invoke a method named AddFriends, which takes an array of CPerson objects as its only argument. What you supply to it are the two other Workbench instances Fred and John.
10.2.8 Generate Sequence Diagrams

With Enterprise Architect you can easily create detailed and comprehensive Sequence diagrams from your recorded debug sessions. You can generate a Sequence diagram:

- **For a method (operation)** of a specific Class (the simplest process), or
- **By stepping through your executing code** and recording specific execution traces between breakpoints, either manually or automatically, or between recording markers.

Enterprise Architect takes the recorded stack history captured during one of these runs and automatically builds the Sequence diagram, including compacting looping sessions for easy reading. You can control aspects of the diagram generation by setting options on the Sequence tab of the Build Script dialog, or after recording execution traces when the Generate Sequence Diagram dialog displays.

If required, you can prepare your project and debugger to show State transitions in the generated Sequence diagrams.

Before recording execution, you must set up your debugger and run and debug your application successfully.

The following diagram illustrates the kind of diagram you can produce by carefully stepping through your running program or letting Enterprise Architect profile it automatically.
10.2.8.1 Record Debug Session For a Method

In the Professional edition of Enterprise Architect, you can automatically generate a Sequence diagram for a method (operation) of a particular Class, using the method’s context menu on the Project Browser.

This menu option is enabled when a script has been configured for the package containing the operation method.

This is the most straightforward technique for creating a Sequence diagram. The debugger executes the application specified in the build script for the parent package, and when the selected method is encountered during execution, the debugger records all the activity from that point. When the method exits, the debugger stops and produces a diagram from the history for that method. If the method is not executed then no diagram is generated.

Note:

If the debugger is already running, the menu option is disabled (grayed). You must stop the debugger to re-enable the menu option.

10.2.8.2 Record a Debug Session Using Breakpoints

The debugger enables you to record your debug session and create Sequence diagrams from the Stack Trace History. Recording can only occur for a given thread. The Debug Toolbar record buttons (Record and Autorecord) become enabled whenever a thread is available for recording. This occurs when a thread encounters a breakpoint, and becomes suspended.

For most applications it is typical that you would set breakpoints at the start and end points in the code for which to generate the diagram.

You can begin recording in either of the following ways:

- Manual recording for a thread
- Automatic recording for a thread

The following example shows a debug sequence recorded for a thread executing managed C++ code under Microsoft .Net 1.1.
Create 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, click on the Create Sequence Diagram button on the Debug Toolbar and define your diagram. The diagram and related artifacts are placed in an interaction under the package that is running your debug session.
Debug Toolbar Commands

- Create Sequence Diagram.
- Save recorded history to HTML file for viewing in browser.

10.2.8.2.1 Record For a Thread Manually

Either:
- Click on the Record button, or
- Right-click on the Stack tab to display the context menu, and select the Record option.

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 is logged. When you have finished tracing, click on the Stop button. Generate the diagram.

© 1998-2008 Sparx Systems Pty Ltd
**Debug Toolbar Commands**

- Record Stack Trace for Thread.
- Stop Recording.

**10.2.8.2.2 Record For a Thread Automatically**

Either:
- Click on the **Autorecord** button, or
- Select the **Stack** tab, right-click to display the shortcut menu, and choose the **Auto Record** option.

The Stack Trace History, Stack tab and Source Code Editor 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 ends when the thread ends, or when you click on the **Stop** button. Generate the diagram.

**Debug Toolbar Commands**

- Auto Record Stack Trace For Thread
- Stop Recording

**10.2.8.3 Recording Markers**

Trace marking is a feature that enables you to silently record code executed between two points, and incorporate it in a Sequence diagram. The feature also enables you to capture the execution of multiple threads. It can be particularly useful in capturing event driven sequences (such as mouse and timer events) without any user intervention.

The recording markers are breakpoints; however, instead of stopping, the debugger behaves according to the type of marker. If the marker is denoted as a recording **start point**, the debugger immediately begins to trace all executed calls from that point for the breaking thread. Recording is stopped again when either the thread that is being captured terminates or the thread encounters a **recording stop point**.

Recording markers are set in the source code editor. If you right-click on the breakpoint margin at the point to begin recording, a context menu displays:
Select the **Start recording here** option, then right-click on the breakpoint margin at the point to stop recording and select the **Stop recording here** option. The markers are shown below:

```
static void main(int argc, TCHAR* argv[])
{
    if(argc>1)
    {
        Painter p;
        TString option(argv[1]);
        if(option == "primary")
            p.Primary();
        else if(option == "compound")
            p.Compound();
        else if(option == "shade")
            p.Shade();
        else

When the debugger is run it continues to run the thread, recording a stack history, until either the **Stop** marker is encountered or the thread terminates, unlike normal breakpoints where the debugger halts and displays the line of code.

Generate the diagram

### 10.2.8.4 Generate the Sequence Diagram

Once you have built up a stack trace history using breakpoints, you can create a Sequence diagram from your results. To do this, click on the **Generate Sequence Diagram** button on the Debug Toolbar. The Generate Sequence Diagram dialog displays.

![Generate Sequence Diagram](image)

The dialog also displays automatically at the end of a debug recording session, when a thread you are recording terminates or the program finishes.

Complete the fields as follows:
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type in the diagram name; this field defaults to the name of the State Machine.</td>
</tr>
<tr>
<td>Include state information if available</td>
<td>Include any State transitions detected during the debug run in the Sequence diagram produced. The checkbox is enabled only if the State Transitions option has been enabled for the Package Script prior to debugging.</td>
</tr>
<tr>
<td>Highlight invalid transitions</td>
<td>Mark with a red border any state transitions detected that are illegal for the State Machine. The checkbox is enabled only if Include state information if available is selected.</td>
</tr>
</tbody>
</table>

Click on the OK button. Your diagram and related artifacts are placed in an interaction under the package that is running your debug session.
Debug Toolbar Commands

- Create Sequence Diagram.
- Save recorded history to HTML file for viewing in browser.

10.2.8.5 Include State Transitions

Overview

You can generate Sequence diagrams that show transitions in state as a program executes. The illustration below shows a simple project that has, in its State Machine, a number of States that correspond to color values of the Painter Class member variable Paint. Notice also that fragments take into account changes in State.
Example Sequence Diagram with State Transitions For Class Painter

Showing State transitions on your debug-generated Sequence diagrams is optional; you set an option in the package script associated with the Class for which you intend to record States.

Note:

If you do not have a package script for the Class or package you must create one. Sequence diagrams can only be generated for a package that has been configured for debug.

Next, you create a State Machine under the Class. On the State Machine you create the State elements that correspond to any states to be captured for your Class. The debugger evaluates your States by checking constraints on the States you create. The States on this diagram are then used by the debugger and State transitions are incorporated into the diagram. These procedures are explained in the subsequent sections, with reference to the following figure:

This figure shows the Class Paint with a State Machine called Compound. It has a child diagram also called Compound, on which the States {Grey, Lime, Olive...} are placed.

10.2.8.5.1 Enable Capture of State Transitions

To enable capture of State transitions on your Sequence diagram, follow the steps below:

1. Select the Package Build Scripts option either from the Build and Run context menu for the package on the Project Browser, or from the Debug Workbench toolbar drop down menu.
2. Open your build script and select the Sequence tab.
3. Ensure the Capture state transitions using checkbox is selected.
4. Click on the drop-down arrow and select the required constraint type to limit the states captured by the debugger to those having constraints of that type. If you select <any> then any state found for the Class-associated State Machine is included in recording.

10.2.6.5.2 Create a State Machine and States

Create State Machine
Open a diagram in the required package and drag a State Machine element from the Enterprise Architect UML Toolbox. In the Project Browser, drag the State Machine under the Class it represents. The State Machine itself has an empty child diagram. Open that diagram.

Create States
The following illustration shows the States on the example State Machine Compound from the Class Painter, as shown in the Overview.

In order to evaluate a State for any object you must have a way to link the State to the Class of the object. You do this by adding constraints to the State. Each constraint is in effect a condition on member variables of the Class.
10.2.8.5.3 Add Constraints

The State Properties dialog below is for the State Turquoise. The Constraints tab is open to show how the State is linked to the Class Painter. A State can be defined by a single constraint or by many; in the example below the State Turquoise has two constraints.

The Painter Class has a member called Paint of type Colour. The Paint member has two member variables called primary and secondary. These variables are enums. The values of constraints can only be compared for elemental, enum, and string types. In native code, equivalence of Class objects is only possible for pointer types.

What this constraint means is:

- when an instance of the Painter Class exists and
- its member variable Paint's member variable primary has the enum value Blue and
- the member variable secondary has the enum value Green then
- this State is evaluated to true.

You can see the member variables for these constraints in the following illustration of the Class Painter, expanded to show members. Notice the enumeration TColor values. These are the values tested in the constraints in the State Properties dialog above.
Operators in Constraints

There are two types of operators you can use on constraints to define a State:

- Logical operators AND and OR can be used to combine constraints
- Equivalence operators (= and !=) can be used to define the conditions of a constraint.

All the constraints for a State are subject to an AND operation unless otherwise specified. You can use the OR operation on them instead, so you could rewrite the constraints in the earlier example as:

```
Paint.primary=Blue OR Paint.secondary=Green
```

Below are some examples of using the equivalence operators:

```
Paint.primary!=Blue AND Paint.secondary!=Green
Paint.primary=Blue OR Paint.primary=Green
Person.FirstName=John
Person.LastName=Carpenter
```

**Note:**

Quotes around strings are optional. The comparison for strings is always case-sensitive in determining the truth of a constraint.
10.3 Unit Testing

Enterprise Architect supports integration with unit testing tools in order to make it easier to develop good quality software.

Firstly, Enterprise Architect 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 test results are automatically recorded inside Enterprise Architect.
10.3.1 Set Up Unit Testing

In order to use unit testing in Enterprise Architect, you must first set it up. This happens in two parts.

Firstly the appropriate tests must be defined. Enterprise Architect 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 an NUnit example in C# that is followed through the rest of this topic, although it could also be any other .Net language or Java and JUnit.

```csharp
[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 Enterprise Architect so that Enterprise Architect 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 to run the tests. These scripts must be set up against a package.

The sample above can be called by setting up the Package Build Scripts dialog as follows.

If Enterprise Architect is to handle unit testing, it is important that you select the Capture Output checkbox and select the appropriate Output Parser for the testing. Without doing this you won't see the program output and therefore you cannot open the source at the appropriate location.
10.3.2 Run Unit Tests

You can run the test script you set up previously, by selecting the Project | Build and Run | Test menu option.

The following output is generated.

```
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
.
.
Tests run: 4, Failures: 1, Not run: 0, Time: 0.031 seconds

Failures:
1) CalculatorTest.testSubtract :
   expected:<0>
   but was:<1>
   at NUnit.Framework.EqualComparer FailNotEqual()
   at NUnit.Framework.EqualComparer.Assert()
   at CalculatorTest.testSubtract() in C:\SimpleCalculator\SimpleCalculatorTest.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. If you double-click on that error, Enterprise Architect opens the editor to that line of code.

This enables you to quickly find and fix the error.

Enterprise Architect also records the run status of each test as described in Record Test Results.
10.3.3 Record Test Results

Enterprise Architect is able to automatically record all results from tests by a testing script in Enterprise Architect. In order to use this feature, you just 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 Enterprise Architect adds test cases to the Class for each test method found. On this and all subsequent test runs all test cases are updated with the current run time and if they passed or failed as shown in the following illustration.

The error description for each failed test is added to any existing results for that test case, along with the current date and time. Over time this provides a log of all test runs where each test case has failed. This can then be included in generated documentation and could resemble 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>
Part XI
11 XML Technologies

Enterprise Architect enables rapid modeling, forward engineering and reverse engineering of two key W3C XML technologies:

- **XML Schema** (XSD)
- **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 2.1 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)**
11.1 XML Schema (XSD)

Enterprise Architect supports Forward and Reverse engineering of W3C XML schemas (XSD). The following topics explain how to use Enterprise Architect to model, generate and import XML schemas:

- Model XSD
- Import XSD
- Generate XSD
11.1.1 Model XSD

XML schemas are modeled using UML Class diagrams. The XML Schema pages of the Enterprise Architect UML Toolbox provide in-built support for the UML profile for XSD. This enables an abstract UML Class model to be automatically generated as a W3C XML Schema (XSD) file.

The following Class diagram 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 code shows the schema 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
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ContactInfo" ...
```
The following topics provide further explanation:

- **UML Profile for XSD**
- **XSD Datatypes Package**
- **Abstract XSD Models**

### 11.1.1.1 UML Profile for XSD

The UML Profile for XSD specifies a set of stereotypes, Tagged Values and constraints that can be applied to the UML model in order to change particular aspects of the resulting schema. For example, you might have to convert certain UML Class attributes to XSD attributes, or use a model group other than the default *Sequence*.

Enterprise Architect provides native support for the UML Profile for XSD via the XML schema pages of the Enterprise Architect UML Toolbox. Alternatively, you can use the profile via Enterprise Architect's generic profile mechanism by downloading the UML Profile for XSD. See the **Using Profiles** topic for details on importing UML profiles into Enterprise Architect. The XSD profile used by Enterprise Architect is an adaptation of the profile defined in *Modeling XML Applications with UML* (David Carlson).

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. Full information on the Tagged Values can be obtained from the W3C XML Schema recommendation. The constraints define any conditions that must be satisfied for the stereotype to apply.

The following stereotypes are provided:

- **XSDschema**
The following tables list the features of the UML Profile for XSD. Tagged Value names are shown in bold followed by the allowed values. If a default value is used by Enterprise Architect's schema generator, it is underlined.

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>All Classes in a package are defined within one schema. This stereotype can be used to specify schema-wide settings.</td>
</tr>
</tbody>
</table>

| 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 that identifies the location of the schema. This value is used in the import and include elements. |
|               | targetNamespace: | The URI that uniquely identifies this schema's namespace. |
|               | targetNamespacePrefix: | The prefix that abbreviates the targetNamespace. |

© 1998-2008 Sparx Systems Pty Ltd
### «XSDcomplexType»

<table>
<thead>
<tr>
<th><strong>UML Construct</strong></th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>complexType definitions are created for generic UML Classes. This stereotypes helps tailor the generation of a complexType definition.</td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
<td>MemberNames: (qualified</td>
</tr>
<tr>
<td></td>
<td>Mixed: (true</td>
</tr>
<tr>
<td></td>
<td>ModelGroup: (all</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>

### «XSDsimpleType»

<table>
<thead>
<tr>
<th><strong>UML Construct</strong></th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>An XSD simpleType is generated for Classes with this stereotype.</td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
<td>Derivation: (restriction</td>
</tr>
<tr>
<td></td>
<td>Length:</td>
</tr>
<tr>
<td></td>
<td>MinLength:</td>
</tr>
<tr>
<td></td>
<td>MaxLength:</td>
</tr>
<tr>
<td></td>
<td>MinInclusive:</td>
</tr>
<tr>
<td></td>
<td>MinExclusive:</td>
</tr>
<tr>
<td></td>
<td>MaxInclusive:</td>
</tr>
<tr>
<td></td>
<td>MaxExclusive:</td>
</tr>
<tr>
<td></td>
<td>TotalDigits:</td>
</tr>
<tr>
<td></td>
<td>FractionDigits:</td>
</tr>
<tr>
<td></td>
<td>WhiteSpace:</td>
</tr>
<tr>
<td></td>
<td>Pattern:</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>This Class can only participate in an inheritance relation with another simpleType. It cannot have any attributes or own any associations; they are ignored if present.</td>
</tr>
</tbody>
</table>

### «XSDsequence»

<table>
<thead>
<tr>
<th><strong>UML Construct</strong></th>
<th>Class</th>
</tr>
</thead>
</table>

Enterprise Architect User Guide
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.

**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 are the children of this model group when added to that complexType.

<table>
<thead>
<tr>
<th>Tagged Values</th>
<th>None.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>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.</td>
</tr>
</tbody>
</table>

### «XSDchoice»

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Creates an XSD choice element. See XSDsequence for more details.</td>
</tr>
<tr>
<td>Tagged Values</td>
<td>None.</td>
</tr>
<tr>
<td>Constraints</td>
<td>As for XSDsequence.</td>
</tr>
</tbody>
</table>

### «XSDelement»

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Attribute: AssociationEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

| Tagged Values | form: (qualified | unqualified) Override 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. |
|               | default fixed See the W3C XML Schema recommendation. |
| Constraints   | None. |
### «XSDAttribute»

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Attribute: AssociationEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
<tr>
<td>Tagged Values</td>
<td>form: (qualified</td>
</tr>
<tr>
<td></td>
<td>Overrides the schema’s attributeFormDefault value.</td>
</tr>
<tr>
<td></td>
<td>use: (prohibited</td>
</tr>
<tr>
<td></td>
<td>See the W3C XML Schema recommendation.</td>
</tr>
<tr>
<td></td>
<td>default</td>
</tr>
<tr>
<td></td>
<td>fixed</td>
</tr>
<tr>
<td>Constraints</td>
<td>The attribute datatype should not see a Class specification, otherwise it is ignored.</td>
</tr>
</tbody>
</table>

### «XSDany»

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Class: Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>If applied to a UML attribute, an XSD anyAttribute element is generated. If applied to a UML Class, an XSD any element is generated.</td>
</tr>
<tr>
<td>Tagged Values</td>
<td>namespace:</td>
</tr>
<tr>
<td></td>
<td>See the W3C XML Schema recommendation.</td>
</tr>
<tr>
<td></td>
<td>processContents: (skip</td>
</tr>
<tr>
<td>Constraints</td>
<td>None.</td>
</tr>
</tbody>
</table>

### «XSDrestriction»

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Overrides the default use of XSD extension for inheritance and generates the child as a complexType with a restriction element instead.</td>
</tr>
<tr>
<td>Tagged Values</td>
<td>None.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Applies only to UML Class parent-child relations.</td>
</tr>
</tbody>
</table>

### «XSDgroup»

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>An XSDgroup is generated for Classes with this stereotype.</td>
</tr>
<tr>
<td>Tagged Values</td>
<td>modelGroup: (sequence</td>
</tr>
<tr>
<td></td>
<td>Overrides the default XSD model for generating this group definition.</td>
</tr>
</tbody>
</table>
### 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 connector.
A group Class cannot be inherited/aggregated.

<table>
<thead>
<tr>
<th>«XSDtopLevelElement»</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UML Construct</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>«XSDtopLevelAttribute»</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UML Construct</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>«XSDunion»</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UML Construct</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
</tr>
</tbody>
</table>
XML Technologies | XML Schema (XSD)

11.1.1.2 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.

Sparx Systems 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 is represented by a UML Class in a package named XSDDatatypes. To import the XSDDatatypes package into your model, follow the steps below:

1. Download the XSDDatatypes package using the following link: XSDDatatypes Package. The file XSDDatatypes.xml is an XMI file.
2. Use Enterprise Architect’s XMI import facility, which is available via the Project | Import/Export | Import Package from XMI menu option.
3. When the XMI import is complete, you have the UML package named XSDDatatypes in your model, from which you can drag and drop the relevant types as required.

11.1.1.3 Abstract XSD models

XML schemas can be modeled using simple, abstract Class models. This can be useful in enabling an architect to start work at a higher level of abstraction, without concern for the implementation details of a schema. Such an abstract model can be refined further using the XML Schema pages of the Enterprise Architect UML Toolbox, or it can be generated directly by Enterprise Architect’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.

The following is a simplified version of the Employee Details example model, which does not use XSD-specific stereotypes or Tagged Values.

<table>
<thead>
<tr>
<th>«XSDattributeGroup»</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UML Construct</strong></td>
<td>Class</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Creates an <code>&lt;XSDattributeGroup&gt;</code> construct which can act as a container for a set of elements for stereotype XSDattribute.</td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>An XSDattributeGroup Class can contain only elements of stereotype XSDattribute and can be associated only with other XSDattributeGroup Classes. Only XSDcomplexType Classes can associate with this Class. This Class cannot be inherited.</td>
</tr>
</tbody>
</table>
The following schema fragment would be generated by Enterprise Architect, given the above model.

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <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="Person" type="Person"/>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="surName" type="xs:string"/>
      <xs:element name="birthDate" type="xs:string"/>
      <xs:element name="gender" type="xs:string"/>
      <xs:element name="contactDetails" type="ContactInfo"/>
    </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="status" type="Status"/>
          <xs:element name="jobTitle" type="xs:string"/>
          <xs:element name="startDate" type="xs:dateTime"/>
          <xs:element name="department" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="EmployeeRecords" type="EmployeeRecords"/>
  <xs:complexType name="EmployeeRecords">
    <xs:sequence>
      <xs:element name="Employee" type="Employee" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ContactInfo" type="ContactInfo"/>
  <xs:complexType name="ContactInfo">
    <xs:sequence>
      <xs:element name="homePhone" type="xs:string"/>
      <xs:element name="mobilePhone" type="xs:string"/>
      <xs:element name="officePhone" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="streetAddress" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
11.1.1.3.1 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 schema generator when generating unsterotyped elements in an abstract model. The XML Schema pages of the Enterprise Architect UML Toolbox (and UML Profile for XSD) override these default mappings through the use of stereotypes and Tagged Values.

<table>
<thead>
<tr>
<th>UML Construct</th>
<th>Default XSD Production Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>A schema element is generated for the target package. If the target package includes</td>
</tr>
<tr>
<td></td>
<td>Classes from another package, which has the Tagged Values targetNamespace and</td>
</tr>
<tr>
<td></td>
<td>targetNamespacePrefix set, these are included as attributes of the schema element.</td>
</tr>
<tr>
<td></td>
<td>In addition, an import or include element is created for each referenced package. (An</td>
</tr>
<tr>
<td></td>
<td>include element is used if the external package shares the same targetNamespace Tagged</td>
</tr>
<tr>
<td></td>
<td>Value as the target package. An import element is used where the targetNamespaces differ).</td>
</tr>
<tr>
<td>Class</td>
<td>A root-level element declaration and complexType definition are generated. The element</td>
</tr>
<tr>
<td></td>
<td>name and type are the same as the Class name. An XSD sequence model group is generated to</td>
</tr>
<tr>
<td></td>
<td>contain UML attributes generated as elements.</td>
</tr>
<tr>
<td>Attribute</td>
<td>An element is declared for each Class attribute. The element name is set to that of the</td>
</tr>
<tr>
<td></td>
<td>UML attribute name. This is prefixed with the Class name to make the element unique. The</td>
</tr>
<tr>
<td></td>
<td>minOccurs and maxOccurs attributes are set to reflect the attribute cardinality.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>If left unspecified, minOccurs and maxOccurs default to 1.</td>
</tr>
<tr>
<td></td>
<td>If the attribute refers to another Class, the element declaration is followed a</td>
</tr>
<tr>
<td></td>
<td>complexType definition, which contains a reference to the appropriate complexType.</td>
</tr>
<tr>
<td>Association</td>
<td>An element is declared for each association owned by a Class. The element name is set to</td>
</tr>
<tr>
<td></td>
<td>that of the association role. The minOccurs and maxOccurs reflect the cardinality of the</td>
</tr>
<tr>
<td></td>
<td>association.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>If the direction of the association is unspecified, the owner is assumed to be the source.</td>
</tr>
<tr>
<td>Generalization</td>
<td>For single inheritances, an extension element is generated with the base attribute set to</td>
</tr>
<tr>
<td>(Inheritance)</td>
<td>the base Classname. The UML attributes of the child Class are then appended to an all</td>
</tr>
<tr>
<td></td>
<td>model group within the extension element.</td>
</tr>
<tr>
<td>«enumeration»</td>
<td>A simpleType element is declared for the enumeration Class with the name attribute set to</td>
</tr>
<tr>
<td>(stereotype)</td>
<td>the Classname. A restriction element is generated with base set to string. Each of the</td>
</tr>
<tr>
<td></td>
<td>Class attributes is appended to the restriction element as XSD enumeration elements with</td>
</tr>
<tr>
<td></td>
<td>value set to the UML attribute name. Any type specification for the UML attributes is</td>
</tr>
<tr>
<td></td>
<td>ignored by the schema generator.</td>
</tr>
</tbody>
</table>
11.1.2 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 Enterprise Architect, therefore XML schema generation is a package-level operation. To generate an XML schema from a package, follow the steps below:

1. In the Project Browser, right-click on the package to be converted to XSD. The context menu displays.
2. Select the Code Engineering | Generate XML Schema menu option. The Generate XML Schema dialog displays, showing the name of the selected package in the Source Package field.

3. In the Encoding field, set the required XML encoding.
4. In the XSD Style panel, the Generate global element for all global ComplexTypes checkbox is selected by default to generate schema in the Garden of Eden style.[1021]
5. In the Referenced Package Options panel, select the:
   • Generate XSD for Referenced packages checkbox to generate schema for packages that are referenced by any of the packages selected in the list box
   • Prompt when missing Filename checkbox to enable Enterprise Architect to prompt for a filename

© 1998-2008 Sparx Systems Pty Ltd
for a referenced package during schema generation, if the filename is missing.

6. In the Child Package Options panel, select the:
   - **Generate XSD for Child Packages** checkbox to generate schema for child packages of the selected package
   - **Include all packages** radio button to list all child packages under the parent package in the list box
   - **Include <XSDschema> packages** radio button to list only those packages that have the stereotype «XSDschema».

   The list box displays, for each package, the package name and the file path where the schema file is to be generated.

7. If it is necessary to change the file path for a package, double-click on the entry in the list box and, on the Select XML File dialog, type or select the appropriate file path.

8. Ensure that the checkbox is selected for each package required for generation.

9. Click on the **Generate** button to generate the schema for each of the selected packages.

10. The progress of the schema generator is shown in the **Progress** box.

11. When schema generation is complete, click on an entry in the list box and click on the **View Schema** button to review the generated schema.

**Tip:**

The **Generate XML Schema** dialog can also be accessed from the active diagram by selecting the **Project | XML Schema | Generate XML Schema** menu option.

### 11.1.2.1 Generate Global Element

Enterprise Architect, by default, generates XML Schema in the Garden of Eden style. For every global XSDcomplexType stereotyped Class, Enterprise Architect generates a global element. For example, the following model by default generates the XSD shown:

![Diagram of a global XSDcomplexType]

```xml
<xs:element name="Address" type="Address"/>
<xs:complexType name="Address">
  <xs:sequence/>
</xs:complexType>

<xs:element name="PersonName" type="PersonName"/>
<xs:complexType name="PersonName">
  <xs:sequence>
    <xs:element name="firstName" type="string"/>
    <xs:element name="lastName" type="string"/>
  </xs:sequence>
</xs:complexType>
```

You can change this default behaviour by deselecting the **Generate global element for all global ComplexTypes** checkbox on the **Generate XSD** dialog. Then, the generated XSD no longer contains the global element, as shown below:

![Diagram of a global XSDcomplexType with global elements deselected]

```xml
<xs:element name="Address" type="Address"/>
<xs:complexType name="Address">
  <xs:sequence/>
</xs:complexType>

<xs:element name="PersonName" type="PersonName"/>
<xs:complexType name="PersonName">
  <xs:sequence/>
</xs:complexType>
```
<xs:complexType name="Address">
    <xs:sequence/>
</xs:complexType>

<xs:complexType name="PersonName">
    <xs:sequence>
        <xs:element name="firstName">
            <xs:complexType>
                <xs:sequence/>
            </xs:complexType>
        </xs:element>
        <xs:element name="lastName">
            <xs:complexType>
                <xs:sequence/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
11.1.3 Import XSD

The XML Schema Import facility is used to reverse engineer a W3C XML Schema (XSD) file as a UML Class model. An XSD file is imported into Enterprise Architect as a UML package. To import an XSD file, follow the steps below:

1. In the Project Browser, right-click on the package to contain the imported XSD package. The context menu displays.

3. In the Directory field, click on the [ ... ] (Browse) button. The Select XML Schema(s) dialog displays.
4. Click on the required input file. To select several individual files, press [Ctrl] as you click on each one. To select a range of files, press [Shift] and click on the first and last file in the range.
5. Click on the Open button to return to the Import XML Schema dialog, which now shows the selected files in the Selected File(s) field.
6. The Import global elements with "Type" postfix checkbox defaults to unselected to import a global element, and the ComplexType to which it refers, as a single ComplexType Class.
7. The Import referenced XML Schema(s) checkbox defaults to selected, to import any other Schema file referenced by the selected input XML Schema file or files.
Note:
If an XML Schema file being imported already exists in the model, Enterprise Architect skips importing the file.

8. The **Create Diagram for XML Schema(s)** checkbox defaults to selected, to display the imported elements on the diagram. If necessary, deselect the checkbox.

9. For the **Import XSD Elements/Attributes as:** field, select the appropriate radio button to import elements and attributes in the XML Schema as:
   - UML Association connectors or
   - UML Class attributes.

10. Click on the **Import** button to import the schema.

11. The progress of the schema import is shown in the **Progress** status bar.

Tip:
The **Import XML Schema** dialog can also be accessed for the active diagram by selecting the **Project | XML Schema | Import XML Schema** menu option.

Note:
Enterprise Architect uses the **schemaLocation** attribute in the Import and Include elements of an XML Schema to determine the dependencies between the files. Ensure that this attribute is set to a valid file path (and not a URL) for the dependent XML Schema(s) to be imported correctly.

### 11.1.3.1 Global Element and ComplexType

Some XML Schemas have **ComplexType** elements with the same name as the referring global elements, but with the suffix **Type** as shown below:

```xml
<xs:element name="Address" type="AddressType"/>
<xs:complexType name="AddressType">
  <xs:sequence/>
</xs:complexType>
```

On XSD import, Enterprise Architect treats this global element and its bounding ComplexType as a single entity and creates a single **XSDcomplexType** stereotyped Class with the same name as the global element as shown below:

![XSDcomplexType](image)

You can change this default behaviour by selecting the **Import global elements with "Type" postfix** checkbox. When you select this option, Enterprise Architect treats the global element and the ComplexType it is referring to as two separate entities. So, for the above example, Enterprise Architect creates an **XSDtopLevelElement** stereotyped Class for the global element and an **XSDcomplexType** stereotyped Class for the ComplexType, and connects them as follows:

![Diagram](image)
Note:

Enterprise Architect treats the following as two separate entities irrespective of whether the Import global elements with "Type" postfix checkbox is selected or unselected:

```xml
<xs:element name="HomeAddress" type="AddressType"/>
<xs:complexType name="AddressType">
  <xs:sequence/>
</xs:complexType>
```
11.2 Web Services (WSDL)

Enterprise Architect supports Forward and Reverse Engineering of the W3C Web Service Definition Language (WSDL). The following topics explain how to use Enterprise Architect to model, generate and import WSDL files:

- Model WSDL
- Import WSDL
- Generate WSDL
11.2.1 Model WSDL

The WSDL pages of the Enterprise Architect UML 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 can contain one or more WSDL components. Each WSDL component can be automatically generated to a WSDL file using Enterprise Architect’s built in WSDL generator. The following topics describe the various WSDL elements and features supported by Enterprise Architect:

- **WSDL Namespace**
- **WSDL Document**
- **WSDL Service**
- **WSDL Port Type**
- **WSDL Message**
- **WSDL Binding**
- **WSDL Port Type Operation**
- **WSDL Message Part**

11.2.1.1 WSDL Namespace

The WSDL namespace in Enterprise Architect 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 can reuse its schema Types, Messages, Port Types, Bindings and Service across multiple physical WSDL documents.

The figure below shows an example WSDL namespace (`WSDDPackage1`, which has a red margin to the bottom right corner), including a single WSDL document:
To create a new WSDL namespace in your model, follow the steps below.

1. Open or create the appropriate diagram.
2. Select the **More Tools | Extended | WSDL** menu option from the Enterprise Architect UML Toolbox.
3. Drag the **Namespace** element from the Toolbox onto the diagram. The WSDL Namespace Properties dialog displays:

   - **WSDL Package Name**: Enter a name for the package.
   - **Target Namespace**: Enter the target namespace URI.

4. Click on the **OK** button to create a package stereotyped as **WSDLnamespace**. This contains 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** and you drag XSD elements onto the Types diagram from the XSD Schema page of the Enterprise Architect UML Toolbox.
   - **Messages**: Contains the WSDL Messages, modeled as UML Classes marked with the stereotype **WSDLmessage**.
   - **PortTypes**: Contains the WSDL Port Types, modeled as UML interfaces marked with the stereotype **WSDLportType**.
   - **Bindings**: Contains the WSDL Bindings, modeled as UML Classes that realize the PortTypes.
   - **Services**: Contains the WSDL Services, modeled as UML interfaces with associations to each exposed Binding.

5. Use the Overview diagram to navigate between the sub-packages, by double-clicking the relevant packages. You can edit the sample WSDL elements created in the previous step, or drag new items from the WSDL pages of the Toolbox onto the relevant diagrams.
You can edit the WSDL-specific properties of the namespace later by double-clicking the package in the Project Browser. Alternatively, on the WSDL Namespace Properties dialog, click on the UML button to invoke the standard Properties dialog for a package. (This button does not display on the initial WSDL Namespace Properties dialog for a new Namespace element.)

### 11.2.1.2 WSDL Document

WSDL documents are represented in Enterprise Architect by UML components stereotyped as «WSDL». These components are modeled as direct child elements of the top-level WSDL namespace package. You can create multiple WSDL documents for a single namespace, thus enabling the services for that namespace to be reused and exposed as required across multiple WSDLs.

To define new WSDL document components for your namespace, follow the steps below:

1. Open the Overview diagram defined for your WSDL namespace package, and drag the WSDL element from the Toolbox onto the diagram. The WSDL Document Properties dialog displays.
2. Type in the **Name** and **File Name** for the document.

3. The **XMLNS panel** lists the default XML namespaces used by the document. If required, click on the **New** button to add further namespaces.

   **Note:**
   You can also delete any namespace entries that you add. It is recommended that you do not delete any of the default entries, as it may cause an invalid WSDL document to be generated.

4. Select one or more services that should be exposed by this document. The list of available services is populated from the **Services package**.

5. Click on the **OK** button.

You can edit the WSDL-specific properties of the document later by double-clicking the component in the diagram or the **Project Browser**. Alternatively, click on the **UML** button in the WSDL Document Properties dialog to invoke the standard Properties dialog for a package. (This button does not display on the initial WSDL Document Properties dialog for a new WSDL element.)

### 11.2.1.3 WSDL Service

WSDL services are represented in Enterprise Architect by UML interfaces, stereotyped as **WSDL service**. Services should be defined under the Services packages in the WSDL namespace structure.

To define new **WSDL service** elements for your namespace, follow the steps below:

1. Open the Overview diagram defined for your WSDL namespace package, and double-click on the **Services package** element to open the Services diagram.

2. Drag the **Service** element from the **Toolbox** onto the diagram. The **WSDL Service** dialog displays.
3. In the **Name** field, type the service name.
4. Click on the **New** button to add Service Ports. The WSDL Port dialog displays.

5. Type in the **Port Name** and **Location**, and select a **Binding**. The list of Bindings is taken from those defined in the Binding package.
6. Click on the **OK** button to close the WSDL Port dialog. For each Port defined in this way, Enterprise Architect creates an Association relationship between the Service and corresponding Binding element.
7. Click on the **OK** button to close the WSDL Service dialog.

You can edit the WSDL-specific properties of the service later by double-clicking the Service interface in the diagram or Project Browser. Alternatively, click on the **UML** button in the WSDL Service dialog to invoke the standard Properties dialog for an interface. (This button does not display on the initial WSDL Service dialog for a new Service element.)

### 11.2.1.4 WSDL Port Type

WSDL Port Types are represented in Enterprise Architect by UML interfaces stereotyped as **WSDLportType**. PortTypes should be defined under the **PortTypes** packages in the WSDL namespace structure.

To define new WSDLportType elements for your namespace, follow the steps below:

1. Open the Overview diagram defined for your WSDL namespace package, and double-click on the PortTypes package to open the PortTypes diagram.
2. Drag the **Port Type** element from the Toolbox onto the diagram. The WSDL PortType dialog displays.
3. Type in the name for the portType.
4. Click on the OK button to close the WSDL PortType dialog.
5. Define operations for the portType by dragging the Port Type Operation item from the WSDL page of the Enterprise Architect UML Toolbox onto the portType interface.

You can edit the WSDL-specific properties of the portType later by double-clicking the interface in the diagram or Project Browser. Alternatively, in the WSDL PortType dialog, click on the UML button to invoke the standard Properties dialog for an interface. (This button does not display on the initial WSDL PortType dialog for a new PortType element.)

11.2.1.5 WSDL Message

WSDL messages are represented in Enterprise Architect by UML Classes stereotyped as WSDLmessage. Messages should be defined under the Messages package in the WSDL namespace structure.

To define new WSDLMessage elements for your namespace, follow the steps below:
1. Open the Overview diagram defined for your WSDL namespace package, and double-click on the Messages package to open the Messages diagram.
2. Drag the Message element from the Toolbox onto the diagram. The WSDL Message dialog displays.

3. Type in the Name for the message.
4. Click on the OK button to close the WSDL Message dialog.
5. You can define parts for the message by dragging the Message Part element from the WSDL Elements page of the Enterprise Architect UML Toolbox onto the Message element.

You can edit the WSDL-specific properties of the message later by double-clicking the Message element in the diagram or Project Browser. Alternatively, on the WSDL Message dialog, click on the UML button to invoke the standard Properties dialog for a Class. (This button does not display on the initial WSDL Message dialog for a new Message element.)

11.2.1.6 WSDL Binding

WSDL bindings are represented in Enterprise Architect by UML Classes stereotyped as WSDLbinding. Bindings should be defined under the Bindings package in the WSDL namespace structure. Each WSDLbinding Class implements the operations specified by a particular WSDLportType interface. Therefore, WSDLportTypes should be defined before creating WSDLbindings.

To define new WSDLbinding elements for your namespace, follow the steps below:
1. Open the Overview diagram defined for your WSDL namespace package, and double-click on the
2. Drag the Binding element from the Toolbox onto the diagram. The WSDL Binding dialog displays.

3. Type in a Name for the Binding.
4. Select the PortType for the Binding; the drop-down 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. Click on the OK button to close the WSDL Binding dialog and create the binding. A realization connector is created between the binding and the corresponding Port Type interface.
8. To specify the Binding operations, select and double-click on an Operation in the Binding element. The WSDL Binding Operation Details dialog displays.
9. Type in or select the Binding Operation details.
10. Click on the Parameters button. The WSDL Binding Operation Parameters dialog displays. For each input, output and fault, click on the Details button and enter the details.
11. Click on the OK button on each of the WSDL Binding Parameter Details, WSDL Binding Operation Parameters and WSDL Binding Operation Details dialogs to close them.

You can edit the WSDL-specific properties of the binding later by double-clicking the binding Class in the
11.2.1.7 **WSDL Port Type Operation**

WSDL portType operations are represented in Enterprise Architect by operations defined as part of a WSDL portType interface (see the [WSDL Port Type](#) topic).

To add portType operations to your WSDL portType interfaces, follow the steps below.

1. Open the Overview diagram defined for your WSDL namespace package, and double-click on the PortTypes package to open the PortTypes diagram.
2. Drag the **PortType Operation** item onto a WSDL portType stereotyped interface. The WSDL PortType Operation dialog displays.

3. Type in the **Name** for the operation.
4. Select the **Operation Type**.
5. Type in or select the **Input**, **Output** and **Fault** details for the operation. The **Message** drop-down list is taken from the WSDL message elements defined under the Messages package.
6. Click on the **OK** button to close the WSDL PortType Operation dialog and create the operation.

You can edit the WSDL-specific properties of the portType operation later by double-clicking the operation in the diagram or **Project Browser**. Alternatively, on the WSDL PortType Operation dialog, click on the **UML**
button to invoke the standard Properties dialog for an operation. (This button does not display on the initial WSDL PortType Operation dialog for a new PortType Operation.)

11.2.1.8 WSDL Message Part

WSDL message parts are represented in Enterprise Architect by UML attributes defined as part of a WSDL message Class (see the WSDL Message topic). To add message parts to your WSDL message Classes, follow the steps below:

1. Open the Overview diagram defined for your WSDL namespace package, and double-click on the Messages package to open the Messages diagram.
2. Drag the Message Part element onto a WSDL message stereotyped Class. The WSDL Message Part dialog displays.

   Name: element
   Type: string

   OK  Cancel  Help

3. Type in a Name and Type for the message part. The type should be selected from the drop-down list of primitive XSD types or from the types defined under the Types package.
4. Click on the OK button.

You can edit the WSDL-specific properties of the message part later by double-clicking the attribute in the diagram or Project Browser. Alternatively, on the WSDL Message Part dialog, click on the UML button to invoke the standard Properties dialog for an attribute. (This button does not display on the initial WSDL Message Part dialog for a new Message part attribute.)
11.2.2 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. To generate one or more WSDL files from a WSDLnamespace, follow the steps below:

1. In the Project Browser, right-click on the target `WSDLnamespace` package to display the context menu.
3. For each WSDL component, set the required output file using the Target File column.
4. Using the Encoding field, set the required XML encoding.
5. Click on the Generate button to generate the WSDL files.
6. The progress of the WSDL generator is shown in the Progress edit box.

**Tip:**

The Generate WSDL dialog can also be accessed from the active diagram by selecting the Project | Generate WSDL menu option.
11.2.3 Import WSDL

The WSDL Import facility is used to reverse engineer WSDL files as UML Class models. To import a WSDL file, follow the steps below:

1. In the Project Browser, right-click on the package to contain the imported WSDL package. The context menu displays.
2. Select the Code Engineering | Import WSDL menu option. The Import WSDL dialog displays.
3. In the Filename field, select the input file.
4. The Target Package field is automatically set to the name of the selected input file. If required, change this name.
5. Click on the Import button to import the schema.
6. The progress of the WSDL import is shown in the Progress status bar.
Part XII
12 Data Modeling

You perform database modeling and database design 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 enable you to model database keys, triggers, constraints, RI and other relational database features.

Typical data modeling tasks you might perform are listed at the end of this topic.

Tables and Columns

The basic modeling structure of a relational database is the table, which represents a set of records, or rows, with the same structure. The basic organizational element of a relational database is the column. Every individual item of data entered into a relational database is represented by a value in a column of a row in a table.

The UML Data Modeling Profile represents:

- Tables as stereotyped Classes; that is, Class elements with a stereotype of table.
- Columns as stereotyped attributes; that is, attributes with a stereotype of column.

Enterprise Architect can generate simple DDL scripts to create the tables in your model.

Database Keys

Two types of key are used to access tables: 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.

A Primary Key consists of one or more columns; a simple Primary Key (single column) is defined as the attribute of a stereotyped operation. A complex Primary Key (several columns) is defined as the stereotyped operation itself.

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). Foreign keys are represented in Enterprise Architect as operations with the stereotype FK; the operation parameters become the columns involved in the key.

Supported Databases

Enterprise Architect supports import of database schema from these databases:

- DB2
- Firebird/InterBase
- Informix
- Ingres
- MS Access
- MS SQL Server
- MySQL
- Oracle 9i, 10g and 11g
- PostgreSQL
- Sybase Adaptive Server Anywhere (Sybase ASA)
- Sybase Adaptive Server Enterprise (Sybase ASE).
**Note:**

Firebird 1.5 database tables can be modeled and generated as InterBase tables. Firebird tables can be imported but are treated as InterBase tables.

---

**Typical Tasks**

Typical tasks you can perform when modeling or designing databases include:

- Create a Data Model Diagram
- Create a Table
- Set Properties of a Table
- Create Columns
- Create Oracle Packages
- Create Primary Keys
- Create Foreign Keys
- Create Stored Procedures
- Create Views
- Create Indexes and Triggers
- Generate DDL for a Table and compare with the database
- Convert Datatypes for a Table
- Convert Datatypes for a Package
- Customize Datatypes for a DBMS
- Import 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.
12.1 A Data Model Diagram

An example of a Data Model diagram is provided below, showing three tables that are linked on primary to foreign key pairs with associated multiplicity.

Note 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.
12.2 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 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.
3. Click on the Table element in the list of elements, then click on the diagram. The Table element is displayed on the diagram.
4. If the Class: Table n Properties dialog does not display, double-click on the Table to display it.
5. In the Name field, type a name for the Table and set any other properties as required.
6. Click on the OK button.
12.3 Set Table Properties

Once you have created your table, you can set its properties. Most table properties can be set from the Properties dialog, as described below. However, some properties must be entered as Tagged Values as 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 are available for defining columns, and also declares which dialect of DDL is generated. Enterprise Architect supports the following databases:

- DB2
- Informix
- Ingres
- InterBase
- MS Access
- MySQL
- Oracle 9i, 10g and 11g
- PostgreSQL
- SQL Server 2000 and 2005
- SQL Server 7
- Sybase Adaptive Server Anywhere (Sybase ASA)
- Sybase Adaptive Server Enterprise (Sybase ASE).

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. In the **Database** field, click on the drop-down arrow and select the database type.
4. Click on the **OK** button to save changes.

By clicking on the **Table Detail** tab on this dialog, you can access the **Columns dialog** or **Operations dialog**, or you can **Generate DDL** for this table.

**Note:**

Stored Procedure elements and Object instances of Views have very similar **Detail** tabs that enable you to display the definition and initial code of these data structures.
12.3.1 Set Table Owner

To define the owner of a table, follow the steps below:

1. Select the View | Tagged Values menu option or press [Ctrl]+[Shift]+[6]. The Tagged Values window displays.
2. Click on the table in a diagram or the Project Browser. The Tagged Values window now shows the tags for the selected table.
3. Click on the New Tag button. The Tagged Value dialog displays.
4. In the Tag field, type the tag name Owner. In the Value field, type a value for the Owner tag.
5. Click on the OK button to confirm the operation. Generated DDL includes the table owner in the SQL script.
12.3.2 Set MySQL Options

In MySQL, to make use of foreign keys you must declare the table type as *InnoDB*. To do this, follow the steps below:

1. Select the View | Tagged Values menu option or press [Ctrl]+[Shift]+[6]. The Tagged Values window displays.
2. Click on the table in a diagram or in the Project Browser. The Tagged Values window shows the table as selected.
3. Click on the New Tag button. The Tagged Value dialog displays.
4. In the Tag field, enter the tag name Type. In the Value field, type *InnoDB* as the value for the Type tag.
5. Click on the OK button to confirm the operation. Generated DDL includes the table type in the SQL script.
6. To allow for later versions of MySQL, additional table options that can be added in the same manner include:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value (Example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGINE</td>
<td>InnoDB</td>
</tr>
<tr>
<td>CHARACTER SET</td>
<td>latin1</td>
</tr>
<tr>
<td>CHARSET</td>
<td>latin1</td>
</tr>
<tr>
<td>COLLATE</td>
<td>latin1_german2_ci</td>
</tr>
</tbody>
</table>
12.3.3 Set Oracle Table Properties

For Oracle, you can set table properties using the table's Tagged Values. Follow the steps below:

1. Select the View | Tagged Values menu option, or press [Ctrl]-[Shift]-[6]. The Tagged Values window displays.
2. Click on the table in a diagram or in the Project Browser. The Tagged Values window displays the table name, selected.
3. Click on the (New Tag) button.
4. Define the table properties as shown in the examples below:

5. Click on the OK button to save the Tagged Value.

All available properties for an Oracle table are listed below.

**Note:**
The same properties can be added to indexes and constraints. Highlight the index or constraint and add the properties as Tagged Values.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER_POOL</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>CACHE</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>DBVERSION</td>
<td>9.0.111</td>
</tr>
<tr>
<td>FREELISTS</td>
<td>1</td>
</tr>
<tr>
<td>GRANT OWNER1</td>
<td>SELECT</td>
</tr>
<tr>
<td>GRANT OWNER2</td>
<td>DELETE, INSERT, SELECT, UPDATE</td>
</tr>
<tr>
<td>INITIAL</td>
<td>65536</td>
</tr>
<tr>
<td>INITTRANS</td>
<td>1</td>
</tr>
<tr>
<td>LOGGING</td>
<td>LOGGING</td>
</tr>
</tbody>
</table>
The properties defined for a given table are listed on the Tagged Values window, as illustrated by the following typical Tagged Value list:
12.4 Create Columns

What is a Column?
The basic organizational element of a relational database is the column. Every individual item of data entered into a relational database is represented as a value 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

Note:
For MySQL, before creating columns first add ENUM and SET datatypes. Select the Settings | Database Datatypes menu option and, on the Database Datatypes dialog, in the Product Name field select MySQL. Add the datatypes ENUM and SET.

To create columns, follow the steps below:

1. Right-click on the Table in a diagram to open the context menu, and select the Attributes menu option.
2. The <Tablename> Columns dialog displays.

<table>
<thead>
<tr>
<th>PK</th>
<th>Name</th>
<th>Type</th>
<th>Not ...</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Support</td>
<td>VARCHAR</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>StaffName</td>
<td>VARCHAR</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PK</td>
<td>StaffIdentifier</td>
<td>VARCHAR</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
3. Type in the column **Name** and **Data Type** and click on the **Save** button.

   **Tip:**
   If the drop-down list of datatypes is empty, this means that you have not selected a target database for the table. Close the Columns dialog and re-open the Table Properties dialog to set a Database type before continuing. To prevent this recurring, set the default database type.

4. The following fields for each column are optional:
   - **Primary Key** - select the checkbox if the column represents the primary key for this table
   - **Not Null** - select the checkbox if empty values are forbidden for this column
   - **Unique** - select the checkbox if it is forbidden for any two values of this column to be identical
   - **Initial** - type a value that can be used as a default value for this column, if required
   - **Access** - click on the drop-down arrow and select a scope of **Private**, **Protected** or **Public** (the field defaults to **Public**)
   - **Alias** - type an alternative name for the field (for display purposes), if any
   - **Notes** - type any other information necessary to document the column; you can format the text using the Rich Text Notes toolbar at the top of the field.

**Notes:**
- Some datatypes, such as the Oracle NUMBER type, require a precision and scale. These fields are displayed where required and should be filled in as appropriate. For example, for Oracle:
  - create NUMBER by setting Precision = 0 and Scale = 0
  - create NUMBER(8) by setting Precision = 8 and Scale = 0
  - create NUMBER(8,2) by setting Precision = 8 and Scale = 2.
- Oracle VARCHAR2(15 CHAR) and VARCHAR2(50 BYTE) datatypes can be created by adding the tag `LengthType` with the value `CHAR` or `BYTE`.
- For MySQL ENUM and SET datatypes, in the **Initial** field type the values as a comma-separated list, in the format `(‘one’,‘two’,‘three’)` or, if one value is the default, in the format: `(‘one’,‘two’,‘three’) default ‘three’`.

**Change the Column Order**
To change the column order, follow the steps below:

1. On the Columns dialog, highlight a column name in the Columns panel.
2. Click on the:
   - button to move the column up one position
   - button to move the column down one position.
12.5 Create Oracle Packages

To create an Oracle package, follow the steps below:

1. Open the project in the Project Browser and create an Enterprise Architect package (and, if required, a Class diagram).
2. Add a Class element to either the package or the diagram.
3. Open the Properties dialog for the element and, in the Stereotype field, type the value Package.
4. For the package specification, create an Operation with the name Specification and with no return type.
5. Open the Properties dialog for the Specification Operation and, on the Behavior tab, type the entire package specification into the Initial Code field.
6. For the package body, create an Operation with the name Body and with no return type.
7. Open the Properties dialog for the Body Operation and, on the Behavior tab, type the entire package body into the Initial Code field.
12.6 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.

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 display the context menu. Select the **Element Features | Attributes** menu option.
2. In the **Attributes** dialog, select the column that makes up the Primary Key.
3. Select the **Primary Key** checkbox and click on the **Save** button.

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

Often, a Primary Key consists 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.
2. Right-click on the table in a diagram to open the context menu. Select the **Element Features | Operations** menu option.
3. Select the Primary Key operation (its name begins with **PK_**) and then click on the **Columns** tab.
4. To add a column to the Primary Key, click on the **New** button, select a column from the **Column Name** list box, and then click on the **Save** button.
5. Click on the **Hand** buttons (up and down arrow) to change the order of columns in the Primary Key, if necessary.

(See also the **SQL Server Non-Clustered Primary Keys** topic).

Define a Primary Key Name Template

To define the name template for a Primary Key, follow the steps below:

1. Select the **Tools | Options | Source Code Engineering | Code Editors** menu option. The DDL page of the **Options** dialog displays.
2. Click on the **DDL Name Template** button. The **DDL Name Template** dialog displays, showing the default name templates.

   ![DDL Name Template Dialog](image)

   - **Foreign Key Name Template:** PK_%forighttablename%_%primarytablename%
   - **Primary Key Name Template:** PK_%tablename%

3. Edit or replace the template in the **Primary Key Name Template** field.

   **Note:**
   If you want to display the Primary Key description as `PK_tablename_columnname` then change the **Primary Key Name Template** field to `PK_%tablename_%columnname%`.

4. Click on the **Save** button.
12.6.1 SQL Server Non Clustered 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. Select the **Element Features | Operations** menu option. The Table Operations dialog displays.
3. Highlight the Primary Key Operation and select **Extended Properties**.
4. Select the **SQL Server Non Clustered Primary Key** checkbox.
5. Click on the **Save & Close** button.
12.7 Foreign Key

What is a Foreign Key?

Two types of key are used to access tables: 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 that 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.

To create a Foreign Key, click on the link.

You might also have to define a Name Template for Foreign Keys.
12.7.1 Create Foreign Key

To create a Foreign Key, follow the steps below:

1. Locate the required Tables in a diagram.
2. Select an Associate connector in the Class Relationships page of the Enterprise Architect UML Toolbox.
3. Click on the Table to contain the Foreign Key (source) and draw the connector to the other Table (target).
4. Right-click on the connector to display the context menu, and select the Foreign Keys option. The Foreign Key Constraint dialog displays.

5. If necessary, edit the default name for the Foreign Key.
6. Highlight the columns involved in the Foreign Key relationship.
7. Click on the Save button to automatically generate the Foreign Key operations.

You have created the Foreign Key. The example below shows how this looks in a diagram:
Composite Foreign Key

To create a composite Foreign Key, select the appropriate columns and click on the Save button. The Foreign Key columns are sorted according to datatype to match the datatypes of the targeted composite Primary Key.

If required, you can change the order of the key columns by clicking on the and buttons.

Tip:
If you are defining a MySQL database and want to use Foreign Keys, you must set the table type to enable this.
This creates the composite Foreign Key. The example below shows how this looks in a diagram:
Table 1

- PK t1_id: int
- PK t1_name: varchar(60)
- PK t1_data_created: datetime

PKs
- PK_Table1(int, varchar, datetime)

Table 2

- FK t2_date: datetime
- FK t2_id: int
- PK t2_pk: int
- PK t2_name: varchar(60)

FKs
- FK_Table2_Table1(int varchar, datetime)
- PK_Table2(int)

Composite key columns are sorted so the foreign key datatypes match the primary key datatypes.
12.7.2 Define Foreign Key Name Template

To define the name template for a Foreign Key, follow the steps below:


2. Click on the DDL Name Template button. The DDL Name Template dialog displays, showing the default name templates.

   Foreign Key Name Template: $FK_%foreigntablename_%fkcolumnname%_%primarytablename%_%pkcolumnname%

   Primary Key Name Template: $PK_%tablename%

3. Edit or replace the name template in the Foreign key Name Template field.

   **Note:**

   If you want to display the Foreign Key description as $FK_foreigntablename_FKcolumnname_primarytablename_PKcolumnname$ then change the Foreign key Name Template field to $FK_%foreigntablename%_%fkcolumnname%_%primarytablename%_%pkcolumnname%$.

4. Click on the Save button.
12.8 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. You can compile and execute stored procedures with different parameters and results, and they can 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 Modeling. Alternatively, you can model stored procedures as individual Classes.

Note:

Stored procedures are currently supported for: DB2; SQL Server; Firebird/Interbase; Informix; Ingres; Oracle 9i, 10g and 11g; MySQL; PostgreSQL; Sybase Adaptive Server Enterprise (ASE) and Sybase Adaptive Server Anywhere (ASA).
12.8.1 Create Container Class Procedure

To create a stored procedure as an operation of a container Class, follow the steps below, in three stages:

Define Stored Procedure Stereotype

1. Select the Settings | UML menu option. The UML Types dialog displays, at the Stereotypes tab.

2. In the Stereotype field, type stored procedures.
3. In the Base Class field, type class.
4. Click on the Save button, and on the Close button.

Create Stored Procedure

1. Open the required diagram.
2. Select the More tools | UML | Class menu option in the Enterprise Architect UML Toolbox.
3. Click on the Class element in the list of elements and then click on the diagram. If the Class Properties dialog does not automatically display, double-click on the element.
4. In the Name field, type a name for the Class. Typically, this is the database name.
5. In the Stereotype field, type stored procedures.
6. Click on the OK button to close the dialog. You now have a stored procedures container.
7. Re-open the Class Properties dialog. In the Database field click on the drop-down arrow and select the target DBMS to model. (The field displays the default database if it has already been set.)
8. Select the Procedures Detail tab and click on the Stored Procedures... button.

(Alternatively:
Select the stored procedures container and press [F10], or
Select Features | Operations from the context menu.)
The <Class name> Operations dialog displays.

9. In the Name field, type the name of the stored procedure.

10. In the Return Type field click on the drop-down arrow and select the return type (or use the default value resultset).

11. In the Stereotype field, type the value proc.

12. Click on the Save button.

Add Parameters

1. Click on the procedure name in the Operations panel and click on the Edit Parameters button. The Parameters dialog displays.

2. In the Name field, type the parameter name, and in the Type field click on the drop-down arrow and select the parameter type.

   If the parameter is a length type, add the length after the parameter type. For example, select VARCHAR from the drop-down list and type (5) just after it, as the length.

   You can also type the values of the Type field directly into the field.

3. Click on the Save button, and then the Close button. The <Class name> Operation dialog redisplays.

4. Click on the Behavior tab. In the Initial Code field, type the text of the procedure.

Notes:

- If using the parameter feature as described above, you only have to add the procedure statements after the AS clause.

- If you prefer not to use the parameter feature as described above, insert the entire stored procedure text in the Initial Code field.

- In either case, the create procedure... text or create or replace procedure... text must be the first line in the Initial Code field.
5. Click on the **Save** button, and then the **Close** button.
12.8.2 Create Individual Class Procedure

To create a stored procedure as an individual Class, follow the steps below, in two stages:

Define Procedure Stereotype
1. Select the **Settings | UML** menu option. The **UML Types** dialog displays, at the **Stereotypes** tab.
2. In the **Stereotype** field, type **procedure**.
3. In the **Base Class** field, type **class**.
4. Click on the **Save** button, and on the **Close** button.

Create Procedure Element
1. Open the required diagram.
2. Select the **More tools | UML | Class** menu option in the Enterprise Architect UML Toolbox.
3. Click on the **Class** element in the list of elements and then click on the diagram. If the **Class Properties** dialog does not automatically display, double-click on the element.
4. In the **Name** field, type a name for the procedure.
5. In the **Stereotype** field, type **procedure**.
6. Click on the **OK** button to close the dialog. The new **procedure** element displays.

7. Double-click on the procedure element. The **Procedure <name>** dialog displays.
8. In the **Database** field click on the drop-down arrow and select the target DBMS to model. (The field displays the default database if it has already been set.)

9. In the **Procedure definition** field, type the entire procedure text.

10. Click on the **Save** button, and then the **Close** button.
12.9 Views

Note:

Views are currently supported for: DB2; SQL Server; Firebird/Interbase; Informix; Ingres; Oracle 9i, 10g and 11g; MySQL; PostgreSQL; Sybase Adaptive Server Enterprise (ASE) and Sybase Adaptive Server Anywhere (ASA).

Create a View Class

To create a database view, follow the steps below:

1. Select the Settings | UML menu option. The UML Types dialog displays, at the Stereotypes tab.

2. In the Stereotype field type view, and in the Base Class field type class. Click on the Save button.
3. Select a suitable diagram.
4. Open the Class pages on the Enterprise Architect UML Toolbox.
5. Click on the Class element in the list of elements and then click on the diagram.
6. In the Class Properties dialog, in the Stereotype field type view.
7. Enter a name for the view.
8. Click on the OK button to close the dialog. You now have a database view.
9. Right-click on the element and select the Properties context menu option. The View Properties dialog now displays.
10. From the **Database** drop-down list, select the target DBMS to model. The default database displays if it has already been set.

11. Click on the **Save** and **Close** buttons.

### Create a View

1. Create a Dependency connector from the view Class to the table or tables on which the view depends.
2. Open the view **Properties** dialog.
3. Enter the full view definition in the **View definition** field.
4. Click on the **Save** button to save your definition. An example is shown below:
12.10 Index, Trigger, Check Constraint

What is an Index?
An index is a sorted look-up for a table. When it is known in advance that a table must 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 required. In Enterprise Architect, 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 usually ensures 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 Enterprise Architect, a trigger is modeled as a stereotyped operation. Currently Enterprise Architect does 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. Right-click on the required table either in a diagram or in the Project Browser.
2. Select the Operations context menu option. The Operations dialog displays.
3. Add an operation (with a name such as IDX_CustomerID or TRG_OnCustomerUpdate; the IDX_ and TRG_ prefixes are optional but help identify the operation).
4. In the Stereotype field for the operation type index or trigger as appropriate (check, proc and unique are also supported).
5. Click on the Behavior tab.
6. In the Initial Code field, type the entire body of the trigger or procedure, or details of the check constraint.
7. Click on the Column tab.
8. Add the required columns in the required order then click on the Save button to save changes.

Create a Check Constraint
1. Locate the required table in either a diagram or the Project Browser.
2. Use the context menu to open the Operations dialog.
3. Add an operation (such as CHK_ColumnName).
4. In the Stereotype field for the constraint type check and click on the Save button to save changes.
5. Select the constraint operation, then the Behavior tab.
6. Enter the entire check constraint clause (e.g. col1 < 1000) in the Initial Code field and click on the Save button to save changes.

The example below shows how an index looks in a diagram (in the Order element):
12.11 Generate DDL For a Table

Note:
In the Corporate edition of Enterprise Architect, if security is enabled you must have Generate Source Code and DDL permission to generate DDL.

To generate simple DDL scripts to create the tables in your model, follow the steps below:

1. In the diagram, right-click on the table for which to generate DDL. The context menu displays.
2. Select the Generate DDL option. The Generate DDL dialog displays.

3. In the Path field, use the [...] (Browse) button to select the filename of the script to create.
4. To include comments in the DDL, in the Comment Level field select the appropriate level. For example, Column for comments on columns, or All for comments on all structures.
5. Select the checkboxes for the appropriate inclusions. For example, to include a 'drop table' command in the script, select the Create Drop SQL checkbox. Deselect the checkboxes for inclusions you do not require.

Notes:
- Some checkboxes display only if the appropriate database is defined for the table. For example, IF EXISTS displays only if the database for the table is PostgreSQL.
- If generating Oracle sequences, you must always select the Generate Triggers and Generate Sequences checkboxes; this ensures that a pre-insert trigger is generated to select the next sequence value to populate the column.

6. To create the DDL, click on the Generate button.
7. To view the output, click on the View button (you must configure a DDL viewer in the Local Settings dialog first).
Note:
You can transport these DDL scripts between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
**12.12 Generate DDL for a Package**

**Note:**
In the Corporate edition of Enterprise Architect, if security is enabled you must have Generate Source Code and DDL permission to generate DDL.

In this procedure, you can generate DDL for a package, and also compare the DDL with the database.

**Generate DDL**

To generate DDL for a package, follow the steps below:

1. Right-click on the required package in the Project Browser. The context menu displays.
Data Modeling | Generate DDL for a Package

Note:
Alternatively you can select the Project | Database Engineering | Generate Package DDL menu option.

3. Select the checkbox against each inclusion required. Deselect the checkboxes for inclusions you do not require.
**Notes:**

- Some checkboxes display only if the appropriate database is defined for the tables in the package. For example, `IF EXISTS` displays only if the database for the tables is PostgreSQL.
- If generating Oracle sequences, you must always select the Generate Triggers and Generate Sequences checkboxes; this ensures that a pre-insert trigger is generated to select the next sequence value to populate the column.

4. To recursively generate DDL, select the Include All Child Packages checkbox.
5. Click on the Generate button to proceed. Enterprise Architect prompts you for file names as the process executes.

**Compare DDL For a Database**

When you have generated the DDL, you can compare it with the database. To do this, follow the steps below:

1. On the Generate Package DDL dialog, click on the Compare button. The Compare With Database dialog displays.

   ![Compare With Database](image)

   - Database Name: 
   - Schema/Owner: 

2. Click on the ... button and locate the required database on the Select Data Source dialog.
3. For an Oracle database, if required you can also specify the Owner in the Schema/Owner field.
4. Click on the View button to perform the comparison. The Comparison Database dialog displays with the results of the comparison.
### Data Modeling | Generate DDL for a Package

#### Type | Name | Owner | Status | Suggested Action
--- | --- | --- | --- | ---
Table | Class3 | | | CREATE TABLE Class3() |
Table | Table2 | | | CREATE TABLE Table2() |
Table | Table1 | | | CREATE TABLE Table1() |
Table | StoreProcedure | | | CREATE TABLE StoreProcedure() |
Table | Class2 | | | CREATE TABLE Class2() |
Table | BookSelection | | | CREATE TABLE BookSelection() |
Table | SpinndClass | | | CREATE TABLE SpinndClass() |
Table | Class4 | | | CREATE TABLE Class4() |
Table | usynsamples | | | DROP TABLE usynsamples |
Table | uasysvalidities | | | DROP TABLE uasysvalidities |

#### Compare Current Item Columns

<table>
<thead>
<tr>
<th>Status</th>
<th>Item</th>
<th>Datatype</th>
<th>Nullable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table</td>
<td>VARCHAR</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Primary</td>
<td>VARCHAR</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DisplayName</td>
<td>VARCHAR</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RowOrder</td>
<td>INT</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TableName</td>
<td>VARCHAR</td>
<td>NOT NULL</td>
<td></td>
</tr>
</tbody>
</table>

#### Generate DDL

There are no items to show in this view.
12.13 Data Type Conversion Procedure

Once a database schema has been set up on an Enterprise Architect diagram (either by importing through ODBC or manually setting up the tables), the DBMS can be changed to another type and the column datatypes are mapped accordingly.

To map the DBMS type of a table to another DBMS type, follow the steps below:

1. Double-click on the table element in a diagram to open the Table Properties dialog.
2. The Database field shows the current DBMS for this table.
3. To map the column datatypes to another DBMS, select the target from the Database drop-down and click on the Apply button.
4. The datatypes are converted to match those of the new DBMS, and these are reflected in any DDL generated from this table.
12.14 Data Type Conversion for a Package

The DBMS Package procedure or mapper enables you 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 tables.

To map the DBMS types of a package to another DBMS type, follow the steps below:

1. Right-click on the package in the Project Browser to display the context menu.
2. Select the Code Engineering | Reset DBMS Options menu option. The Manage DBMS Options dialog displays.

   ![DBMS Options dialog]

3. In the Current DBMS field, click on the drop-down arrow and select the current DBMS. In the New DBMS field click on the drop-down arrow and select the target DBMS.
4. Select the Convert DBMS Type checkbox.
5. If there are child packages that also require changing, select the Process Child Packages checkbox.
6. Click on the OK button. All tables in the selected packages are mapped to the new DBMS.

To change the owner of the table or all of the tables in a package, follow the steps below:

1. Right-click on the package in the Project Browser to display the context menu.
2. Select the Code Engineering | Reset DBMS Options menu option. The Manage DBMS Options dialog displays.

   ![DBMS Options dialog with change owner option]

3. In the Current Owner field, enter the current owner's name. In the New Owner field, enter the new owner's name.
4. Select the Change Table Owner checkbox.
5. Click on the OK button. The selected tables are mapped with the new owner's name.
3. In the **New Owner** field, type the name for the new table owner.

4. In the **Current Owner** field, click on the drop-down arrow and select the current owner to change, or select `<All>` to change the ownership of all tables in the package to the name you typed in the **New Owner** field.

5. Select the **Change Table Owner** checkbox.

6. If there are child packages that also require changing, select the **Process Child Packages** checkbox.

7. Click on the **OK** button. The ownership changes for all Tables in the selected packages with the specified current owner.

For more information on setting the table owner see the [Set Table Owner](#) topic. To display the table owner in the current diagram see the [Diagram Properties](#) topic.
12.15 **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 enables you to add and configure custom data types. For some data types you must add the size and precision, defaults and maximum values.

To access the **Database Datatypes** screen, select the **Settings | Database Datatypes** menu option. You can also add a DBMS product and configure the inbuilt data types.

You can also map database datatype sizes between products. To do this, follow the steps below:

1. On the **Database Datatypes** dialog, click on the **Datatype Map** button. The **Database Datatypes Mapping** dialog displays.
2. In the **From Product Name** field, click on the drop-down arrow and select the DBMS product to map datatypes *from*. The **Defined Datatypes for Databases** panel displays all the defined datatypes for the product and, where appropriate, their sizes and values.

3. Click on the datatype to map (this must have a defined size unit and value). The **Datatype** and **Common type** fields under the **From Product Name** field display this datatype.

4. In the **To Product Name** field, click on the drop-down arrow and select the DBMS product to map datatypes *to*. The **Datatype** and **Common Type** fields under this field display the corresponding values to those in the fields for the *from* product.

5. In the **Size** panel, click on the radio button for the appropriate size unit and type the default values in the corresponding data fields.

6. Click on the **Save** button to save the mapping.

7. To map further datatypes, repeat this process from step 3.

8. When you have finished mapping datatypes, click on the **Close** button, and again on the **Database Datatypes** dialog.
12.16 Import Database Schema from ODBC

Analysis of legacy database systems is possible using Enterprise Architect’s reverse engineering capabilities. By connecting to a live database via ODBC, you can import the database schema into a standard UML model. Subsequent imports enable you to maintain synchronization between the data model and the live database.

Enterprise Architect supports importing database tables from an ODBC data source. Tables are imported as stereotyped Classes with suitable data definitions for the source DBMS.

Note:
Import of stored procedures and views is supported for: DB2; SQL Server; Firebird/Interbase; Informix; Ingres; Oracle 9i, 10g and 11g; MySQL; PostgreSQL; Sybase Adaptive Server Enterprise (ASE) and Sybase Adaptive Server Anywhere (ASA).

Import Database Tables and Stored Procedures
To import database tables and stored procedures, follow the steps below:

1. Select any package in the Logical View.
2. To import into:
   - The package only, right-click on the package to display the context menu, and select the Code Engineering | Import DB Schema from ODBC menu option.
   - A diagram, right-click on the diagram in the selected package to open the context menu, and select the Import DB schema from ODBC menu option.

Note:
Alternatively you can select the Project | Database Engineering | Import DB Schema from ODBC menu option.

The Import DB Schema from ODBC Source dialog displays.
3. In the Database Name field, click on the [...] (Browse) button and select a suitable ODBC data source from the ODBC dialog (ODBC must be installed and configured on your machine for this to work correctly).

4. If importing from Oracle, to restrict the import to a specific owner, type the owner name in the Schema/Owner field.

   For imports from other types of database, leave this field blank.

5. In the Filter panel, select the appropriate checkboxes for additional items to include in the import.

   If you select to import User Stored Procedures, User Functions and/or User Sequences as individual Classes, then they are imported as separate elements and the Properties dialog is solely concerned with the Stored Procedure, Function or Sequence definition.

   If you select to import them as Class operations, then they are imported as operations (methods) and you view and edit them through the Operations Properties dialog of the parent Class.

6. When synchronizing existing Classes, select the appropriate checkbox in the Synchronization panel to determine whether the model comments, default values or constraints are to be synchronized with the ODBC tables.

7. To also import stored procedures, select the Include User Stored Procedures checkbox. To import user views, select the Include User Views checkbox.
**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 displays to give the option to cancel the import or to continue importing into the package only. The **Import DB Schema from ODBC Source** dialog includes checkbox options to import into the diagram and package, or into the package only.

If no diagram is open, the **Package Only** radio button defaults to selected and the options are disabled. If the open diagram is in the selected package, you can select either option.

8. Click on the **Import** button to start the import.
9. **Select the tables** and - if appropriate - stored procedures to import.

This completes the procedure. See the **Imported Class Elements** topic.
### 12.16.1 Select a Data Source

To import DDL from existing data sources, you must have a suitable ODBC connection installed and configured. From the Import DB Schema from ODBC Source dialog you can select the ODBC data source using the standard windows ODBC set-up dialog. Click on the data source name and then click on the OK button.

<table>
<thead>
<tr>
<th>Data Source Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL_SPARK</td>
<td>System</td>
<td>MySQL ODBC 3.51 Driver DSN</td>
</tr>
<tr>
<td>MISISCA</td>
<td>System</td>
<td>MySQL ODBC 3.51 Driver DSN</td>
</tr>
<tr>
<td>sample_MySQL</td>
<td>User</td>
<td>MySQL ODBC Driver</td>
</tr>
<tr>
<td>SQLSVI</td>
<td>System</td>
<td></td>
</tr>
<tr>
<td>Text Files</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>Visual FoxPro Database</td>
<td>User</td>
<td></td>
</tr>
<tr>
<td>Visual FoxPro Tables</td>
<td>User</td>
<td></td>
</tr>
</tbody>
</table>

A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.
12.16.2 Select Tables

When you have opened the ODBC data source, Enterprise Architect acquires 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 stored procedures to import and clear those you do not require.

Selection shortcuts:
- To select all tables and procedures, click on the **Select All** button
- To clear all tables and procedures, click on the **Select None** button
- Hold down [Ctrl] while clicking on tables and procedures to select multiple objects
- Hold down [Shift] and click on tables and procedures to select a range.

When you have selected the tables and procedures, click on the **OK** button.
12.16.3 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 XIII
13 MDA Transformations

Model Driven Architecture (MDA) Transformations provide a fully configurable way of converting model elements and model fragments from one domain to another. This typically involves converting Platform-Independent Model (PIM) elements to Platform-Specific Model (PSM) elements. A single element from the PIM can be responsible for creating multiple PSM elements across multiple domains.

Transformations are a huge productivity boost, and reduce the necessity of manually implementing stock Classes and elements for a particular implementation domain: for example, database tables generated from persistent PIM Classes. Enterprise Architect 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 further Transformations available over time, either as built in Transformations or as downloadable modules from the Sparx Systems website.

For a further productivity boost, Enterprise Architect can automatically generate code for your transformed Classes that target code languages. See the Generate Code on result option on the Model Transformation dialog.

A Transformation is defined using Enterprise Architect's simple code generation template language, and involves no more than writing a template to create a simple intermediary source file. Enterprise Architect reads the source file and binds that to the new PSM.

Enterprise Architect also creates internal bindings (Transformation Dependencies) between each PSM created and the original PIM. This is essential, as it enables you to forward synchronize from the PIM to the PSM many times, adding or deleting features as you go. For example, adding a new attribute to a PIM Class can be forward synchronized to a new column in the Data Model. You can observe the Transformation Dependencies for a package using the Hierarchy Window. This enables you to check the impact of changes to a PIM element on the corresponding elements in each generated PSM, or to verify where a change required in a PSM should be initiated in the PIM (and also to reflect back in other PSMs). The Transformation Dependencies are a valuable tool in managing the traceability of your models.

Enterprise Architect does not delete or overwrite any element features that were not originally generated by the transform. Therefore, you can add new methods to your elements, and Enterprise Architect does not act on them during the forward generation process.

**Note:**

If you are using the Enterprise Architect Corporate edition, if security is enabled you must have Transform Package access permission to perform an MDA Transform on a package.

The following diagram highlights how Transformations work and how they can significantly boost your productivity:
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 topics:
- Transform Elements
- Import Transformations
- Transformation Templates
- Built-in Transformations
- Write Transformations
- Chaining Transformations
13.1 Transform 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:
  - Select the Project | Model Transformations | Transform Selected Elements menu option, or
  - From the context menu for the Classes on the diagram, select the Transform Selected Element(s) option.

- To transform elements in the package currently selected in the Project Browser, either:
  - Select the Project | Model Transformations | Transform Current Package menu option, or
  - From the context menu of the package in the Project Browser, select the Transform Current Package option.

The Model Transformation dialog displays.

When the dialog displays, all elements are selected and all transformations previously performed from any of these Classes are checked.

<table>
<thead>
<tr>
<th>Elements:</th>
<th>Transformations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element Name</td>
<td>Element Type</td>
</tr>
<tr>
<td>RetirementSpecialistImpl</td>
<td>Class</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- [ ] Generate Code on result
- [ ] Perform Transformations on result

Intermediate File (optional for debugging only):

C:\Documents and Settings\John\Desktop\lab.txt

- [ ] Write Always
- [ ] Write Now

Do Transform

Close

Help

When the dialog displays, all elements are selected and all transformations previously performed from any of these Classes are checked.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>Select (click on) the individual elements to be included in the transformation.</td>
</tr>
<tr>
<td>All</td>
<td>Select all of the elements from the list to be included in the transformation.</td>
</tr>
<tr>
<td>None</td>
<td>Deselect all of the elements from the list.</td>
</tr>
<tr>
<td>Transformations</td>
<td>Select which transformations to perform and the package each of them should be transformed to. (Use the [...] button to select the package in which the transformed elements are being created.)</td>
</tr>
</tbody>
</table>

Enterprise Architect User Guide
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Code on result</td>
<td>Specify whether or not to automatically generate code for transformed Classes that target code languages.</td>
</tr>
<tr>
<td></td>
<td>Automatically generating code helps boost productivity in development. With this option selected, the first time you transform to the selection Class Enterprise Architect enables you to select a filename to generate to. Subsequent transformations automatically generate any Class with a filename set.</td>
</tr>
<tr>
<td>Perform Transformations on result</td>
<td>Specify if transformations previously done on target Classes should be automatically executed. See <a href="#">Chaining Transformations</a> for more information.</td>
</tr>
<tr>
<td>Intermediary File</td>
<td>Specify the filename of the intermediary file (if any).</td>
</tr>
<tr>
<td>Write Always</td>
<td>Write the intermediary file to disk.</td>
</tr>
<tr>
<td>Write Now</td>
<td>Generate the intermediary file but do not perform the transform.</td>
</tr>
<tr>
<td>Do Transform</td>
<td>Execute the transform command.</td>
</tr>
</tbody>
</table>
13.1.1 **Chaining Transformations**

Chaining transformations provide an extra degree of flexibility and power to transformations. For example, you might have a situation where two transformations have a common element. This can be separated out 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.

Enterprise Architect provides for chaining transformations, by enabling transformations that have already been performed on target Classes to be performed automatically next time that Class is transformed to. To enable this, select the **Perform Transformations on result** checkbox in the **Model Transformation** dialog.
13.2 Import Transformations

You can transfer Transformation templates between models. To import a Transformation template, follow the steps below:

1. Select the **Tools | Import Reference Data** menu option. The **Import Reference Data** dialog displays.

2. Click on the **Select File** button and browse to a .XML file containing the required Transformation template.

3. Select the name of one or more template datasets and click on the **Import** button.
### 13.3 Transformation Templates

To modify Transformation templates:

1. Select the **Settings** | **Transformation Templates** menu option. The Transformation Templates Editor displays.
2. In the Language field, type or select the name of the transformation to modify.
3. Select a template from the Templates list, and edit its contents in the editor pane.
4. Click on the **Save** button.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td>Select the name of the transformation.</td>
</tr>
<tr>
<td><strong>Template</strong></td>
<td>Display the contents of the active template. Provide the editor for modifying templates.</td>
</tr>
<tr>
<td><strong>Templates</strong></td>
<td>List the base transformation templates. The active template is highlighted. The <strong>Modified</strong> field indicates whether you have changed the default template for the current transformation.</td>
</tr>
<tr>
<td><strong>New Transformation Type</strong></td>
<td>Create a new transformation.</td>
</tr>
<tr>
<td><strong>Stereotype Overrides</strong></td>
<td>List the stereotyped templates, for the active base template. The <strong>Modified</strong> field indicates whether you have modified a default stereotyped template.</td>
</tr>
</tbody>
</table>

<p>| <strong>Add New Stereotyped Override</strong> | Invoke a dialog for adding a stereotyped template, for the currently selected base template. |
| <strong>Add New Custom Template</strong>     | Invoke a dialog for creating a custom stereotyped template. |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Launch the Enterprise Architect help file.</td>
</tr>
<tr>
<td>Get Default Template</td>
<td>Update the editor display with the default version of the active template.</td>
</tr>
<tr>
<td>Save</td>
<td>Overwrite the active templates with the contents of the editor.</td>
</tr>
<tr>
<td>Delete</td>
<td>If you have overridden the active template, delete the override and replaced it with the corresponding default transformation template.</td>
</tr>
</tbody>
</table>

Note that the Transformation Template mechanism is based very strongly on the Code Generation Template mechanism. For further information on Transformation Templates see the [Code Template Editor](#) section of SDK for Enterprise Architect.
13.4 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 Enterprise Architect:

- C#
- DDL
- EJB Entity
- EJB Session
- Java
- JUnit
- NUnit
- WSDL
- XSD
13.4.1 C# Transformation

The C# Transformation converts Platform-Independent Model (PIM) elements to language-specific C# Class elements. The transformation converts PIM model types to C# types and creates encapsulation according to Enterprise Architect’s options for creating properties from C# attributes, which you set on the C# Specifications page of the Options dialog.

The PIM:

After Transformation, becomes the PSM:
### 13.4.2 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 can then be used to automatically generate DDL statements to run in one of the Enterprise Architect supported database products.

It uses and demonstrates support in the intermediary language for the following database-specific concepts:

<table>
<thead>
<tr>
<th>Table</th>
<th>Mapped one-to-one onto Class elements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Mapped one-to-one onto attributes.</td>
</tr>
<tr>
<td>Primary Key</td>
<td>Lists all the columns involved; this ensures that they exist in the Class and creates a primary key method for them.</td>
</tr>
<tr>
<td>Foreign Key</td>
<td>This is a special sort of connector. In the Source and Target sections, lists all of the columns involved; this ensures that they exist and that a matching primary key exists in the destination Class, and creates the appropriate foreign key.</td>
</tr>
</tbody>
</table>

The following two diagrams show a typical PIM to Data Model Transformation.

**The PIM:**

After transformation becomes the PSM:
Generalizations are handled by providing the child element with a foreign key to the parent element, as in the following diagram. Copy-down inheritance is not supported.
13.4.3 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 enabling you to concentrate on modeling at a higher level of abstraction.

The EJB Session Bean Transformation generates 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 generates 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 also generate a META-INF package containing a deployment descriptor element.

The Platform-Independent Model (PIM):

After transformation generates a set of Entity Beans, where each one takes the following form (for the Account Class):

- **Account**
  - `billingAddress: String`
  - `deliveryAddress: String`
  - `emailAddress: String`
  - `name: String`
  - `createNewAccount(): void`
  - `loadAccountDetails(): void`
  - `markAccountClosed(): void`
  - `renewAccountData(): void`
  - `submitNewAccountDetails(): void`

- **Order**
  - `date: Date`
  - `deliveryInstructions: String`
  - `orderNumber: String`
  - `cancelOutstandingOrder(): void`

- **OrderLine**
  - `date: Date`
  - `orderNumber: String`
  - `quantity: integer`

- **ShoppingBasket**
13.4.4 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 Enterprise Architect’s options for creating properties from Java attributes; that is, producing the getters and setters according to the rules you have defined.

You set the code generation options for Java code generation on the Java Specifications page of the Options dialog.

The PIM:

![Diagram of Account, Order, Transaction, ShoppingBasket, and StockItem classes with their attributes and methods.]

After Transformation becomes the PSM:
13.4.5 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.

The Java model originally transformed from the PIM:

After Transformation becomes the PSM:
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 your responsibility to fill in the details of each test.
13.4.6 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.

The C# model originally transformed from the PIM:

After Transformation becomes the PSM:
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 your responsibility to fill in the details of each test.
13.4.7 WSDL Transformation

The purpose of the WSDL transformation is to create from a simple model an expanded model of a WSDL interface that is suitable for generation.

Take the following example interface.

```
@interface
HelloWorld
+ SayHello(name: String): string
```

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 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 Enterprise Architect, and finally be generated using WSDL generation.
13.4.8 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 an «XSDcomplexType» element.

For more information, see the XML Schema Generation topic.

The PIM:

After Transformation becomes the PSM:
Which in turn generates the following XSD:

```xml
<?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="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 name="LineItem"/>
      <xs:element name="status" type="OrderStatus"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
<xs:schema>
  <xs:element name="ShoppingBasket" type="ShoppingBasket"/>
  <xs:complexType name="ShoppingBasket">
    <xs:sequence>
      <xs:element ref="LineItem"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType 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: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: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>
13.5 Write Transformations

This topic 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 from a close study of the Transformation Templates provided with Enterprise Architect. Note also that writing transformations is very similar to writing code generation templates, so an understanding of the Code Template Framework can greatly assist in understanding transformations.

Transformation Templates are accessed from the Settings | Transformation Templates menu option.
13.5.1 Default Transformation Templates

In most transformations, there is a lot of information that is simply copied to the target model. In order to make writing new transformations simpler, Enterprise Architect 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 required changes.

**Note:**

When creating a new transformation you must modify at least one template before the transformation becomes available.
13.5.2 Intermediary Language

All transformations in Enterprise Architect work by generating a text form of the model to generate. Any element is represented in this language by the type of element (e.g., 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 resembles the following.

- elementName
  - elementProperty (elementProperty | element)*
- packageName
  - stereotype
  - propertyName = "propertyValueSymbol" "

- packageName
  - name = "propertyValueSymbol" (" propertyValueSymbol" )*
  - 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 = """"Some string value."""""
13.5.3 **Objects**

Objects are created in Enterprise Architect by generating text in the following form:

```
objectType
  {  
    objectProperties
  }
```

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 one 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 one or more of the following elements:

- Attribute
- Classifier
- Parameter
- Operation
- Parent
- Tag
- XRef
- Any object.

Notes:

- Some of the above only apply to certain object types.
- Every object created in a transformation should include an XRef element, as it enables Enterprise Architect 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:

```java
Class
{
    name = "Example"
}
```

It is then easy to add to this. The following example sets the language to C++, and adds a Tagged Value and an attribute:

```java
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
- Classifier
- Collection
- Container
- Containment
- Constant
- Default
- Derived
- LowerBound
- Name
- Notes
- Ordered
- Scope
- Static
- Stereotype
- Type
- UpperBound
- Volatile.

and the following elements:

- Classifier
- Tag
- XRef.

Operations
Operations are created with the same structure as objects, and include the following properties:

- Abstract
- Alias
- Behavior
- Classifier
- Code
- Constant
- IsQuery
- Name
- Notes
- Pure
- ReturnArray
- Scope
- Static
- Stereotype
- Type.

and the following elements:
- Classifier
- Parameter
- Tag
- XRef.

**Parameters**

Parameters are created with the same structure as objects, and include the tag element and the following properties:
- Classifier
- Default
- Fixed
- Name
- Notes
- Kind
- Stereotype.

**Packages**

Packages differ from other objects in the following ways:
- A reduced set of properties of alias, author, name, namespaceRoot, notes, scope, stereotype and version
- The 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 differences from other object types:
- Can include columns and primary keys
- Can't include attributes.

**Columns**

Columns are similar to attributes, but have an autonumber element containing Startnum and increment, and the following added properties:
- Length
- NotNull
- Precision
- PrimaryKey
- Scale
- Unique.
**Note:**

In the column definition, you cannot assign a value to the **NotNull, PrimaryKey or Unique** properties.
13.5.4 Connectors

Creating connectors in a transformation can be complex, but the process has the same form as creating elements. The difference is that you must also specify each end.

The different connectors that can be created are as follows.

- Aggregation
- Assembly
- Association
- Collaboration
- ControlFlow
- Connector
- Delegate
- Dependency
- Deployment
- ForeignKey
- Generalization
- InformationFlow
- Instantiation
- Interface
- InterruptFlow
- Manifest
- Nesting
- NoteLink
- ObjectFlow
- Package
- Realisation
- Sequence
- Transition
- UseCase
- Uses

**Note:**

ForeignKey is a special case where not just a connector is created; you must also list the columns involved in the transformation. In addition, tags specified on the connector are actually created on the foreign key operation in the source Class, and a cascade property can be added; for example, `cascade="update","delete"`.

There are two different types of Class that you can use as a connector end: one created by a transformation, and one for which you already know the GUID.

Connect to a Class Created by a Transformation

The most common connection is to connect to a Class created by a transformation. To do this you must have three items of information:

- The original Class GUID
- The name of the transformation
- The name of the transformed Class.

This type of connector 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 simplest example of this is when you have created multiple Classes from a single Class in a transformation and want a connector between them. Consider this example from the EJB Entity transformation:

```xml
  %TRANSFORM_REFERENCE("EJBRealizeHome",classGUID)%
  stereotype="EJBRealizeHome"
  Source
```
There are three uses of the \texttt{TRANSFORM\_REFERENCE} macro: one to identify this connector for synchronization 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 have to specify the transformation because the two references are referencing something in the current transformation. Each of them then only has 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 have to worry about only generating it once, because if you have created a \texttt{TRANSFORM\_REFERENCE} for the connector then Enterprise Architect automatically synchronizes them. The following copies the source connector.

\begin{verbatim}
%connectorType% {
   %TRANSFORM\_CURRENT()%
   %TRANSFORM\_REFERENCE("Connector",connectorGUID)%
   Source {
      %TRANSFORM\_REFERENCE("Class",connectorSourceGUID)%
      %TRANSFORM\_CURRENT("Source")%
   }
   Target {
      %TRANSFORM\_REFERENCE("Class",connectorDestGUID)%
      %TRANSFORM\_CURRENT("Target")%
   }
}
\end{verbatim}

\textbf{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 for which you know the current GUID. 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.

\begin{verbatim}
Dependency {
   %TRANSFORM\_REFERENCE("SourceDependency",classGUID)%
   stereotype="transformedFrom"
   Source {
      %TRANSFORM\_REFERENCE("Class",classGUID)%
   }
   Target {
      GUID=%qt%classGUID%qt%
   }
}
\end{verbatim}
13.5.5 Duplicate Information

In many transformations there is a substantial amount of information to 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 \texttt{TRANSFORM\_CURRENT} function macro to do exactly this.

\textbf{TRANSFORM\_CURRENT(\texttt{<listOfExcludedItems>})}

Generates an exact copy of all the properties of the current item, except for the items named in \texttt{<listOfExcludedItems>}. Another form of this is available when transforming connectors that enables either end of the connector to be copied.

\textbf{TRANSFORM\_CURRENT(\texttt{<connectorEnd>,<listOfExcludedItems>})}

Generates an exact copy of the connector end specified by \texttt{<connectorEnd>} except for the items named in \texttt{<listOfExcludedItems>}, where \texttt{<connectorEnd>} is either \texttt{Source} or \texttt{Target}.
13.5.6 Convert Types

Different target platforms almost certainly require different types, so you often require a method of converting between different types. The following macro offers this.

CONVERT_TYPE(<destinationLanguage>, <originalType>)

Converts <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.
13.5.7 Convert Names

Different target platforms use different naming conventions. As a result you might not want to copy the names of your elements directly into the transformed models. To facilitate this requirement, Enterprise Architect's transformation templates provide a `CONVERT_NAME` function macro.

Another way in which you can transform a name is to remove a prefix from the original name, with the `REMOVE_PREFIX` macro.

**CONVERT_NAME(<originalName>, <originalFormat>, <targetFormat>)**

This macro converts `<originalName>`, which is assumed to be in `<originalFormat>`, to `<targetFormat>`.

The supported formats are:

- **Camel Case**: New words start with a capital letter except for the first word, which begins with a lower case letter; for example, `myVariableTable`
- **Pascal Case**: Same as Camel Case but the first letter of the first word is upper case; for example, `MyVariableTable`
- **Spaced**: Words are separated by spaces; the case of letters is ignored
- **Underscored**: Words are separated by underscores; the case of letters is ignored.

**Note:**

Acronyms are not supported when converting from Camel Case or Pascal Case.

The original format might also specify a list of delimiters to be used. For example a value of '_ ' breaks words whenever either a space or underscore is found.

The target format might also use a format string that specifies the case for each word and a delimiter between them. It takes the following form:

```<firstWord><delimiter><otherWords>`

- `<firstWord>` controls the case of the first word (see below)
- `<delimiter>` is the string generated between words
- `<otherWords>` applies to all words after the first word.

`<firstWord>` and `<otherWords>` are both a sequence of two characters. The first character represents the case of the first letter of that word, and the second character represents the case of all subsequent letters. An upper case letter forces the output to upper case, a lower case letter forces the output to lower case, and any other character preserves the original case.

**Example 1:** To capitalize the first letter of each word and separate multiple words with a space:

```
"Ht( )Ht" to output "My Variable Table"
```

**Example 2:** To generate the equivalent of Camel Case, but reverse the roles of upper and lower case; i.e. all characters are upper case except for the first character of each word after the first word:

```
"HT( )hT" to output "MY vARIABLE tABLE"
```

**REMOVE_PREFIX(<originalName>, <prefixes>)**

This macro removes any prefix found in `<prefixes>` from `<originalName>`. The prefixes are specified in a semi-colon separated list.

The macro is often used in conjunction with the `CONVERT_NAME` macro. For example, this code 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%```
### 13.5.8 Cross References

Cross References are an important part of transformations. They are used to:

- Find the transformed Class to synchronize with
- Create connectors between transformed Classes
- Specify a classifier of a type
- Determine where to transform to for future transformations.

Each cross reference has three different parts:

- A *Namespace*, corresponding to the transformation that generated the element
- A *Name*, which is a unique reference to something that can be generated in the above transformation
- A *Source*, which 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` macro that is defined for this purpose. It has three optional parameters.

**TRANSFORM_REFERENCE**(*name*, *sourceGuid*, *namespace*)

Generates a reference that can be used in the ways described above. It resembles 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 Enterprise Architect. 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 must identify the two Classes it connects using cross references while having its own unique cross reference.

### Specify Classifiers

Objects, attributes, operations and parameters can all reference another element in the model as their type. When this type is created from a transformation you must use a cross reference to specify it, using the `TRANSFORM_CLASSIFIER` macro.

**TRANSFORM_CLASSIFIER**(*name*, *sourceGuid*, *namespace*)

Generates a cross reference within a classifier element, where the parameters are identical to the `TRANSFORM_REFERENCE` macro but the name `Classifier` is generated instead of `XRef`.

If the target classifier already exists in the model before the transformation, a `TRANSFORM_CLASSIFIER` is inappropriate and instead the GUID can be given directly to a classifier attribute.

**Note:**

If a classifier is specified for any type it overrides the type specified.
Part XIV
14 Enterprise Architect Reports

Model documentation is essential to realizing the full benefit of Enterprise Architect. Enterprise Architect provides a powerful mechanism for generating high quality, customized documentation directly from your model, in either RTF or HTML format.

There are many ways to specify the Enterprise Architect content being documented. You can:

- Document a package and/or its child packages by manually highlighting the package and selecting a documentation control
- Specify embedded packages for exclusion if child packages are recursively documented
- Link a package to an RTF document template to simplify generating consistent types of documentation (e.g. Use Case Reports) using the Documents feature
- Select, group and order packages together in a different manner from the Project Browser by creating 'Virtual' Documents either linked through a master document with headers, footers and contents list, or as separate individual documents.

RTF Documentation

Rich text reports are documents produced by Enterprise Architect in Rich Text Format (RTF). 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 separate RTF documents output from the model by connecting and interweaving them into a linked master document with headers, footers and contents list.

Enterprise Architect has a fully-featured RTF Document Generator that features:

- Powerful WYSIWYG RTF style template editor support
- An easy-to-use document generator
- An embedded RTF viewer that enables you to view RTF documents generated by Enterprise Architect within Enterprise Architect.

For further information, see:

- RTF Documents
- Use MS Word

RTF Reports

You can also generate a number of RTF reports on different aspects of your model. See the Other Documents topic.

HTML Documentation

Enterprise Architect provides automated web-based publishing of models, making it simple to explore large models efficiently on-line. Enterprise Architect enables the export of an entire model or a single branch of the model to HTML Web pages. You can also create web style templates to customize the HTML output.

For further information, see the HTML Reports topic.
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 enables you to link a number of rich text documents into a single master document.

Typically you create a Word master document, then some Enterprise Architect RTF reports. You link the reports back into sub-sections of the master document, and refresh them as required during project development. In this way the project document becomes an easily-managed and feature-rich work product.

You can also populate a Word document from specific sections of reports, based on bookmarks. For example, a Word document might 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 Enterprise Architect reports. The RTF Generator performs one pass for one template, but using a Word master document and Enterprise Architect bookmarks enables you to incorporate material from several RTF documents with different formats based on different templates.

By adding tables of contents, figure tables, sections, and headers and footers, you can manage a complex document with relative ease. Simply update the Enterprise Architect RTF reports then refresh the links in MS Word.

You can also maintain complex documents by creating virtual documents in Enterprise Architect, setting up a Master Document (package) element and/or Model Document elements (Class elements of stereotype Model Document) and linking packages into the document, in whatever order or combination is most appropriate to your requirements. You can select packages from different areas of the model, arrange them in any order, and edit or delete the packages. The virtual document automatically incorporates the changes each time you generate it.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have Generate Documents permission to generate RTF documents.

The RTF Generator

Enterprise Architect has an enhanced RTF Document Generator that features:

- Powerful WYSIWYG RTF style template editor support, enabling:
  - 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.

- A document generator that:
  - Provides simplified options
  - Generates complex documents based on RTF templates.

- An embedded RTF viewer that you use to view RTF documents generated in Enterprise Architect directly within Enterprise Architect.
More Information

A tutorial on using the RTF Generator and creating RTF documentation is provided on the Sparx Systems website. Click on the following link:

http://www.sparxsystems.com/resources/whitepapers/

For more information, see:

- Generate RTF Documents
- Generate RTF Documentation Dialog
- RTF Document Options
- RTF Templates Dialog
- Custom Language Settings
- Include or Exclude a Package from Report
14.1.1 Generate RTF Documents

Creating a Rich Text Format (RTF) document is a simple and flexible process. An RTF document is based on a package or an element in your project (more usually a package). To produce a document, you must select the package or element to report on in the Project Browser, Element List or Model Search.

**Tip:**

Reports can be configured to include all packages within a parent package, or just the top level.

You should also set the diagram properties to determine how the diagrams in the package are set out in the RTF document. When you have prepared and selected your package, use the context menu to open the Generate RTF Documentation dialog and configure the details of your document. The next topic guides you through creating a rich text report.

Open the Generate RTF Documentation Dialog

Use one of the following methods:

- Select the **Project | Documentation | Rich Text Format (RTF) Report** menu option
- In the **Project Browser**, right-click on the required package and, on the context menu, select the **Documentation | Rich Text Format (RTF) Report** menu option
- In the **Project Browser** or a diagram, click on the required package or element and press [F8]
- In a diagram, click on a specific element and select the **Element | Rich Text Format (RTF) Report** menu option
- In the **Element List** or **Model Search**, select one or more items, right-click and, from the context menu, select either the **RTF Report | Generate report for each selected object** option or the **RTF Report | Generate one report for all selected** option.

See the **Generate RTF Documentation Dialog** topic for more information.

Quick Start

To generate an RTF report right now, follow the steps below:

1. Open the **EAExample project**.
2. Open the **QA Model** package and right-click on the **Testing** package.
4. In the **Output to file** field, select a convenient file location in which to hold the generated report.
5. In the **Use Template** field, click on the drop-down arrow and select (basic template).
6. Click on the **Generate** button.
7. When the report has been generated, click on the **View** button.

Generate RTF Report From Element List or Model Search

When you select to create an RTF Report from the **Element List** or **Model Search** tools, you can generate an element-level report rather than a package-level report, and you have additional flexibility in selecting:

- The type of element to report on
- The specific elements to report on, together or separately, whether in the same package or not.

For example, you might want to find all elements with test cases, with the intention of reporting on some or possibly all such elements. With the **Element List**, you would identify these elements yourself within the list of all elements in a selected package, but with the **Model Search** you could specifically identify the elements across a section of the model or across the whole model, as required. The search filtering could be for specific test cases; however, the results are by element so if there are test cases outside the range in any element that has a filtered test, these elements are listed as well.

Having generated the list of elements, you can select individual elements, blocks of elements, or all elements, and then (as above) use the context menu to generate a report on all of the elements, or separate reports on each element.
14.1.1.1 Diagram Options

This topic refers to options on the Diagram page of the Diagram Properties dialog, used when generating RTF reports for a particular diagram (using either the extended or Legacy RTF Report Generator). Display the dialog page by double-clicking on the diagram background, and select the Diagram tab.

Exclude Image from RTF Documents
Select this checkbox to exclude the image of the current diagram from any RTF reports.

Document Each Contained Element in RTF
Select this checkbox to ensure the RTF generators include details of elements that belong in other - external - packages but that are linked in to this diagram within the package being reported on. The relevant section in your (customized) document generation templates must also be enabled in order to generate this information in the report.

To enable the template section, select the Project | Documentation | Rich Text Format (RTF) Report menu option and select the appropriate customized template (not a system-provided one) in the Use Template field. Click on the Edit Template button to display the RTF Template Editor. In the Sections panel on the left-hand side of the editor window, select the Package::Diagram::Element checkboxes.

Notes:

- If using the Legacy RTF generator, the Element checkbox is automatically checked in your customized template, when you select the Document each contained element in RTF checkbox.
- Package::Diagram::Element, if left blank, replicates the format of: Package::Element including sub-element sections (e.g. Package::Element::Scenario). Package::Diagram::Element does not have an option to add these sections.

Selecting the checkboxes adds the following set of sections to your report template:

```
package>
  [right-click-to-insert-Package-field(s)]

  diagram>
  [right-click-to-insert-Diagram-field(s)]

  elements>
  [right-click-to-insert-Element-field(s)]

  <element>
  [right-click-to-insert-Element-field(s)]

  </element>

  </diagram>

  </package>
```

To report on the linked elements in:
the same style as defined in the Package::Elements section, delete the [right-click-to-insert-Element-field(s)] text to leave a blank area

• a different style to the elements within the selected package, set up the style as appropriate.

**Divide Diagram Into Multiple Pages**
Select this checkbox to divide each large diagram into separate pages in the RTF document.

**Note:**
This option is only effective when the Scaled Printing option is set to None on the Print Advanced dialog.

**Rotate Images**
Select this checkbox to rotate each diagram image by 90 degrees in the RTF document.

**Note:**
Only valid for bitmap (.bmp) images.

### 14.1.1.2 Exclude Package from Report
You can exclude a particular package from any RTF reports by marking it for exclusion.

**Notes:**
• By default, packages are included in any RTF reports.
• When you exclude a package from RTF reports, all of the selected package's subpackages are also excluded.

**To Mark a Package for Exclusion**
1. In the Project Browser, right-click on the required package. The context menu displays.
2. Select the Documentation | RTF Report Options menu option. The RTF Generation Options dialog displays.
3. Select the Exclude Package from RTF Reports radio button.
4. Click on the OK button to save changes.

### 14.1.1.3 Generate RTF Documentation Dialog

**Note:**
For an introduction to generating RTF documentation, see RTF Documents.

The Generate RTF Documentation dialog enables you to set the exact contents and look and feel of your report.
Each tab of the dialog offers a number of RTF document generation options, as described in the following sections:

- **Generate Tab (see below)**
- **RTF Templates**
- **Document Options**
- **Advanced Options**
- **Word Substitution**
- **Language Substitution (Codepage)**

### Generate Tab Options

The **Generate** tab of the dialog has the following fields:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Document</strong></td>
<td>Confirm the name of the element selected from the Project Browser, Element List or Model Search. If this is the specially-created model document element for a Virtual Document, the field is Model Document. Otherwise, this field identifies the selected element of the hierarchy to be reported on; i.e. the Root Element or Root Package.</td>
</tr>
<tr>
<td><strong>Root Element</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Root Package</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Output to File</strong></td>
<td>Type or select the location and filename for the generated documentation. The [...] (Browse) button enables you to navigate to the location.</td>
</tr>
<tr>
<td><strong>Use Template</strong></td>
<td>Type or select the name of the RTF template to apply to document generation. You can select either a standard template (enclosed in parentheses) or a user-generated template.</td>
</tr>
<tr>
<td><strong>Use Language Substitutions</strong></td>
<td>Switch custom language word substitutions on. Deselect to switch custom language word substitutions off.</td>
</tr>
<tr>
<td><strong>View Document On Completion</strong></td>
<td>Open the document as soon as it has been generated.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Use Internal Viewer</td>
<td>Enable the View button to launch the generated RTF Documentation in the Enterprise Architect internal viewer. Deselect to enable the View button to launch the generated RTF Documentation in the MS Windows default RTF file viewer.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>If you use Open Office as your default document editor, you must select the Overwrite Document Fields checkbox in the document options in order to show field value text. Open Office defaults to showing field codes only, and you cannot toggle to the field values.</td>
</tr>
<tr>
<td>Include all Diagram</td>
<td>Include elements from external packages that are referenced from a diagram in the report when the Diagram.Element section is enabled in the selected template. When deselected, or when the Diagram.Element section is not enabled in the selected template, only elements in the current package are documented.</td>
</tr>
<tr>
<td>Elements in Report</td>
<td>Generate the document.</td>
</tr>
<tr>
<td>View</td>
<td>Launch the generated RTF Documentation in the MS Windows default RTF file viewer, or in the Enterprise Architect internal viewer if you have selected the Use Internal Viewer checkbox.</td>
</tr>
<tr>
<td>Edit Template</td>
<td>Edit the currently-named template using the RTF Style Template Editor. You can only edit user-defined templates, not the standard templates provided with Enterprise Architect. Standard template names are enclosed in parentheses.</td>
</tr>
<tr>
<td>Resource Document</td>
<td>Save the current options as a document definition.</td>
</tr>
<tr>
<td>Abort</td>
<td>Cancel report generation.</td>
</tr>
</tbody>
</table>

### 14.1.1.4 RTF Templates Tab

The Templates tab of the Generate RTF Documentation dialog enables you to create, edit and delete your own RTF style templates. You can also import RTF templates saved as XML files.

**Note:**

In the Corporate edition of Enterprise Architect, if security is switched on, you must have Configure Resources access permission to create RTF templates.
The tab has the following functions:

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete a template</td>
<td>Click on the template name and click on the <strong>Delete</strong> button.</td>
</tr>
<tr>
<td>Create a new template</td>
<td>Click on the <strong>New</strong> button. The <strong>New Document Template</strong> dialog displays, on which you specify the template name and the name of any existing template to act as the base for the new template. To make it easier to get up and running, Enterprise Architect provides a basic template with default settings on which you can base new templates. Modify the template as required, using the <strong>RTF Style Template Editor</strong>.</td>
</tr>
<tr>
<td>Open the RTF Style Template Editor</td>
<td>Click on the template name and click on the <strong>Edit</strong> button. The <strong>Document Template Editor</strong> screen displays, presenting the facilities of the <strong>RTF Style Template Editor</strong>.</td>
</tr>
<tr>
<td>Import RTF Templates saved to XML files using the **Tools</td>
<td>Export Reference Data** menu option</td>
</tr>
<tr>
<td>1. Click on the <strong>Import From Reference File</strong> button.</td>
<td></td>
</tr>
<tr>
<td>2. On the <strong>Import Reference Data</strong> dialog, click on the <strong>Select File</strong> button and browse for and select the required file.</td>
<td></td>
</tr>
<tr>
<td>3. In the <strong>Select Datasets to Import</strong> panel, click on the required datasets.</td>
<td></td>
</tr>
<tr>
<td>4. Click on the <strong>Import</strong> button to import the template.</td>
<td></td>
</tr>
<tr>
<td>The imported template displays in the list on the <strong>Templates</strong> tab of the <strong>Generate RTF Documentation</strong> dialog.</td>
<td></td>
</tr>
</tbody>
</table>
Note:

There are two methods of exporting and importing RTF templates out of and into models:

- If the template is in a batch file, export it using the Tools | Export Reference Data menu option and import it using the Templates tab, as above.
- If the template is a one-off copy, use the RTF Template Editor File menu options, Export and Import.

14.1.1.4.1 RTF Style Template Editor

The RTF Style Editor enables you to create and edit custom RTF templates to define output RTF documentation associated with various sections of the RTF Report facility in Enterprise Architect. You typically use this facility to customize the look and feel of a report for your company or client. You access the RTF Style Editor by either:

- Selecting to edit the current template on the Generate tab of the Generate RTF Documentation dialog, or
- Selecting to edit a style template on the Templates tab of the Generate RTF Documentation dialog.

You select particular model components and specify, from the component type, the fields to include in the generated document. You can define formatting styles in the RTF Style Template Editor, and add a range of items such as tables of contents or headers to the document.

For information regarding specific commands to alter the format of the RTF documentation, see the entries under the RTF Style Template Editor Options topic.

Note:

You can transport these RTF templates between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
14.1.4.2 Select Components for Reporting

To select model components to be documented in the report, using the RTF Style Template Editor:

1. Expand the Sections tree on the <template name> screen.
2. Select the checkbox next to the component name; the component name is then displayed as a section tag in the Content panel.

The position of the section tags within the Sections tree determines the position of the model component in the Content panel. For encapsulated components, selecting a child component automatically selects the parent also.

To move a model component to a different position in the documentation template:

1. Select the component.
2. Click on and to move the component up and down the Content panel.

For example, the section tag for Package | Diagram | Element is displayed in the Content panel above the section tag for Package | Diagram | Connector.
Linked Documents and Document Artifact Contents

Linked documents and documents created via the Document Artifact element are rendered into RTF Documentation by selecting the Model Document checkbox in the RTF Style Template Editor.
The Model Document checkbox is within the Element hierarchy, towards the end. Remember that checkboxes can be moved up and down the hierarchy (as has been done above) to position information in the generated document as you require.

The linked document is rendered into the RTF documentation at:

```
<model document>
```

---

### Connections

<table>
<thead>
<tr>
<th>Connector</th>
<th>Source</th>
<th>Target</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency</td>
<td>Public</td>
<td>Public</td>
<td></td>
</tr>
<tr>
<td>Link to</td>
<td>Artifact</td>
<td>Model</td>
<td></td>
</tr>
<tr>
<td>Source -&gt; Dest</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Linked Document Editor

The text of the document displays where ‘Linked Document Editor’ appears, above.

#### 14.1.1.4.3 Add Content

The RTF Style Template Editor uses tags to arrange the content layout of the documentation. To insert a model component tag, use the Sections panel of the RTF Style Template Editor. When a selection has been made, model component tags are inserted into the Content section of the Editor. The yellow highlighted model component names specify the beginning of a tag, which is represented by `sectionname>`, and the end of the model component is shown as `<sectionname`.

To add model component content, right-click in the area between Tag Heading and Tag End. This displays a model component ‘type’-sensitive list of fields that can be added to the RTF Documentation. Any additional information entered between the document tags is included in the generated RTF Documentation.
Note:

If you select a field with short date format (such as Pkg.DateCreatedShort, Diagram.DateModifiedShort or Element.DateCreatedShort) the format is actually drawn from the MS Windows settings. To use a different short date format, click on the Start icon on the Windows desktop and select the Control Panel | Regional and Language Options | Customize option.

14.1.1.4.4 Tabular Sections

The RTF Style Template Editor supports rendering a document section as a table. 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, which you type in and format yourself. The second row defines the output, which you do by right-clicking in each cell and selecting from the field list. The output is then rendered iteratively for every occurrence of the section in question.

Example tabular section:

```
<model>
  <glossary>
    <term>Term</term> <type>Type</type> <meaning>Meaning</meaning>
  </glossary>
</model>
```

In this example the Model> Glossary> section is defined as a tabular section. This renders the following
Document output:

## Model Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting Periods</td>
<td>Business</td>
<td>A defined period of time whereby performance reports may be extracted (normally 4 week periods).</td>
</tr>
<tr>
<td>Association</td>
<td>Technical</td>
<td>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.</td>
</tr>
<tr>
<td>Class</td>
<td>Technical</td>
<td>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.</td>
</tr>
<tr>
<td>Component Model</td>
<td>Technical</td>
<td>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 interrelate with other components. Component requirements detail what responsibilities a component has to supply functionality or behavior within the system.</td>
</tr>
<tr>
<td>Customer</td>
<td>Business</td>
<td>A person or a company that requests an entity to transport goods on their behalf.</td>
</tr>
<tr>
<td>Deployment Architecture</td>
<td>Technical</td>
<td>A view of the proposed hardware that will make up the new system, together with the physical components that will execute on that hardware.</td>
</tr>
</tbody>
</table>

### 14.1.1.4.5 Child Sections

Child sections can be rendered in RTF documentation using one of the following two methods:

- Render model components directly into the RTF as defined by the section’s content and fields.
- Render indirectly to the RTF by using a parent section to describe the content.

The second option occurs as a result of creating a section that 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 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 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

<table>
<thead>
<tr>
<th>Child Section</th>
<th>Section Rendered when used as a placeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package-&gt;Child Package</td>
<td>Package</td>
</tr>
<tr>
<td>Package-&gt;Element-&gt;Child Element</td>
<td>Package-&gt;Element</td>
</tr>
<tr>
<td>Package-&gt;Element-&gt;Diagram</td>
<td>Package-&gt;Diagram</td>
</tr>
<tr>
<td>Package-&gt;Diagram-&gt;Element</td>
<td>Package-&gt;Element</td>
</tr>
<tr>
<td>Package-&gt;Diagram-&gt;Connector</td>
<td>Package-&gt;Element-&gt;Connector</td>
</tr>
</tbody>
</table>

14.1.1.4.6 RTF Style Template Editor Options

The following topics provide assistance on using the RTF Style Template Editor.

- **Scroll Through Text**
- **File and Print Options**
- **Cut and Paste Options**
- **View Options**
- **Image and Object Inserts**
- **Character Formatting**
- **Paragraph Formatting**
- **Tab Support**
- **Page Breaks and Repagination**
- **Headers and Footers**
- **Hyperlinks and Bookmarks**
- **Table Commands**
- **Sections and Columns**
- **Stylesheets and Table of Contents**
- **User-Defined Section Numbering**
- **Frame and Drawing Objects**
- **Search/Replace Commands**

**Note:**

Throughout your template editing, be aware that:

- To undo one or more immediately previous edits, press [Ctrl]+[Z], or select the Edit | Undo menu option. You can still undo a change even after you have saved the change.
- To redo one or more immediately previous undone edits, press [Ctrl]+[Y], or select the Edit | Redo menu option.

14.1.1.4.6.1 Scroll Through Text

<table>
<thead>
<tr>
<th>Scroll Using</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard Keys</td>
<td>Press</td>
</tr>
<tr>
<td></td>
<td>[1], [i], [←] or [→] to scroll up or down a line, or left or right one character</td>
</tr>
<tr>
<td></td>
<td>[Home] to move to the beginning of the current line</td>
</tr>
<tr>
<td></td>
<td>[End] to move to the end of the current line</td>
</tr>
<tr>
<td></td>
<td>[Ctrl]+[Page Up] to move to the beginning of a file</td>
</tr>
<tr>
<td></td>
<td>[Ctrl]+[Page Down] to move to the end of a file</td>
</tr>
<tr>
<td></td>
<td>[Page Up] to display the previous page</td>
</tr>
<tr>
<td></td>
<td>[Page Down] to display the next page</td>
</tr>
<tr>
<td></td>
<td>[Ctrl]+[←] to move to the previous word</td>
</tr>
<tr>
<td></td>
<td>[Ctrl]+[→] to move to the next word</td>
</tr>
</tbody>
</table>
### Scroll Using

<table>
<thead>
<tr>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>• [Ctrl]+[†] to move to the first column of the current line (if not already on the first column) or the first column of the previous line</td>
</tr>
<tr>
<td>• [Ctrl]+[¥] to move to the first column of the next line</td>
</tr>
<tr>
<td>• [F10] (or select the Other</td>
</tr>
</tbody>
</table>

### Mouse

Click on the vertical and horizontal scroll bar to perform various scrolling functions. These functions are available only if the horizontal or the vertical bar has been enabled by the startup parameters.

**Vertical Scroll Bar:** Click on the arrows on either end to scroll the screen up or down by one line.

Click above the elevator to scroll the screen up by one page. Similarly, click below the elevator to scroll the screen down by one page.

You can also drag the elevator to any position in the bar. As the elevator is dragged, the editor scrolls the screen up or down accordingly.

**Horizontal Scroll Bar:** Click on the arrows on either end to scroll the screen left or right by one column. Click on either side of the elevator to scroll the screen left or right by 1/2 screen.

You can also drag the elevator to any position in the bar. As the elevator is dragged, the editor rolls the screen left or right accordingly.

---

### 14.1.4.6.2 File and Print Options

<table>
<thead>
<tr>
<th>Menu Option &amp; Function Keys</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Clear an existing template from the edit window and start an empty, unnamed template. The editor prompts you to save any modification to the previous template.</td>
</tr>
<tr>
<td>Revert</td>
<td>Revert to the previously-saved copy of the template.</td>
</tr>
<tr>
<td>Save [Ctrl]+[S]</td>
<td>Save the text to the current file name. If a file is not yet specified, the editor prompts you for a template name.</td>
</tr>
<tr>
<td>Save As [Ctrl]+[Shift]+[S]</td>
<td>Similar to Save File, but you specify a new template name for saving the template.</td>
</tr>
<tr>
<td>Import</td>
<td>Import an existing RTF document into the Template Editor, so as to insert model elements from that document into the template. This is useful when creating templates from a predefined document with a particular 'look and feel'.</td>
</tr>
<tr>
<td>Export</td>
<td>Save the template as an RTF document rather than as a template. This can be useful for saving the template for other models.</td>
</tr>
<tr>
<td>Document Options</td>
<td>Display the Document Options dialog which enables you to set the filter and order the elements.</td>
</tr>
<tr>
<td>Page Layout</td>
<td>Specify the page layout, before selecting the Print option. You can specify the margins (left, right, top and bottom) in inches.</td>
</tr>
<tr>
<td>Printer Setup [Ctrl]+[Shift]+[P]</td>
<td>Invoke a printer-specific dialog for the default printer (the default printer selection is made from the Windows Control panel). You select the parameters from a set of printer-specific options. These options include page size, page orientation, resolution and fonts.</td>
</tr>
<tr>
<td>Print [Ctrl]+[P]</td>
<td>Print the contents of the current file. The editor displays a dialog where you can select the scope of the printing. You can also choose to print only a selected part of the file. To print a block of text, highlight the required text before invoking the Print function.</td>
</tr>
</tbody>
</table>
### Menu Option & Function Keys

<table>
<thead>
<tr>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>This command prints a highlighted:</td>
</tr>
<tr>
<td>- Line block</td>
</tr>
<tr>
<td>- Character block</td>
</tr>
<tr>
<td>The <strong>Print</strong> function prints on a default printer selected from the Windows Control panel. You can alter the printer setup or page layout prior to invoking the <strong>Print</strong> function.</td>
</tr>
</tbody>
</table>

**Print Preview**

Preview the document before printing. The editor displays up to two pages at a time. You can scroll to a different page using **[Page Up]**, **[Page Down]** 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 required.

Click on the **Edit** button or the **File | Print Preview** menu option again to return to editing mode.

**Close**

Close the Template Editor. The editor prompts you to save any unsaved information.

### 14.1.1.4.6.3 Cut and Paste Options

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Highlight a word</strong></td>
<td>Double-click on the word.</td>
</tr>
<tr>
<td><strong>Highlight a line</strong></td>
<td>Move the cursor onto the line and press <strong>[F8]</strong>.</td>
</tr>
<tr>
<td><strong>Select all file content</strong></td>
<td>Press <strong>[Ctrl]+[A]</strong> or select the **Edit</td>
</tr>
<tr>
<td><strong>Copy a block</strong></td>
<td>Highlight the lines of text to be copied and press <strong>[Ctrl]+[C]</strong>, or select the **Edit</td>
</tr>
<tr>
<td></td>
<td>Move the cursor to the point at which to insert the text and press <strong>[Ctrl]+[V]</strong> or select the **Edit</td>
</tr>
<tr>
<td><strong>Move a block</strong></td>
<td>Highlight the lines of text to be moved and press <strong>[Ctrl]+[X]</strong>, or select the **Edit</td>
</tr>
<tr>
<td></td>
<td>Move the cursor to the point at which to insert the text and press <strong>[Ctrl]+[V]</strong> or select the **Edit</td>
</tr>
<tr>
<td><strong>Delete a block</strong></td>
<td>Highlight the lines of text to be deleted and press <strong>[Delete]</strong></td>
</tr>
<tr>
<td><strong>Delete a line</strong></td>
<td>Press <strong>[Shift]+[F9]</strong> to delete the current line. The remaining lines close up.</td>
</tr>
<tr>
<td><strong>Paste special objects</strong></td>
<td>Select the **Edit</td>
</tr>
<tr>
<td></td>
<td>Click on the <strong>Paste</strong> button to embed the data into your application, or click on the <strong>Paste Link</strong> button to create a link to the original file.</td>
</tr>
<tr>
<td><strong>Native Object Format</strong></td>
<td>If available, this is the first format in the list box. You can edit data in this format using the original application, by double-clicking the object.</td>
</tr>
<tr>
<td><strong>Formatted Text</strong></td>
<td>A text format. 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.</td>
</tr>
<tr>
<td><strong>Unformatted Text</strong></td>
<td>Another text format. This option pastes the text without retaining the formatting information.</td>
</tr>
<tr>
<td><strong>Picture Format</strong></td>
<td></td>
</tr>
</tbody>
</table>

© 1998-2008 Sparx Systems Pty Ltd
To Do this...
The data is available in Picture format. You can later edit the object, by double-clicking on it and invoking the Microsoft MS Draw application.

**Note:**
This format is preferred over the Bitmap and the Device Independent Bitmap formats.

- **Device Independent Bitmap and regular Bitmap formats**
The data is available in bitmap formats. You can later edit the object, by double-clicking on it and invoking the Microsoft MS Draw application.
The editor converts these formats into the Picture format before calling the drawing application.

### 14.1.1.4.6.4 View Options

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Page Mode</strong></td>
<td>Turn <strong>Page Mode</strong> on (the equivalent of Print View in Word) or off (the equivalent of Normal View in Word). In <strong>Page Mode</strong>, the editor displays one page at a time. It is most useful for documents containing multiple columns, as the columns are displayed side by side.</td>
</tr>
<tr>
<td><strong>Fitted View</strong></td>
<td>Turn <strong>Fitted View</strong> on or off. This is a special case of <strong>Page Mode</strong>, in which the text wraps to the window width and the soft page breaks are not displayed. If you select <strong>Fitted View</strong>, <strong>Page Mode</strong> is automatically selected too. If you deselect <strong>Page Mode</strong>, <strong>Fitted View</strong> is automatically deselected too.</td>
</tr>
<tr>
<td><strong>Ruler</strong></td>
<td>Display or hide the ruler at the top of the page. The ruler shows tab stops and paragraph indentation marks; it can also be used to create or delete tab stops.</td>
</tr>
<tr>
<td><strong>Tool Bar</strong></td>
<td>Display or hide the tool bar above the ruler. 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.</td>
</tr>
<tr>
<td><strong>Status Ribbon</strong></td>
<td>Display or hide the status ribbon at the bottom of the editing panel. The status ribbon displays the current page number, line number, column number and row number. It also indicates the current insert/overtype mode.</td>
</tr>
<tr>
<td><strong>Paragraph Marker</strong></td>
<td>Display or hide the paragraph marker (an inverted ‘P’) at the end of each paragraph. This option is useful when working with lines with many different heights.</td>
</tr>
<tr>
<td><strong>Hidden Text</strong></td>
<td>Show or hide ‘hidden’ text. Text formatted with the hidden attribute (see Character Formatting Options) is shown with a dotted underline. When the option is turned off, the hidden text is not visible.</td>
</tr>
<tr>
<td><strong>Field Names</strong></td>
<td>Insert, show and hide field names. As you develop your RTF document, you right-click on sections to insert field markers. You cannot insert these markers unless you have selected the <strong>Field Names</strong> option. When you deselect the option, existing field names are obscured.</td>
</tr>
<tr>
<td><strong>Page Header/ Footer</strong></td>
<td>Display or hide the text of page headers and footers. If <strong>Page Mode</strong> is not selected, this option turns <strong>Page Mode</strong> on. When <strong>Page Mode</strong> is selected, you cannot edit the header or footer text unless you also select the **Edit</td>
</tr>
</tbody>
</table>
### Menu Option

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Border</td>
<td>Display or hide a page outline in Page Mode. When Page Mode is deselected, this option is not available. When Page Border is selected, the document contents are shown within a page outline. When Page Border is deselected, the document contents are formatted within the boundaries of the editing screen.</td>
</tr>
<tr>
<td>Zoom</td>
<td>Shrink or enlarge the display of the document text, by selecting the appropriate percentage enlargement. The editor supports zoom percentages from 25 to 200.</td>
</tr>
</tbody>
</table>

### 14.1.1.4.6.5 Image and Object Inserts

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
</table>
| Embed a picture in the document | Position the cursor at the point at which to insert the picture bitmap or Windows metafile, and either:  
- Select the Insert | Embed Picture menu option, or  
- Press [Alt]+[F8].  
A browser dialog displays, through which you select the picture to embed in the document. The picture displays at the current cursor location. The embedded picture is saved within the document. |
| Link a picture file to the document | Position the cursor at the point at which to link the picture bitmap or Windows metafile, and select the Insert | Link Picture menu option. A browser dialog displays, through which you select the picture to link to the document. The picture displays at the current cursor location. Linked picture data is not saved with the document, only the filename is stored within the document. |
| Embed an Ole object in the text | Position the cursor at the point at which to embed the object, and select the Insert | Ole Object menu option. The Insert Object dialog displays, listing the applications that are available to create the object. When you select an application, the editor launches it and you create the required object using this application. When you save the application, the editor inserts an icon that indicates the inserted object. You can later edit the object using the application, by double-clicking on the object.  
**Note:** You can also use the Edit | Paste Special menu option to import the OLE objects, provided that the object is available in the clipboard. |
| Edit an embedded picture    | Click on the picture and select the Edit | Edit Picture menu option. The Edit Current Picture Parameters dialog displays, through which you can change the width and height of the picture, in inches. You can also align the top, bottom, or middle of the picture with the base line of the text. |
| Edit an embedded Ole object | Double-click on the icon that indicates the inserted Ole object. Alternatively, position the cursor on the icon and select the Edit | Edit OLe Object menu option. The editor opens the object in the application used to create it, and you can edit the object. |
| Insert a background picture for the text | Select the Other | Background Picture menu option. A browser dialog displays, through which you select the bitmap or metafile file to insert as a background picture. The picture occupies the entire text area. To remove the background picture, deselect the Background Picture menu option. |
### 14.1.4.6.6 Character Formatting

When you change the format of existing text, any new characters you type immediately following automatically assume the formatting characteristics of the existing text.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
</table>
| **Apply character format** | Highlight the text to which to apply the format, and use one or more of the following menu options or key combinations, as required:  
- **Font | Normal**, or press `[Alt]-[0]`  
- **Font | Bold**, or press `[Ctrl]+[B]`  
- **Font | Underline**, or press `[Ctrl]+[U]`  
- **Font | Double Underline**, or press `[Ctrl]+[D]`  
- **Font | Italic**, or press `[Ctrl]+[I]`  
- **Font | Superscript**, or press `[Alt]+[4]`  
- **Font | Subscript**, or press `[Alt]+[5]`  
- **Font | Strike**, or press `[Alt]+[6]` (puts a line through the text)  
- **Font | All Caps**  
- **Font | Small Caps** |

To reset any character format, highlight the text and select the **Font | Normal** menu option, or press `[Alt]-[0]`. |

| **Change font typeface and point size** | Highlight the text to change and select the **Font | Fonts** menu option, or press `[Alt]+[F10]`. The **Font Selection** dialog displays, from which you select the required typeface and point size. Click on the **OK** button. |

| **Change character style** | Highlight the text to change and select the **Font | Style** menu option. The **Select a Style** dialog displays, listing the currently-defined character styles in the template stylesheet. Select the required style and click on the **OK** button. |

| **Change the color of text, background (permanent highlight) or underline** | Highlight the text to change and select one or more of the following options, as required:  
- **Font | Text Color**  
- **Font | Background Color**  
- **Font | Underline Color** |

In each case, the **Color** dialog displays, through which you can select or define the required color. When you have selected a color, click on the **OK** button. |

| **Change character spacing** | Normal character spacing is 20 Twips. If you want to change this (or return to it), highlight the text to adjust, and select the **Font | Spacing** menu option. The **Character Spacing** dialog displays. Select the radio button to expand or compress spacing, or to return to normal spacing. If you are changing from normal spacing, enter the number of Twips to set the spacing to. Click on the **OK** button. |

| **Hide text** | Hidden text is not displayed on the screen or printer, but remains in the document and is not deleted.  
Highlight the text to hide and select the **Font | Hidden** menu option, or press `[Ctrl]+[H]`. The highlighted text is not displayed and the rest of the text closes up.  
To view hidden text, select the **View | Hidden Text** menu option. You can then make the text normal again by highlighting it and deselecting the **Font | Hidden** menu option. |

| **Box text** | Highlight the text to box and select the **Font | Boxed** menu option. This creates a broken-line border around the selected text. |

| **Insert a non-breaking space** | Move the cursor to the point at which to insert the non-breaking space and select the **Insert | Non-breaking Space** menu option. |

| **Insert a non-breaking dash** | Move the cursor to the point at which to insert the non-breaking dash and select the **Insert | Non-breaking Dash** menu option. |

| **Insert an optional hyphen** | Move the cursor to the point at which to insert the hyphen and select the **Insert | Optional Hyphen** menu option. |
### 14.1.4.6.7 Paragraph Formatting

The functions described below operate on the current paragraph, or on a highlighted block of text.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clear all paragraph formatting</strong></td>
<td>Select the Paragraph</td>
</tr>
</tbody>
</table>
| **Set text flow in document** | To set the text flow for:  
- A selected block of text, select the Paragraph | Text Flow menu option; the Paragraph Text Flow dialog displays  
In either case, select the required text flow direction and click on the OK button. |
| **Center text** | Select the Paragraph | Center menu option or press [Alt]+[8]. |
| **Right-justify text** | Select the Paragraph | Right Justify menu option or press [Alt]+[9]. |
| **Justify both sides of text** | Select the Paragraph | Justify Both menu option. |
| **Set double line spacing** | Select the Paragraph | Double Space menu option. A double-spaced paragraph has a blank line between each text line. |
| **Indent paragraph left** | Select the Paragraph | Indent Left menu option or press [Alt]+[L]. Select the option again to increase the indent. |
| **Indent paragraph right** | Select the Paragraph | Indent Right menu option or press [Alt]+[R]. Select the option again to increase the indent. |
| **Create hanging indent** | Select the Paragraph | Hanging Indent menu option or press [Alt]+[T]. Select the option again to increase the indent of all lines below the first. |
| **Keep paragraph lines together** | Select the Paragraph | Keep Together menu option. The editor attempts to keep all lines within the paragraph on the same page. |
| **Keep paragraph with next** | Select the Paragraph | Keep with Next menu option. The editor attempts to keep the last line of the current paragraph and the first line of the next paragraph on the same page. |
| **Prevent 'widow' and 'orphan' lines** | Select the Paragraph | Widow/Orphan Control menu option. The editor attempts to avoid having:  
- the first line of the paragraph on the previous page ('widow' line)  
- the last line of the paragraph on the next page ('orphan' line). |
| **Start text on new page** | Move the cursor to the point at which to start the new page, and select the Paragraph | Page Break Before menu option. |
| **Insert border and shading for text block** | Highlight the required text and select the Paragraph | Border and Shading menu option. The Paragraph Box Parameters dialog displays, on which you specify which sides of the box to display (including a line between text lines), whether the lines are thick or doubled, the degree of gray shading behind the text, and the color of the lines. |
| **Define line spacing** | Highlight the required lines and select the Paragraph | Paragraph Spacing menu option. The Paragraph Spacing Parameters dialog displays, on which you specify the line spacing and the point spacing before and after lines. |
| **Set a background color for text space** | Highlight a text string or block of text and select the Paragraph | Background Color menu option. The Color dialog displays, on which you select the background color. The editor highlights the full width of the page in that color, for the selected lines. |
| **Create a bulleted list** | Highlight the required lines of text and select the Paragraph | Bullet menu option. The editor formats the lines into a simple bullet list. |
| **Create a numbered list** | Highlight the required lines of text and select the Paragraph | Numbering menu option. The editor formats the lines into a simple numbered list. |
14.1.1.4.6.8 Tab Support

The RTF Style Template Editor supports left, right, center and decimal tabs. Tabs are very useful for creating columns and tables. A paragraph can have as many as twenty tab positions.

A tab usually applies to every line of the current paragraph. However, if you highlight a block of text before setting a tab, the tab then applies to every line in the highlighted text.

You can create tabs quickly and easily using the ruler at the top of the screen. To create a:

- **Left** tab, click on the required tab position on the ruler; the left tab is indicated on the ruler by an L shape.
- **Right** tab, right-click on the required tab position on the ruler; the right tab is indicated on the ruler by a reversed L shape.
- **Center** tab, press [Shift] and click on the required tab position on the ruler; the center tab is indicated on the ruler by an inverted T shape.
- **Decimal** tab, press [Shift] and right-click on the required tab position on the ruler; the decimal tab stop is indicated on the ruler by an inverted T shape with a dot on the right hand side. This tab is for numbers with a decimal point; numbers scroll left from the tab until you type a point, then numbers scroll right.

You can also set tabs using the Paragraph | Set Tab menu option, which displays the Set a Tab Position dialog. This enables you to specify the tab type, and provides two advantages over the ruler: you can set the tab position with more precision and with a clear value that you can duplicate; and you can add a tab leader line (dot, hyphen, or underline).

- To clear a single tab position for selected text, select the Paragraph | Clear Tab menu option. The Clear a Tab Position dialog displays, on which you select the tab to clear.
- To clear all tab stops for selected text, select the Paragraph | Clear All Tabs menu option.
- To move a tab position using the mouse, click on the tab symbol on the ruler and drag it to the new position.

**Note:**

The Other | Snap To Grid menu option affects the movement of the tabs (and the paragraph indentation markers) on the ruler. When you select this option, the movements of the tab markers are locked on to an invisible grid at intervals of 1/16 inch (half a ruler division).

14.1.1.4.6.9 Page Breaks and Repagination

You can force a page break in the document by selecting the Insert | Insert Break | Page Break menu option, or by pressing [Ctrl]+[Enter]. The forced page break is indicated by a solid line in the editing window.

If Page Mode is off, the editor also marks automatic page breaks when the text overflows a page; these are indicated by a dotted line.

You can repaginate your document, using the Edit | Repaginate menu option. This updates the Page Number and Page Count fields, and recompiles the table of contents.

You insert the Page Number and Page Count fields as follows:

To

**To**

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply a paragraph style from the template stylesheet</td>
<td>Select the Paragraph</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tab Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert the page number</td>
<td>Position the cursor at the point at which to display the page number, and select the Insert</td>
</tr>
<tr>
<td>Insert the page count</td>
<td>Position the cursor at the point at which to display the total number of pages in the document, and select the Insert</td>
</tr>
</tbody>
</table>
### 14.1.1.4.6.10 Headers and Footers

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
</table>
| Edit the page header and footer text | In Page Mode, select the **Edit | Edit Page Header/Footer** menu option. A paragraph marker displays at the top and bottom of each page, and you can type in, format or delete the appropriate text.  
If Page Mode is turned off, all page headers and footers are displayed in a block at the start of the document, with identifying labels. You can also edit the text here.  
Each section in a document can have its own page header and footer. |
| Create the header and footer for the initial page of the document | In Page Mode, select the **Edit | Edit Page Header/Footer** menu option and then select the **Edit | First Page Header/Footer | Create First Page Header** or **Create First Page Footer** menu option. A paragraph marker displays at the top or bottom of the first page, and you can type in and format the appropriate text.  
If Page Mode is turned off, all page headers and footers are listed in a block at the start of the document, with identifying labels. You can also edit the text here. |
| Delete the header and footer for the initial page of the document | Whilst you can delete the first page header or footer text by simple editing, you must specifically delete the ‘first page’ assignment in order to display the header and footer text of the next section on the first page.  
In Page Mode, select the **Edit | Edit Page Header/Footer** menu option and then select the **Edit | First Page Header/Footer | Delete First Page Header** or **Delete First Page Footer** menu option. This removes the first page text and assignment, and displays the next-defined header and footer text. |
| Create a footnote | Move the cursor to the position at which to insert the footnote marker, and select the **Insert | Footnote/Endnote | Footnote** menu option. The Footnote Parameters dialog displays, on which you enter the footnote marker and footnote text, and select whether to make the marker a superscript.  
Click on the **OK** button. The editor inserts the footnote marker at the current cursor location and, in Page Mode, displays the footnote text at the bottom of the page. |
| Edit footnote text | Select the **Edit | Edit Footnote/Endnote | Edit Footnote Text** menu option. The text of each footnote displays in the document text where its marker was inserted. Locate the text and make the required changes. In Page Mode, the modified footnote displays at the bottom of the page.  
When you have finished editing footnote text, **deselect** the **Edit | Edit Footnote/Endnote | Edit Footnote Text** menu option. The footnote text is no longer shown in the document text. |
| Create an endnote | Move the cursor to the position at which to insert the endnote marker, and select the **Insert | Footnote/Endnote | Endnote** menu option. The Endnote Parameters dialog displays, on which you enter the endnote marker and endnote text, and select whether to make the marker a superscript.  
Click on the **OK** button. The editor inserts the endnote marker at the current cursor location and, in Page Mode, displays the endnote text at the end of the section or document. |
| Edit endnote text | Select the **Edit | Edit Footnote/Endnote | Edit Endnote Text** menu option. The text of each endnote displays in the document text where its marker was inserted. Locate the text and make the required changes. In Page Mode, the modified endnote displays at the bottom of the page.  
When you have finished editing endnote text, **deselect** the **Edit | Edit Footnote/Endnote | Edit Endnote Text** menu option. The endnote text is no longer shown in the document text. |
| Manage bookmarks | Each template contains a number of bookmarks that mark the sections. You can apply these bookmarks to related sections, create and assign your own, delete those that are not required, and locate specific bookmarks in the document.  
In **Page Mode**, select the **Insert | Bookmark** menu option. The **Bookmark** dialog displays.  
- To assign a bookmark to the current cursor position, either type a new bookmark... |
### Manage bookmarks

Each template contains a number of bookmarks that mark the sections. You can apply these bookmarks to related sections, create and assign your own, delete those that are not required, and locate specific bookmarks in the document.

In **Page Mode**, select the **Insert | Bookmark** menu option. The **Bookmark** dialog displays.

- To assign a bookmark to the current cursor position, either type a new bookmark in the top field or select one from the list, and click on the **Insert** button.
- To delete an existing bookmark, click on it in the list and click on the **Delete** button.
- To mark a bookmark with the number of the page it is on, click on the bookmark in the list and click on the **Set Page Reference** button.
- To locate a bookmark in the text, click on it in the list and click on the **Go to** button.

### Insert hyperlinks

You can create a hyperlink within an RTF document to an external document, Help topic or web page.

To insert a hyperlink, right-click on the point at which to create the hyperlink and select the **Insert | Hyperlink** menu option. The **Insert Hyperlink** dialog displays.

In the **Link Text** field type the text to be hyperlinked, and in the **Link Code** field type or paste the web page URL, help topic file or external file path and name.

**Tip:** To capture the help topic file name, right-click on the displayed topic, select the **Properties** menu option, and copy the file name. When you insert the file name in the **Link Code** field, ensure that the file name has the prefix `$Help://`.

Click on the **OK** button. The hyperlinked text displays in the document. Double-click on the link to display the web page or external document.

---

### 14.1.1.4.6.11 Hyperlinks and Bookmarks

**Manage bookmarks**

Each template contains a number of bookmarks that mark the sections. You can apply these bookmarks to related sections, create and assign your own, delete those that are not required, and locate specific bookmarks in the document.

In **Page Mode**, select the **Insert | Bookmark** menu option. The **Bookmark** dialog displays.

- To assign a bookmark to the current cursor position, either type a new bookmark in the top field or select one from the list, and click on the **Insert** button.
- To delete an existing bookmark, click on it in the list and click on the **Delete** button.
- To mark a bookmark with the number of the page it is on, click on the bookmark in the list and click on the **Set Page Reference** button.
- To locate a bookmark in the text, click on it in the list and click on the **Go to** button.

**Insert hyperlinks**

You can create a hyperlink within an RTF document to an external document, Help topic or web page.

To insert a hyperlink, right-click on the point at which to create the hyperlink and select the **Insert | Hyperlink** menu option. The **Insert Hyperlink** dialog displays.

In the **Link Text** field type the text to be hyperlinked, and in the **Link Code** field type or paste the web page URL, help topic file or external file path and name.

**Tip:** To capture the help topic file name, right-click on the displayed topic, select the **Properties** menu option, and copy the file name. When you insert the file name in the **Link Code** field, ensure that the file name has the prefix `$Help://`.

Click on the **OK** button. The hyperlinked text displays in the document. Double-click on the link to display the web page or external document.
### 14.1.1.4.6.12 Table Commands

The Table menu enables you to create a new table, or to edit an existing table's attributes.

<table>
<thead>
<tr>
<th>To</th>
<th>Do This...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert a table in the document</strong></td>
<td>Position the cursor at the appropriate point, and select the Table</td>
</tr>
<tr>
<td><strong>Insert a new row above the current row</strong></td>
<td>Select the Table</td>
</tr>
<tr>
<td><strong>Insert a new column to the left of the current column</strong></td>
<td>Select the Table</td>
</tr>
<tr>
<td><strong>Merge cells</strong></td>
<td>Select the cells to merge and select the Table</td>
</tr>
<tr>
<td><strong>Split a cell</strong></td>
<td>Select the cell to split and select the Table</td>
</tr>
<tr>
<td><strong>Delete cells</strong></td>
<td>Select the cells to delete and select the Table</td>
</tr>
<tr>
<td></td>
<td>• Cells - deletes the highlighted cells</td>
</tr>
<tr>
<td></td>
<td>• Columns - deletes all the cells in the highlighted column or columns</td>
</tr>
<tr>
<td></td>
<td>• Rows - deletes all the cells in the highlighted row or rows.</td>
</tr>
<tr>
<td></td>
<td>If you delete all cells in a table, the table itself is automatically deleted.</td>
</tr>
<tr>
<td><strong>Position the table on the page</strong></td>
<td>Click on any part of the table and select the Table</td>
</tr>
<tr>
<td><strong>Set the height of a row, or all rows</strong></td>
<td>Select the row to adjust and select the Table</td>
</tr>
<tr>
<td><strong>Keep row text together if it continues over a page</strong></td>
<td>Select the rows to protect (preferably all rows in the table) and select the Table</td>
</tr>
<tr>
<td><strong>Set text flow in rows</strong></td>
<td>Select the rows and select the Table</td>
</tr>
<tr>
<td><strong>Set the width of selected cells</strong></td>
<td>Select the cells to act on and select the Table</td>
</tr>
<tr>
<td></td>
<td>• All cells in a highlighted block</td>
</tr>
<tr>
<td></td>
<td>• The selected cells only</td>
</tr>
<tr>
<td></td>
<td>• All cells in the selected column or columns, or</td>
</tr>
<tr>
<td></td>
<td>• All cells in the selected row or rows.</td>
</tr>
</tbody>
</table>
| **Define the cell** | Select the cells to act on and select the Table | Cell Border Width menu option. The Set...
<table>
<thead>
<tr>
<th>To</th>
<th>Do This...</th>
</tr>
</thead>
</table>
| border width        | **Cell Border** dialog displays, on which you set the width of the line at any or all of the top, bottom, left and right of a cell, or whether to draw a uniform border around the cells. You can also set the text margin, and apply all the settings to:  
  - All cells in a highlighted block  
  - The selected cells only  
  - All cells in the selected column or columns, or  
  - All cells in the selected row or rows. |
| Define the cell border color | Select the cells to act on and select the **Table | Cell Border Color** menu option. The **Set Cell Border Color** dialog displays, on which you set the color of the line at any or all of the top, bottom, left and right of a cell, or whether to have a uniformly colored border around the cells. You then apply the settings to:  
  - All cells in a highlighted block  
  - The selected cells only  
  - All cells in the selected column or columns, or  
  - All cells in the selected row or rows. |
| Define the cell shading | Select the cells to act on and select the **Table | Cell Shading** menu option. The **Cell Shading Parameters** dialog displays, on which you set the shading percentage. The value 0 indicates the palest background, whereas the value 100 indicates a black background. You then apply the setting to:  
  - All cells in a highlighted block  
  - The selected cells only  
  - All cells in the selected column or columns, or  
  - All cells in the selected row or rows. |
| Define the cell background color | Select the cells to act on and select the **Table | Cell Color** menu option. The **Cell Color Parameters** dialog displays, on which you set the cell background color. You then apply the color to:  
  - All cells in a highlighted block  
  - The selected cells only  
  - All cells in the selected column or columns, or  
  - All cells in the selected row or rows. |
| Vertically align cells | Select the cells to act on and select the **Table | Cell Vertical Align** menu option. The **Cell Vertical Alignment** dialog displays, on which you select to align the selected cells by top, center, bottom or baseline. You then select to align:  
  - All cells in a highlighted block  
  - The selected cells only  
  - All cells in the selected column or columns, or  
  - All cells in the selected row or rows. |
| Rotate cell text     | Select the cells to act on and select the **Table | Cell Rotate Text** menu option. The **Cell Text Rotation** dialog displays. On which you select to display text horizontally across the cell, vertically up the cell, or vertically down the cell. You then select to apply the rotation to:  
  - All cells in a highlighted block  
  - The selected cells only  
  - All cells in the selected column or columns, or  
  - All cells in the selected row or rows. |
| Select column        | Click on a cell and select the **Table | Select Current Column** menu option. The whole column is highlighted and selected for further formatting. |
| Show / hide table outline | Click on a table cell and select the **Table | Show Gridlines** menu option. This displays or hides the grid lines around the table cells. The grid lines are for display purpose only and do not appear on the printed document. |
14.1.1.4.6.13 Sections and Columns

The editor enables you to divide a document into multiple sections. A multiple section document is useful to:

- Vary the page margins from one page to another
- Create multiple columns of text.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new section</td>
<td>Select the Insert</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This option is not available when Edit</td>
</tr>
<tr>
<td>Edit the section parameters</td>
<td>Select the Edit</td>
</tr>
<tr>
<td></td>
<td>- The number of columns and column spacing; text in a multiple column section wraps at the end of the column. When the text reaches the end of the page, it resumes in the next column. When Page Mode is off, the page contains a single very thin and long column. When Page Mode is on, the correct column layout is shown.</td>
</tr>
<tr>
<td></td>
<td>- The orientation - Portrait or Landscape</td>
</tr>
<tr>
<td></td>
<td>- Whether to start the new section on the next page</td>
</tr>
<tr>
<td></td>
<td>- The direction of text flow</td>
</tr>
<tr>
<td></td>
<td>- Any special printing characteristics for the section.</td>
</tr>
<tr>
<td></td>
<td>You can also define any special page margins by selecting the File</td>
</tr>
<tr>
<td>Delete a section break</td>
<td>Move the cursor onto the section break line and press [Delete].</td>
</tr>
<tr>
<td>Create a column break</td>
<td>Move the cursor to the appropriate point in the text and select the Insert</td>
</tr>
</tbody>
</table>

14.1.1.4.6.14 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 both a set of character formatting attributes and a set of paragraph formatting attributes, and is applied to one or more paragraphs.

You can also include special, structured text in the document, such as page number, date and time and text input fields.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and edit styles</td>
<td>Select the Edit</td>
</tr>
</tbody>
</table>
To | Do this...
--- | ---
document automatically reflects the updated style. If you created a new stylesheet item, you can apply the style to highlighted text by selecting the Font | Style or Paragraph | Style menu options.

Apply character styles | Select the Font | Style menu option to apply a stylesheet style to a highlighted character string.

Apply paragraph styles | Select the Paragraph | Style menu option to apply a stylesheet style to a highlighted paragraph or range of paragraphs.

Insert a table of contents | Create and apply the required heading styles using the Edit | Edit Style menu option, as above. Move the cursor to the point at which to insert the table of contents and select the Insert | Table of Contents menu option. The table of contents is automatically updated whenever repagination occurs.

Insert date and time fields | Move the cursor to the point at which to insert the current date and time, and select the Insert | Date and Time menu option. The Insert Current Date/Time dialog displays, from which you select the required date and time format. The date and time are automatically updated whenever the page text is refreshed.

Insert your own data fields | Move the cursor to the point at which to insert the data field and select the Insert | Data Field menu option. The Data Field Parameters dialog displays, in which you enter the field name and data value.

Insert a text entry field | Move the cursor to the point at which to insert the text entry field and select the Insert | Text Input Field menu option. The Input Field Parameters dialog displays, in which you enter the field name, initial value, maximum field length and text font, and specify whether or not the field has a border.

Insert a selectable checkbox | Move the cursor to the point at which to insert the checkbox and select the Insert | Checkbox menu option. The Checkbox Field Parameters dialog displays, in which you enter the field name, whether it defaults to selected, and the size of the box surrounding the check.

Define level numbering in generated document | Select the Edit | List and Overrides menu option, set up the numbering list and the list overrides, then apply the numbering list to the headings set for packages and elements, using Paragraph Numbering.

### 14.1.1.4.6.15 User-Defined Section Numbering

You might want to define the numbering format for the section levels in your generated RTF document.

For example:

1. Package level 1
   1.1 Package level 2 (child package)
   1.1.1 Element Level 1
   1.1.1.1 Element (child element)

To define the numbering format you first create a numbering list and then create a set of list overrides for this list. The overrides must also have the initial 1.0.0 setting altered to 1.1.1. You can then apply the numbering list to the headings set for packages and elements, using paragraph numbering.

**Procedure**

To define the numbering format, follow the steps below.

1. In the Template Editor, select the Edit | List and Overrides | Create Item menu option. The List Properties dialog displays.
2. In the **List Name** field, type a name for the list. Click on the **OK** button to close the dialog.

3. Select the **Edit | List and Overrides | Create List Override** menu option. The **List Override Properties** dialog displays.

4. In the **List to Override** field, type or select the name of the list you have just created. Click on the **OK** button to close the dialog.

5. To set the list level properties for each level, select the **Edit | List and Overrides | Edit List Level** menu option. The **List Level Properties** dialog displays.

6. To set the first level numbering (used in the **Package Section**), select the **List override** radio button and type or select the list override item you have just created.

7. Ensure that the **List Level** field is set to 1 (for Packages) and the **Number text** field is set to ~1~. Click on the **OK** button to save the values and close the dialog.

8. Open the dialog again (**Edit | List and Overrides | Edit List Level**) and set:
Use Numbering Levels

To apply the numbering levels, follow the steps below:

1. In the Content window of the Template Editor, select the first item of text to be numbered (for example, Package).

2. Set the text style, using the style drop-down field in the Template Editor toolbar.

3. Click on the Paragraph | List Numbering menu option. The Apply paragraph numbering using Lists dialog displays.

4. Select the required Numbering List and Override, and set the Level field to the required level (1, for the top level). Click on the OK button to close the dialog, and check that the required level has been applied to the selected text.

5. Repeat steps 1 to 4 for the next level (Element), but at step 4 change the Level field to 2.
Continue applying the overrides for lower section levels as necessary, then save the template and generate your RTF report. The output should now resemble the following example:

1. **Package: Formal Requirements**
   1.1 **Package: Manage Users**
      1.1.1 Element: REQ011 - Manage User Accounts
      1.1.2 Element: REQ016 - Add Users
      1.1.3 Element: REQ017 - Remove User
      1.1.4 Element: REQ018 - Report on User Account
      1.1.5 Element: REQ024 - Secure Access
      1.1.6 Element: REQ025 - Store User Details
      1.1.7 Element: REQ026 - Validate User
   1.2 **Package: Manage Inventory**
      1.2.1 Element: REQ019 - Manage Inventory
      1.2.2 Element: REQ020 - Receive Books
      1.2.3 Element: REQ021 - List Stock Levels
      1.2.4 Element: REQ022 - Order Books
      1.2.5 Element: REQ023 - Store and Manage Books
      1.2.6 Element: REQ027 - Add Books
      1.2.7 Element: REQ032 - Update Inventory

14.1.4.6.16 **Frames and Drawing Objects**

A frame is a rectangular area that can contain both text and pictures on the page. The text outside the frame flows around it. A drawing object can be a text box, rectangle or a line. The drawing object overlays the text around it. You can see frames and drawing objects only if View | Page Mode is selected. However, the content of a frame or text box is still visible if Page Mode is deselected.

To:

- Embed a frame or drawing object at the current cursor position, select the Insert | Frame or Insert | Drawing Object menu option.
- Insert text into the frame or drawing object text box, click inside the outline and type the text at the cursor position.
- Rotate the text to display it down the side of the frame or text box, select the Edit | Edit Frame/Drawing Object | Rotate Text menu option and select the text direction.
- Insert a picture into a frame or drawing object text box, copy the picture, click inside the outline and paste the picture at the cursor position.
- Size a frame or drawing object, click inside the outline, click on a sizing tab on the outline and drag the tab to the required position. If the frame or text box contains only a picture, the picture size is automatically adjusted to fill the outline. Any text inside the outline is automatically wrapped to adjust to the new width. In a frame, if necessary, the frame height is automatically adjusted to enclose all lines. In a text box, the height must be adjusted manually to enclose the text.
- Move the frame or drawing object, click inside the outline and then move the cursor just outside the outline until the cursor changes to a plus-shape. Drag the plus shape (and hence the outline) to the new location.
- Edit the relationship between a frame or drawing object and a point on the page (the vertical base position), click on the outline and select the Edit | Edit Frame/Drawing Object | Vertical Base Position menu option. Select the point to lock the outline to. Outlines locked to the top of the page or the top of the margin retain their vertical position when you insert text before them.
- Edit the border and the background of a drawing object, select the Edit | Edit Frame/Drawing Object | Edit Drawing Object menu option. On the Line and fill attributes dialog, select the options for the preferred border, line color, fill color and wrapping effect on the template text.
- Delete a frame or drawing object, click on the outline and press [Delete]. The editor prompts you to confirm the deletion. Click on the OK button. Note that the deletion is actually reversible - press [Ctrl]+[Z] or select the Edit | Undo menu option.
14.1.1.4.6.17 Search and Replace Commands

The first three menu options below all invoke the Search String Parameters dialog, if no search term has been defined.

Specify the term to search for, whether to search from the start of the file or forwards or backwards from the current cursor position, and whether the search should exactly match the case of the search term.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for a text string</td>
<td>Select the Other</td>
</tr>
<tr>
<td>Find the next instance of a previously-defined text string in the file</td>
<td>Select the Other</td>
</tr>
<tr>
<td>Find the previous instance of the previously-defined text string in the file</td>
<td>Select the Other</td>
</tr>
<tr>
<td>Replace a text string</td>
<td>Select the Other</td>
</tr>
</tbody>
</table>

14.1.1.5 Import RTF Template

Enterprise Architect provides a number of RTF document templates, and enables you to create others. However, you might already have corporate formats and templates in use in your organization, so Enterprise Architect also enables you to import existing templates into the RTF Generator.

To import an external template, follow the steps below:

1. Save the external template as an RTF document.
2. In Enterprise Architect, create a new blank template. Name the template but do not specify an existing template to copy from.
3. When the template is listed on the Templates tab of the Generate RTF Documentation dialog, click on the name and click on the Edit button. The RTF Document Template Editor dialog displays.
4. Select the File | Import menu option. The Microsoft Word file Open dialog displays.
5. Locate your template RTF file, and click on the Open button. The Open dialog closes, returning you to the Document Template Editor dialog. This now contains your imported template.
6. Select the File | Save menu option. If necessary, make any changes to the template and select File | Save again before selecting File | Close to exit the dialog.
Note:
Standard graphical images (such as a logo in the header, main text or footer) are imported. However, any Word-based meta-file graphics are not imported.

You can select the new template to use in generating an RTF document, either on the Generate RTF Documentation dialog or in a Master Document or Model Document element.

14.1.1.6 Resource Documents

The Resource Document feature enables a particular documentation configuration to be 'remembered', linking the loaded template within the Generate RTF Documentation dialog to the current highlighted package. If a particular template is always used with a specific package, and multiple cases of documentation exist to be propagated, saving these as Resource Documents can ease document generation later.

To create and use Resource Documents, follow the steps below:

1. Open the Generate RTF Document dialog.
2. Click on the Resource Document button. The Save current as document definition dialog displays:

   Enter Value: 
   
   OK  Cancel

3. In the Enter Value field, type a name for the document and click on the OK button. The document is added to the Resources window for easy future access (as for the usecasedoc entry in the illustration below).

4. To generate documentation from the Resources window, right-click on the required document. The context menu displays.
5. Select the required option.

The context menu options are:

- **Open Document** - Opens the corresponding .RTF file, as specified by the RTF template Filename property
- **Generate Document** - Opens the Generate RTF Documentation dialog, loaded with the specified template
- **Auto Generate Document** - Generates documentation, with the document located at the path specified by the template's Filename property
- **Delete Document** - Removes the specified document.

Batch Generate Resource Documents

You can generate a number of RTF documents at the same time, by right-clicking on the RTF Documents folder name. Select the Generate Documents context menu option to display the Batch Document Generation dialog.
The dialog lists all resource documents in the RTF Documents folder. Deselect the checkbox against each document that you do not want to generate, then click on the OK button to generate each of the remaining reports into their respective target file locations.

The Generate All Documents option automatically generates every document in the RTF Documents folder, without displaying the Batch Document Generation dialog.

### 14.1.1.7 Document Options

The RTF documentation options enable you to set the filter and order the elements. You can access the options from two different places; the start point affects the number and persistence of options selected:

1. If you access the options on the Options and Advanced tabs of the Generate RTF Documentation dialog, you can create filtering settings for the current document set to be run. Selections are non-persistent, and are reset when you select a different template.

2. If you access the options by clicking on the File | Document Options menu option on the RTF Style Template Editor dialog, the settings are saved with the template as the default settings for any run of this report; the Document Options dialog provides the options from both of the Options and Advanced tabs of the Generate RTF Documentation dialog.

The Options tab of the Generate RTF Documentation dialog has the following fields:

<table>
<thead>
<tr>
<th>Document</th>
<th>Template</th>
<th>Target File</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHPdoc</td>
<td>(basic template) C:\Documents and Settings\rchester\My Doc...</td>
<td></td>
</tr>
<tr>
<td>Shopcart</td>
<td>(interaction ... C:\Documents and Settings\rchester\My Doc...</td>
<td></td>
</tr>
<tr>
<td>usecase.doc</td>
<td>Component ...</td>
<td></td>
</tr>
</tbody>
</table>

The dialog lists all resource documents in the RTF Documents folder. Deselect the checkbox against each document that you do not want to generate, then click on the OK button to generate each of the remaining reports into their respective target file locations.

The Generate All Documents option automatically generates every document in the RTF Documents folder, without displaying the Batch Document Generation dialog.

### 14.1.1.7 Document Options

The RTF documentation options enable you to set the filter and order the elements. You can access the options from two different places; the start point affects the number and persistence of options selected:

1. If you access the options on the Options and Advanced tabs of the Generate RTF Documentation dialog, you can create filtering settings for the current document set to be run. Selections are non-persistent, and are reset when you select a different template.

2. If you access the options by clicking on the File | Document Options menu option on the RTF Style Template Editor dialog, the settings are saved with the template as the default settings for any run of this report; the Document Options dialog provides the options from both of the Options and Advanced tabs of the Generate RTF Documentation dialog.

The Options tab of the Generate RTF Documentation dialog has the following fields:
<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Filter</strong></td>
<td></td>
</tr>
<tr>
<td>Only include objects</td>
<td>Filter elements according to date created or modified. In the first two fields, select the qualifiers from the drop-down lists. In the third field, select the appropriate date.</td>
</tr>
<tr>
<td>Where Package Phase</td>
<td>Filter elements according to the value of the Package Phase field. In the first field select the qualifier, and in the second type the required phase (or leave the default value All).</td>
</tr>
<tr>
<td>With element status</td>
<td>Filter elements according to status. In the first field, select the qualifier (like, not like, in, not in) and in the second field type the value to be used. Values should be enclosed in quotes; for example: &quot;Proposed&quot;, If you type more than one value, separate them with a comma; for example: &quot;Proposed&quot;, &quot;Implemented&quot;.</td>
</tr>
<tr>
<td>Connector Direction</td>
<td>Filter connectors according to direction.</td>
</tr>
<tr>
<td></td>
<td>If you select Both, the connector is documented twice; once for the source element and once for the target. For the remaining two values, the connector is documented only for the source or target element, as appropriate.</td>
</tr>
<tr>
<td><strong>Order</strong></td>
<td></td>
</tr>
<tr>
<td>Packages by</td>
<td>Order packages in the generated documentation in either ascending or descending order of Name, Tree Order, Modified Date or Created Date.</td>
</tr>
<tr>
<td>Elements by</td>
<td>Order elements in the generated documentation in either ascending or descending order of Name, Tree Order, Modified Date or Created Date.</td>
</tr>
<tr>
<td>Diagrams by</td>
<td>Order diagrams in the generated documentation in either ascending or descending order of Name, Tree Order, Modified Date or Created Date.</td>
</tr>
<tr>
<td>Exclude details for</td>
<td>Exclude all elements of the selected type or types from the generated document.</td>
</tr>
<tr>
<td>Exclude connector type</td>
<td>Exclude all connectors of the selected type or types from the generated document.</td>
</tr>
</tbody>
</table>
The **Advanced** tab of the **Generate RTF Documentation** dialog has the following fields:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide 'note-less' elements</td>
<td>Exclude all elements without notes from the documentation.</td>
</tr>
<tr>
<td>Diagram Format</td>
<td>Set the diagram format for the images included within the documentation to either Metafile or Bitmap.</td>
</tr>
<tr>
<td>Skip root package</td>
<td>Exclude the parent package from the documentation and include only the child packages.</td>
</tr>
<tr>
<td>Overwrite document fields</td>
<td>Generate a document where the value fields defined in a document section are overwritten with the actual value text. If you open the generated report in Word, you can format the text but you cannot display the original field code. If the checkbox is deselected, the fields remain and are populated with the appropriate values; when you display the generated report in Word, you can right-click on the field and toggle between the value text and the field code.</td>
</tr>
<tr>
<td>Note:</td>
<td>If you use Open Office as your text editor, you must select this checkbox to show the field values. Open Office defaults to showing the field codes only, and you cannot toggle to the field values.</td>
</tr>
<tr>
<td>Adjust Heading Levels</td>
<td>Enable the RTF Generator to automatically restrict the levels of heading generated for nested sub-packages in a document. The generator reproduces heading levels down to the value you set. For example, if you have four nested levels of sub-packages and you set this field to:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Heading 2</strong>, all sub-packages in the report are documented under level 2 headings.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Heading 4</strong>, the first level of subpackages are documented under level 2 headings, the next level under level 3 headings, and the remainder all under <strong>Heading 4</strong>.</td>
</tr>
</tbody>
</table>
The Document Options dialog provides both sets of options (other than the Switch generator button); click on the OK button to save your changes.
14.1.1.8 Word Substitution

The Word Substitution tab of the Generate RTF Documentation dialog enables you to define translations of technical terms, in particular field names used in Enterprise Architect, into a language 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; the Enter Value field displays.

2. Type the foreign language translation in the Enter Value field, and click on the OK button.

14.1.1.9 Language Substitution

The Codepage tab of the Generate RTF Documentation dialog enables you to define languages other than English for direct substitution into RTF documentation.

If you export RTF-format documents from Enterprise Architect in languages other than English, you can customize the codepage, default language ID and character set that Enterprise Architect uses when generating RTF. This makes it much easier to generate documentation appropriate to your country or locale.
To define the language substitution, follow the steps below:

1. From the drop-down lists in the Language, Codepage and Charset fields, select the language, codepage and character set that most closely match your location.

2. If required, modify the Substitute Tag 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.

Now when you generate RTF documents, the substitute tags are used in the output.

**Note:**

You can transport these tag definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
14.1.2 The Legacy RTF Report Generator

Note:

The Legacy Generator is available if you have RTF templates created in releases of Enterprise Architect prior to 4.1, and you prefer to generate RTF reports using the original generator. However, as you can generate reports from these templates using the post-Enterprise Architect 4.1 RTF Generator, the Legacy Generator and instructions for its use are no longer updated.

However, reports produced using the Legacy RTF Generator do reflect the Rich Text Notes formatting feature in any text associated with elements.

Creating a Rich Text Format (RTF) document is a simple and flexible process. An RTF document is based on a package or an element in your project (more usually a package). To produce a document, you must select the package or element to report on in the Project Browser, Element List or Model Search, and press [F8].


The following topics provide assistance on using the Legacy Rich Text Format Report dialog to document your project.

- Document a Single Element
- The RTF Report Dialog
- Set the Main RTF Properties
- Apply a Filter
- Exclude Elements
- RTF Diagram Format
- Model Include
- RTF Report Options
- RTF Report Selections
14.1.2.1 Document a Single Element

RTF documentation can also be generated for a single element. Select the element to generate the documentation for, and then select the Element | Rich Text Format (RTF) Report menu option. The Generate RTF Documentation dialog displays. Click on the Switch Generator button to display the Rich Text Format Report dialog. See The RTF Report Dialog and related topics for further information.

14.1.2.2 The RTF Report Dialog

The Rich Text Format Report dialog enables 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 can 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.

You can switch back to the Generate RTF Documentation dialog by clicking on the Switch RTF Generator button.

Note:
The Rich Text Format Report dialog has a lot of options. Get to know them all to produce output at the level of detail suited to your project.
14.1.2.3 Set the Main RTF Properties

The main section of the Rich Text Format Report dialog enables you to set the output location and appearance of the final RTF document.

Setting Options for the RTF Document

1. Open the Rich Text Format Report dialog (see The Legacy RTF Report Generator topic 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 to save this report set-up.
   4. Select a report Style: Formal or Basic.
   5. Type a Heading for your report; this appears as the first heading item in your output.
   6. Select your required Heading Style and Initial Heading Level Indent from the drop-down 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. Also, if you move your report you must also move the images directory.

14.1.2.4 Apply a Filter

You can apply a filter on the Rich Text Format 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 Rich Text Format Report dialog (see The Legacy RTF Report Generator for how to do this). The Filter section of this dialog is shown below.

   • To enable the date filter, select the checkbox in the date field.
   • In the first two Only include objects fields, click on the drop-down arrows and select the appropriate criteria (Modified/Created, Before/After).
   • The package phase filter applies at the package level (not the element level) and ignores the phase of the
root package that you are documenting. To enable the phase filter, in the **Where Package Phase** field click on the drop-down arrow and select an operator; Enterprise Architect filters out all packages that do not meet the selection criteria. All elements in the package are ignored, regardless of their individual phase.

- The element status filter enables you to limit the output by element status. Unlike the package phase filter, this filter applies to every element. You can filter against a status of *like* or *unlike* a criterion, e.g. *like proposed*, or against the *in* and *not in* operators, e.g. *in approved, not in validated*. When using the *in* and *not in* operators, enter a comma-separated list of status types as your criteria expression.

### 14.1.2.5 Exclude Elements

The **Rich Text Format Report** dialog enables you to exclude elements of any type from your final output. This is useful when you want to highlight particular items and not clutter up a report with too much detail.

Open the **Rich Text Format Report** dialog (see [The Legacy RTF Report Generator](#) for how to do this). Look at the **Exclude details for** panel on the right of the dialog.

Click on each element to exclude, or click on the **All** button to exclude all elements. Click on the **None** button to clear your selections.

### 14.1.2.6 RTF Diagram Format

You can output diagrams to Bitmap files, GIF files or Windows Metafiles.

Open the **Rich Text Format Report** dialog (see [The Legacy RTF Report Generator](#) for how to do this). In the **Diagram format** panel (bottom center of the dialog) select the required format for the 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 can 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, they do not work very well with line drawings and lose clarity when re sized; JPEG file sizes are typically very small.
Note:
Generally metafiles are the best option, although it sometimes pays to experiment.

14.1.2.7 Project Include
The Model Include panel of the Rich Text Format Report dialog has the following options:
- Glossary to include the project glossary
- Tasks to include project tasks
- Issues to include project issues

Select the appropriate checkbox to include the items in the generated RTF documentation.

14.1.2.8 RTF Report Options
Additional RTF report options you can select from the Options panel on the Rich Text Format Report dialog are shown below.

You can select whether or not to recursively document packages, show diagrams or add a page break before each new package. Select the:
- Process all Children checkbox to recursively process all child packages within the main package
- Show Diagrams checkbox to include diagrams in your document. Clear this item for no diagrams
- New page per package checkbox to force a page break on each new package (excepting empty packages)
- Document all elements checkbox to include all elements included in the project
- Document Packages checkbox to document the package as an element in addition to the documentation that would normally be produced for package documentation
- Hide 'note-less' elements checkbox to exclude all elements without notes from the documentation
- Embed Diagrams in Document checkbox to ensure that the diagram images are contained within the RTF document rather than stored in a linked external file
- Skip root package checkbox to exclude the parent package from the documentation and include only the child packages
- Document Linked Elements checkbox to include the object details for linked elements that do not
originates from the selected package

- Use Heading styles for Details checkbox to ensure that the details are formatted as heading styles rather than formatted text; this option is only available when the Heading Style field in the Rich Text Format Report dialog is set to Max 9 levels - elements are package + 1.

### 14.1.2.9 RTF Report Selections

The For each Object Include section of the Rich Text Format Report dialog enables you to select the documentation sections to include in your report.

<table>
<thead>
<tr>
<th>For each Object Include</th>
</tr>
</thead>
</table>
| ![Image of a tree diagram with checkboxes for different sections such as General, Advanced, Features, Relationships, Links, Maintenance, and Project Management.](image)

What you include or exclude governs how simple or detailed your report is. You can 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.

Selecting the checkbox against a category item in the list selects all of the options that are contained in the category. To expand a category, click on the +symbol next to the category name. To exercise greater control over a category of options expand the top level item and then select the required individual items from the list.

Sometimes an item applies only to a certain type of element; for example, Attributes only applies to Class elements and a few other element types. The Child Diagrams option shows or hides any diagrams that are attached under a model element; for example, a Use Case might have a Scenario diagram attached.

**Note:**

Use this feature to produce the right level of detail for your audience. Technical readers might want to see everything, whilst management might require only the general outline.
14.1.2.10 Generate the Report

Once you have set up the document properties as required, click on the Create button to generate the report. When you have generated the document, click on the View button to open the report in MS Word.

14.1.2.11 Legacy RTF Style Templates

The Legacy RTF Style Editor enables 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’s look and feel for your company or client.

Notes:

- The RTF Style Editor discussed here automatically displays when you modify or create a Legacy RTF template. If you select a template created in the enhanced RTF Style Template Editor, that editor opens automatically instead.
- You can transport these RTF templates between models, using the Export Reference Data and Import Reference Data options on the Tools menu.

If you have previously defined and saved a template, click on the Load button on the Rich Text Format 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.

Create or Edit RTF Style Templates

1. Select the View | Resources menu option to display the Resources window.
2. Expand the Templates folder.
3. To edit an existing Legacy template, expand the RTF Style Templates tree and double-click on the template name, or right-click and select the Modify RTF Style Template context menu option. The RTF Style Editor displays. See RTF Style Editor for further details.
4. Alternatively, to create a new Legacy template, right-click on RTF Style Templates and select the Create RTF Style Template (Legacy) context menu option. Enterprise Architect displays a prompt for the new template name.
5. Type the name of the new template and click on the **OK** button. The RTF Style Editor displays. See RTF Style Editor for further details.

**Tip:**

To delete a template, right-click on it and select the Delete RTF Style Template context menu option.

**RTF Style Editor**

The RTF Style Editor 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 Enterprise Architect replaces 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; for example, #NOTES#

Click on the:

- **Get Default** button to retrieve the default Enterprise Architect template for the currently-selected template item in the left hand list
- **Save** button to save the version of the template for this style only
- **Delete** button to remove your modified version of the template, which causes Enterprise Architect to use the default template during report generation.

To select a template during report generation, click on the **Style** drop-down arrow on the Rich Text Format Report dialog. Once a style is selected, Enterprise Architect applies that to the current report. Select `<basic>` for the inbuilt style.
Tip:
You can also alter the custom language settings.

### 14.1.2.12 Save as Document

The Document feature enables a particular documentation configuration to be 'remembered', linking the loaded template within the Rich Text Format Report dialog to the current highlighted package. If a particular template is always used with a specific package, and multiple cases of documentation exist to be propagated, saving these as Documents can ease document generation later.

To create and use Documents, follow the steps below:

1. Open the Rich Text Format Report dialog (see Create a Rich Text Document for instructions on how to do this).
2. Click on the Save as Document button. The Save current as document definition dialog displays:

   ![Save as Document dialog](image)

3. In the Enter Value field, type a name for the document and click on the OK button. The document is added to the Resources window for easy future access (as for the usecasedoc entry in the illustration below).

4. To generate documentation from the Resources window, right-click on the required document. The context menu displays.
5. Select the required option. The context menu options are:
   - **Open Document** - Opens the corresponding .RTF file, as specified by the RTF template Filename property
   - **Generate Document** - Opens the Rich Text Format Report dialog, loaded with the specified template
   - **Auto Generate Document** - Generates documentation, with the document located at the path specified by the template's Filename property
   - **Delete Document** - Removes the specified document.

### 14.1.2.13 Custom Language Settings

If you export RTF-format documents from Enterprise Architect in languages other than English, you can customize the codepage, default language ID and character set that Enterprise Architect uses when generating RTF. This makes it much easier to generate documentation appropriate to your country or locale.

You can also set up a list of word substitutions. For instance, where Enterprise Architect 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. Open the Rich Text Format Report dialog (see The Legacy RTF Report Generator for how to do this).
2. In the Language panel (bottom left of dialog) click on the Adjust button. The Customize RTF Language dialog displays.
3. Double-click on an item to set or clear its Substitute word.
4. When you have finished, click on the OK button.

To Set Up Codepage and Character Set

1. From the drop-down lists in the Language, Codepage and Charset fields, select the language, codepage and character set that most closely match your location.
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 have completed the settings, click on the OK button to save them.

Now when you generate RTF documents, the substitute tags are used in the output.

Note:
You can transport these language and tag definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
14.1.3 Use 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 documents with sustainable links to generated 'pieces' of Enterprise Architect output, pre-divided by Enterprise Architect using bookmarks.

As an alternative to the Word master document, internal to Enterprise Architect, see Virtual Documents. In addition to creating Word master documents, you can:

- Open a Report in Microsoft Word
- Change Linked Images to Embedded
- Apply Other Features of Word

14.1.3.1 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 converts the file. 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 click on the View Output button on the Generate RTF Documentation dialog.

14.1.3.2 Change Linked Images to Embedded

One of the options 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 in 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 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.

Break Image Links in Word

1. Open the required RTF file in Word.
2. Select the Edit | Links menu option.
3. Highlight all links in the Links list.
4. Select the Save Picture in Document checkbox.
5. Click on the Break Link button.
6. When prompted, click on the Yes button to break links.

Word breaks the links and saves copies of the images inside the document. You can distribute this document without the image directory.
14.1.3.3 RTF Bookmarks

Bookmarks are markers that are automatically placed in your rich text document when you generate it. You can create a master document in Word and link to sections of an Enterprise Architect report based on bookmarks. For example, a Word document might 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 Enterprise Architect reports. If you link into Enterprise Architect reports, then you can regenerate the report and refresh Word links to update the master document without having manually changed anything. For more information on refreshing links, see the Refresh Links topic.

Bookmarks are GUID-based numbers that can be created for packages, diagrams and elements. A package bookmark applies from the beginning of a package to the end, and includes all child packages and elements underneath.

Notes:
You cannot use RTF Bookmarking in Master Document elements, which effectively replace RTF Bookmarking in Word.

RTF Bookmarking requires each bookmark to be unique. When you generate a report with a standard RTF template (including in a single Model Document element), each bookmark is unique and there is a 1:1 association between the Elements-details being generated and the elements in the repository. As Master Documents are intended to contain multiple sub-documents, the association ceases to be 1:1. There is no simple method that enables the generated data to be uniquely identified directly in association with the original element.

Bookmark a Section of Enterprise Architect for RTF Documentation

1. In the Enterprise Architect Project Browser, right-click on the package to include in the documentation. The context menu displays.
2. Select the Documentation | Copy RTF Bookmark menu option to paste the package into the clipboard as a bookmark for use in Word.
Insert a Bookmarked Section of an Enterprise Architect RTF Document into Word

1. Open the Word document and position the cursor at the point at which to insert the file.
2. Select the Word **Insert** | **File** menu option. The **Insert** dialog displays.
3. Locate and click on the file to insert, then click on the **Range** button.
4. In the **Range** cell type or paste the information from the clipboard.
5. Click on the OK button.
6. Click on the drop-down arrow next to the Insert button. Select the Insert as Link option.

The Insert option sets a permanent copy; the Insert as Link option creates a link that is updateable on altering the source document. For Insert as Link to operate you must first 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.

### 14.1.3.4 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 Enterprise Architect 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.

#### 14.1.3.4.1 Add Table of Contents

Among the features of MS word that can be incorporated into generated Enterprise Architect 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 Enterprise Architect RTF reports. This option provides 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 follow the steps below:

1. Open the Enterprise Architect RTF report to which to add a Table of Contents in MS Word.
2. Select the Insert | Reference | Index and Tables menu option.
3. Click on 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 heading levels created when the RTF is generated. To set the heading style for details in Enterprise Architect RTF documentation, see the RTF Document Options topic.

### 14.1.3.4.2 Add Table of Figures

Among the features of MS word that can be incorporated into generated Enterprise Architect 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 Enterprise Architect 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, follow the steps below:

1. Open the Enterprise Architect RTF report to which to add a Table of Figures in MS Word.
2. Select the **Insert | Reference | Index and Tables** menu option.
3. Click on the **Table of Figures** tab to set the options that are available for formatting the table of figures.
14.1.3.4.3 Add Headers and Footers

Among the features of MS Word that can be used to enhance the appearance of Enterprise Architect RTF reports is the ability to add headers and footers to the documentation. To include headers and footers in the RTF documentation follow the steps below:

1. Open the Enterprise Architect RTF report to which to add headers and footers in MS Word.
2. Select the View | Header and Footer menu option.

This enables you to enter information into the header section and the footer section of the RTF Documentation.
14.1.3.4.4 Manipulate Tables in Word

When generating RTF documentation from Enterprise Architect, 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 customization 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.

Resize 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 follow the steps below:

1. Select the table that exceeds the margin size.
2. Mouse over the border of the table until the mouse pointer changes into the icon shown below.
3. Drag the cursor to the left to reduce the width of the table and then select the File | Print Preview menu.
option 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 enables you to rapidly change the appearance of the table. To achieve this effect follow the steps below:

1. Open the Enterprise Architect RTF report in which to change the table styles.
2. Locate and select the table for which to adjust the appearance.
3. Select the **Table | Table Auto Format** menu option. The **Table Autoformat** dialog displays.

   From here you can specify a predefined table style from the **Table styles** list, or create a new style by clicking on the **New** button. The table styles defined in the **Table Autoformat** dialog only apply to one table at a time so you must apply the style to each table created individually.

14.1.3.4.5 Refresh Links

If you link into Enterprise Architect reports, then you can regenerate the report and refresh MS Word links to update the Word master document without having manually changed anything.

To ensure that links are refreshed in the master document, you must select the **Update automatic links at Open** checkbox in Word. To ensure that this setting is established follow the steps below:

1. From within MS Word select the **Tools | Options** menu option.
2. Select the **General** tab and select the **Update automatic links at Open** checkbox.
<table>
<thead>
<tr>
<th>General options</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Background repagination</td>
</tr>
<tr>
<td>- Blue background, white text</td>
</tr>
<tr>
<td>- Provide feedback with sound</td>
</tr>
<tr>
<td>- Provide feedback with animation</td>
</tr>
<tr>
<td>- Confirm conversion at Open</td>
</tr>
<tr>
<td>- Update automatic links at Open</td>
</tr>
<tr>
<td>- Mail as attachment</td>
</tr>
<tr>
<td>- Recently used files list: 4 entries</td>
</tr>
<tr>
<td>- Help for WordPerfect users</td>
</tr>
<tr>
<td>- Navigation keys for WordPerfect users</td>
</tr>
<tr>
<td>- Allow background open of web pages</td>
</tr>
<tr>
<td>- Automatically create drawing canvas when inserting AutoShapes</td>
</tr>
</tbody>
</table>

**Measurement units:**

- Inches

- Show pixels for HTML features

[Web Options...][Email Options...]

[OK][Cancel]
14.1.4 Other Documents

Enterprise Architect has other RTF based documentation that you can output:

- **Dependency Report**
- **Diagram Only Report**
- **Implementation Report**
- **Resource Report**
- **Testing Report**
- **Testing Details Report**

14.1.4.1 Dependency Report

A *Dependency Report* shows, for the selected package, a list of any elements that are dependent on another element for their specification. For example, a Use Case derives its specification from the Requirement that it realizes. Each of the elements in the first column is the source or dependent in a *Dependency* connector to the corresponding target element in the second column.

To view a Dependency report, follow the steps below:

1. In the *Project Browser*, right-click on the package to report on; the report includes all sub-packages of this package. The context menu displays.
2. Select the *Documentation | Dependency Report* menu option.
3. The *Dependency Details* dialog displays, showing the results of the report. Save or print the results if required.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Package</td>
<td>Confirm the root package. All elements and packages under this package appear in the report.</td>
</tr>
<tr>
<td>Locate Object</td>
<td>Locate the selected element in the <em>Project Browser</em>.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Run the report again.</td>
</tr>
<tr>
<td>Details</td>
<td>List dependency details; lists the elements in the current package and the elements that they implement.</td>
</tr>
<tr>
<td>Print</td>
<td>Print the list.</td>
</tr>
</tbody>
</table>
14.1.4.2 Diagram Only Report

You can also produce an RTF report that contains only the relevant diagrams from the target package. This is convenient for printing or handling a lot of diagrams in batch, rather than exporting or printing each one at a time.

To Produce a Diagram Only Report

1. Right-click on a package in the Project Browser. The context menu displays.
3. Select the options you require, as follows:
   - Select the Embed Diagrams in Document checkbox to ensure the diagrams are created within the RTF file, not as linked image files.
   - Select the Include all child packages checkbox to document all of the diagrams included in any child package.
   - Select the Include Diagram Name checkbox to include the diagram name within the generated documentation.
   - Select the Order Diagrams Alphabetically checkbox to generate the documentation in alphabetical order.
4. Click on the Generate button to run the report.
5. When the report is generated, click on the View button to show the RTF output.

14.1.4.3 Implementation Report

An Implementation report lists, for a specified package, the elements that require implementers, together with any source elements in Realize (Implements) relationships with those elements.

To view an Implementation report, follow the steps below:

1. In the Project Browser, right-click on the package to report on; the report includes all sub-packages of this package. The context menu displays.
2. Select the Documentation | Implementation Report menu option.
3. The Implementation Details dialog displays with the results for the package. Save or print the results if required.
### Option | Use to
--- | ---
Root Package | Confirm the root package. All elements and packages under this package appear in the report.
Show Unimplemented | Show non-implemented elements. Non-implemented elements are those that don't have any other element to realize them (e.g., 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 selected element in the Project Browser.
Refresh List | Run the report again.
Details | List elements in the current hierarchy and elements that implement them.
Print | Print the list.
Set Target Types | Set the list of types to report on. By default, Enterprise Architect only reports on a limited number of element types, such as Use Cases and Requirements. For further information, see the Set Target Types Dialog topic.
14.1.4.3.1 Set Target Types Dialog

The Set Target Types dialog is accessed by clicking on the Set Target Types button on the Implementation dialog. This dialog enables you to set the types of elements that appear in the report as requiring implementation. Double-click on an element in either list to move it to the other list.

14.1.4.4 Testing Report

To view a testing report, follow the steps below:

1. In the Project Browser, right-click on the package to report on (you can configure the report to include all sub-packages as well) to open the context menu.
3. The Generate Test Documentation dialog specifies which test types are documented in the report, and the output file location. You can also select the Include child packages checkbox to report on all child packages of the selected package. The test data originates in the docked testing window, where tests are created and attached to the respective object.
14.1.5 Virtual Documents

Virtual documents enable you to structure and filter your RTF reports by selecting, grouping and ordering individual packages independent of the organization of the Project Browser. You can create separate virtual documents defining, say, Requirements, Use Cases or Design elements of a project, or you can combine these separate reports - retaining their own different formats - into a single generated document with common headers and footers and a central contents list. This combined document could apply your corporate standards.

You generate virtual documents in Enterprise Architect from individual Model Document elements. You can also, if required, combine several Model Documents under a Master Document package element.

Each Model Document element identifies its own template; for example, a specifically-designed Requirements template for a Requirements document, or a Use Case template for a section on Use Cases. The template is identified in a Tagged Value, and defines the content as either:

- A list of packages (defined as attributes) in whatever order or combination is most appropriate to your requirements - you can easily add or delete packages as necessary;
- A standard model search (defined as Tagged Values) created within the Model Search facility (note that diagram searches are not supported); when you generate the document, this search captures the required data throughout the model, and populates the document.

Notes:

- You cannot use RTF Bookmarking in Master Document elements, which effectively replace RTF Bookmarking in Word.

RTF Bookmarking requires each bookmark to be unique. When you generate a report with a standard RTF template (including in a single Model Document element), each bookmark is unique and there is a 1:1 association between the Elements-details being generated and the elements in the repository. As Master Documents are intended to contain multiple sub-documents, the association ceases to be 1:1. There is no simple method that enables the generated data to be uniquely identified directly in association with the original element.

- In a Model Document, you should not define both a list of packages and a search; if both are present, when you generate the document Enterprise Architect works from the package list only.

You can control the sequence in which information is presented in the document; see the Document Order topic.

Tip:

You can create as many Model Documents as required, for as many combinations of information as required.

The Master Document element contains its own template Tagged Value, which defines the headers, footers and central contents list. You can import your corporate standards template and edit the Tagged Value to identify that.

Document Elements

14.1.5.1 Create Master Document

Whilst you can create Model Document elements separately and generate individual documents from each one, you have added flexibility and scope if you organize Model Documents under a Master Document. You can generate a document with a corporate template for the covers, contents, headers and footers, whilst each section (generated from a separate Model Document) has its own appearance defined by a template appropriate to the section content.

To create a Master Document element, follow the steps below:

1. Open or create the diagram in which to create the Master Document.
3. Drag the Master Document icon onto the diagram. The system prompts you for the name of the Master Document.
4. Type the element name and click on the OK button. The system creates the Master Document element and a child Custom diagram of the same name.
5. Open the Tagged Values window (View | Tagged Values) and click on the Master Document element. The RTFTemplate Tagged Value displays in the window.
6. The RTFTemplate Tagged Value defaults to *(model document: master template)*. If you want to use an alternative master template, click on the drop down arrow at the right of the field and click on that template in the list.

7. Return to the Project Browser and open the Master Document child diagram.

At this point, you *create the Model Document* for the elements in the child diagram, to provide the content for the generated document.

When you have added all your Model Document elements to the Master Document diagram, the Master Document element resembles the following:

![Diagram of Master Document element]

Your completed Master document element and child diagram display in the Project Browser as shown below:
14.1.5.2 Create Model Document

You can create as many Model Document elements as are necessary to provide the sections of your generated document (under a Master Document) or to provide the independent documents you require.

To create a Model Document element, follow the steps below:

1. Open the Master Document child diagram or (if you are creating independent Model Documents) create a new diagram.

   Tip:
   The new diagram can live anywhere outside the packages you are adding to the document; you could create a Class diagram called Documentation within a specific Documentation package, and use this to hold the independent Model Document elements for your virtual documents.

2. From the Documentation page of the Enterprise Architect UML Toolbox (More Tools | Extended | Documentation) drag the Model Document icon onto the diagram to create a new Model Document element. Give the element an appropriate name; for example, if the documentation is relevant to the shopping cart requirements of a model, you could call it ShoppingCartReqDocumentation. Click on the OK button.

3. Open the Tagged Values window (View | Tagged Values) and click on the Model Document element. The RTFTemplate, SearchName and SearchValueTagged Values display in the window.

4. Click on the drop-down arrow to the right of the RTFTemplate field, and click on the template to use for this Model Document.

5. If you are creating a list of packages for the Model Document, go now to Add Packages to Model Document. Otherwise, click on the drop-down arrow to the right of the SearchName field, and click on the model search type to populate this Model Document.

Notes:
- Diagram Searches are not supported.
- Custom SQL searches are supported if they are returning elements. The SQL must include `ea_guid AS CLASSGUID` and the object type.
6. If necessary, type a search term in the SearchValue field.
7. Create further Model Document elements as required.
Your Model Document element appears in the Project Browser with a Class icon, as shown below:

When you have created all the required Model Document elements, see the Document Order topic.

14.1.5.3 Add Packages to Model Document

To add packages to your Model Document element, follow the steps below. As the example Model Document element here is called ShoppingCartDocumentation, the steps indicate how to add shopping cart-related packages to the element.

1. Keeping the documentation diagram open, find a package in the Project Browser to add to the documentation. For example, a Communication package in a Dynamic view.
2. Drag and drop the package from the Project Browser onto the Model Document element as shown below:

3. The title of the package displays in the Model Document element in the Attributes compartment, as shown below:

4. This means that the Communication package is included in the document when you generate it. Using the above method, you can add as many packages from as many different views as required.
The next step is to generate your document, but consider the impact of your package list on the Document Order. You can also delete packages if required.

14.1.5.4 Delete Package in Model Document

You can delete a package from your Model Document element.
This example includes four packages:
- Communication Package
- Activity Diagram Package
- View Cart Package
- Sequence Diagram Package.
To delete a package from a Model Document element, follow the steps below:

1. In the Project Browser, expand the element to list the package attributes.
2. Right-click on the package to delete, and select the Delete Attribute context menu option.

Alternatively:

1. In either the Project Browser or the diagram, right-click on the Model Document element and select the Attributes context menu option. The Attributes dialog displays.
2. On the Attributes list, click on the package to delete.
3. Click on the Delete button to remove the package from the document element.

### 14.1.5.5 Document Order

The order in which information is compiled into an RTF document depends on:

- The sequence of Model Document elements in a Master Document element
- Whether you define a Model Search in a Model Document element
- Whether you define a package list in a Model Document element.

When you have considered and, if necessary, amended the order in which information is compiled, you can generate the document.

### Model Document Sequence

When you generate a document from a Master Document element, the sequence in which the sections are generated is determined by the order in which the child Model Document elements are listed in the Project Browser. You can create elements anywhere in a diagram, therefore the generator refers to the Project Browser sequence.

If necessary, change the sequence using the green Up and Down arrows in the Project Browser toolbar to move an element up or down within the package.

### Model Search

A Model Search operates on the database, and accesses records in the order in which they are stored. This order depends on many factors, and can change with database maintenance. Therefore, the sequence of information provided by the search is unpredictable.
Package Order

When you create a package list in a Model Document element, the sequence of information is determined by the order in which the package attributes are listed within the element. You can change the sequence using the Attributes dialog, and if you prefer a package to be in a different section of the document, you can move the attribute from one Model Document element to another. Both these procedures are described below.

Change Sequence of Packages In List

To rearrange the order of packages in a Model Document element, refer to the example element and follow the steps below:

1. Right-click on the Model Document and select the Attributes option from the context menu. The Attributes dialog displays.
2. On the Attributes list, click on a package to move and click on the Up or Down (hand) buttons to change the order in which the packages are included in the documentation.
3. When you are satisfied with the order of your packages, click on the Close button.

Move Package Between Elements

To move a package attribute from one Model Document to another, follow the steps below:

1. Expand the Model Document elements in the Project Browser, so that both list their package attributes.
2. Click-and-hold on the attribute to move, and drag it onto the name of the target Model Document element.
3. Release the mouse button. The attribute is removed from the source element and added to the top of the list of attributes in the target element.
4. If necessary, move the attribute down the attribute list, as described in Change Sequence of Packages in List, above.

14.1.5.6 Generate the Document

To generate the documentation defined in the Master Document and/or Model Documents, follow the steps below:

1. On the documentation diagram, click on the Master Document element (or on an independent Model Document element).
2. Select the **Element | Rich Text Format (RTF) Report** menu option. The **Generate RTF Documentation** dialog displays.

3. Set the options for your RTF document as required. See **The RTF Report Dialog** and related topics for further information on these settings.

4. Click on the **Generate** button to create the documentation.

5. If you have not selected the **View Document on Completion** checkbox, click on the **View** button to view the documentation.

The RTF Report Generator works through the defined content of the Master Document element and/or the Model Document elements and pulls in the information from either the listed packages or the executed searches, formatted according to the templates identified in the **RTFTemplate** Tagged Value for each document element.
14.2 HTML Reports

Enterprise Architect provides automated web-based publishing of models. A new outline structure closely mirrors the model hierarchy and makes it very simple to explore models on-line. With a great new look and the ability to explore very large models efficiently on-line, the new web-publishing capability is a significant enhancement.

Enterprise Architect enables 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 very simple.

The current implementation is based on internal and external templates and generated Javascript. The ability to edit all templates is to be added in a future version of Enterprise Architect.

Tip:

You can create Web Style Templates to customize your HTML output.

Note:

In the Corporate edition of Enterprise Architect, if security is enabled you must have Generate Documents permission to generate HTML documents.
14.2.1 Create 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 are included in the output). The context menu displays.
3. In the Output to field, select an output directory for your report. Set any other required options.
4. Click on the Generate button to generate the report. The Progress field shows total percentage complete.
5. Once the report is complete, click on the View button to launch your default HTML viewer and view the web pages.

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 it within the context of your web server. All path names should be relative and case sensitive.

Quick Start

To generate an HTML report right now, follow the steps above on the System Model package of the EAEexample project.

Note:

If you are using Microsoft Internet Explorer 7.0 or later, and you do not have it open, its security profile might block the report display. Click on the explanation banner at the top of the screen and select the Allow Blocked Content context menu option.
### 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, as described below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Display the name of the package you are creating documentation for.</td>
</tr>
<tr>
<td>Title</td>
<td>Type the title for your HTML documentation; defaults to Package.</td>
</tr>
<tr>
<td>Output to</td>
<td>Type the directory path your documentation is saved to.</td>
</tr>
<tr>
<td>Style</td>
<td>Select a web style template to apply to your documentation (optional).</td>
</tr>
<tr>
<td>File extension</td>
<td>Specify the file extension for your HTML documentation files; the default is .htm.</td>
</tr>
<tr>
<td>Preserve White space in Notes</td>
<td>Preserve existing white space in your notes; deselect to remove white space.</td>
</tr>
<tr>
<td>No page for Note and Text items</td>
<td>Omit the page for your notes and text items in the HTML report.</td>
</tr>
<tr>
<td>Default Diagram</td>
<td>Select the diagram the report should open to when the generated documentation is loaded.</td>
</tr>
<tr>
<td>Image Format</td>
<td>Select the image file format to save your images in, either PNG or GIF.</td>
</tr>
<tr>
<td>Include</td>
<td>Select each area of your model to include in your report.</td>
</tr>
<tr>
<td>System</td>
<td>Select each section to generate in your report.</td>
</tr>
</tbody>
</table>

Click on the **Generate** button to generate the HTML report with the settings you have defined.

Click on the **View** button to display the report you have generated.
14.2.3 Web Style Templates

The HTML and CSS Style Editor enables 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’s look and feel for your company or client.

To create or edit web style templates, follow the steps below:

1. In the Resources window, expand the Templates folder.
2. To edit an existing template, expand the Web Style Templates folder and either double-click on the template name, or right-click and select the Modify HTML Style Template option from the context menu. The HTML and CSS Style Editor displays. See below for further details.
3. To create a new template, right-click on the Web Style Templates folder and select Create HTML Template from the context menu. Enter a name for the new template when prompted to do so. The HTML and CSS Style Editor displays. See below for further details.

Note:

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 Enterprise Architect replaces 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 - e.g. #NOTES#

- Get Default retrieves the default Enterprise Architect 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 causes Enterprise Architect 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, Enterprise Architect applies that to the current report. Select <default> for the inbuilt style.
Note:
Each time Enterprise Architect generates the web report it overwrites these files, so you must back up your modified versions and copy them back in after every update.
Part XV
15 UML Dictionary

The Unified Modeling Language (UML)

Enterprise Architect's modeling platform is based on the Unified Modeling Language (UML) 2.1, 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 can 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.

UML is not a prescriptive process for modeling 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. This language is designed to be flexible, extendable and comprehensive, yet generic enough to serve as a foundation for all system modeling requirements. 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, Tagged Values and profiles. Enterprise Architect supports many different kinds of UML elements (as well as some custom extensions). Together with the connectors between elements, these form the basis of the model.

See:
- UML Diagrams
- UML Elements
- UML Connectors

Wide Range of Applications

Although initially conceived as a language for software development, UML can be used to model a wide range of real world domains and processes (in business, science, industry, education and elsewhere), organizational hierarchies, deployment maps and much more. Enterprise Architect also provides additional custom diagrams and elements, to address further modeling interests. This topic is intended to provide an introduction to Enterprise Architect's diagrams, elements and connectors, and its modeling process. It also illustrates 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 Enterprise Architect can be tailored to address a particular modeling domain not explicitly covered in the original UML specification. Enterprise Architect makes extending UML simple and straightforward and, best of all, the extension mechanism is still part of the UML Specification.

Find Out More

UML is an open modeling standard, defined and maintained by the Object Management Group. Further information, including the full UML 2.1.1 documentation, can be found on the OMG website at http://www.omg.org.

Tip:
If you are unfamiliar with UML, please explore the topics in this UML Dictionary and the Enterprise Architect UML Toolbox descriptions, and the EAExample project supplied with Enterprise Architect. The online UML Tutorial (parts 1 and 2) and UML 2.0 Tutorial are also very helpful.

Recommended Reading:
In addition to the UML Specification available from the OMG, two books that 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.
15.1 UML Diagrams

What is a UML Diagram?
A UML diagram is a representation of the components or elements of a system or process model and, depending on the type of diagram, how those elements are connected or how they interact from a particular perspective. For example, how and why an object changes state, or how requirements are realized by the process or a system.

Types of Diagram
There are two major groupings of UML diagrams:
- **Structural Diagrams** which depict the structural elements composing a system or function, reflecting the static relationships of a structure, or run-time architectures.
- **Behavioral Diagrams** which show a dynamic view of the model, depicting the behavioral features of a system or business process.

Enterprise Architect provides the following additional diagram types that extend the core UML diagrams for business process modeling, formal requirements specifications and other domain-specific models:
- **Analysis** diagrams
- **Custom** diagrams
- **Requirements** diagrams
- **Maintenance** diagrams
- **User Interface** diagrams
- **Database** diagrams
- **Business Modeling and Business Interaction** diagrams.

Enterprise Architect also supports diagram types specific to MDG Technologies, including integrated technologies such as **Mind Mapping**, **Iconix**, **BPMN**, and **Data Flow Diagrams**.

Work with Diagrams
Diagrams are developed in the main workspace in which you create and connect model elements. You create them by right-clicking a package and selecting the **New Diagram** context menu option, or load them by double-clicking their diagram icon in the **Project Browser**.

For full details on how to work with diagrams, see **Diagram Tasks**.
15.1.1 Behavioral Diagrams

Behavioral diagrams depict the behavioral features of a system or business process. Behavioral diagrams include the following diagram types:

**Activity Diagrams**

Activity diagrams model the behaviors of a system, and the way in which these behaviors are related in an overall flow of the system.

**Use Case Diagrams**

Use Case diagrams capture Use Cases and relationships among Actors and the system; they describe the functional requirements of the system, the manner in which external operators interact at the system boundary, and the response of the system.

**State Machine Diagrams**

State Machine diagrams illustrate how an element can move between states, classifying its behavior according to transition triggers and constraining guards.

**Timing Diagrams**

Timing diagrams define the behavior of different objects within a time-scale, providing a visual representation of objects changing state and interacting over time.

**Sequence Diagrams**

Sequence diagrams are structured representations of behavior as a series of sequential steps over time. They are used to depict work flow, message passing and how elements in general cooperate over time to achieve a result.

**Communication Diagrams**

Communication diagrams show the interactions between elements at run-time, visualizing inter-object relationships.

**Interaction Overview Diagrams**

Interaction Overview diagrams visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose.

15.1.1.1 Activity Diagram

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:**

You can create Analysis diagrams (Simplified Activity), containing the elements most useful for business process modeling, using the New Diagram dialog.

**Example Diagram**

The following diagram illustrates some of the features of Activity diagrams, including Activities, Actions, Start Nodes, End Nodes and Decision points.
Toolbox Elements and Connectors

Select Activity diagram elements and connectors from the Activity pages of the Enterprise Architect UML Toolbox.

Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Activity Diagram Elements</th>
<th>Activity Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>Fork/Join</td>
</tr>
<tr>
<td>Structured Activity</td>
<td>Fork/Join</td>
</tr>
<tr>
<td>Action</td>
<td>Control Flow</td>
</tr>
<tr>
<td>Partition</td>
<td>Object Flow</td>
</tr>
<tr>
<td>Object</td>
<td>Interrupt Flow</td>
</tr>
</tbody>
</table>
### Activity Diagram Elements

<table>
<thead>
<tr>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Buffer Node</td>
</tr>
<tr>
<td>Datastore</td>
</tr>
<tr>
<td>Decision</td>
</tr>
<tr>
<td>Merge</td>
</tr>
<tr>
<td>Send</td>
</tr>
<tr>
<td>Receive</td>
</tr>
<tr>
<td>Synch</td>
</tr>
<tr>
<td>Initial</td>
</tr>
<tr>
<td>Final</td>
</tr>
<tr>
<td>Flow Final</td>
</tr>
<tr>
<td>Region</td>
</tr>
<tr>
<td>Exception</td>
</tr>
</tbody>
</table>

### Activity Diagram Connectors

<table>
<thead>
<tr>
<th>Connector</th>
</tr>
</thead>
</table>

### 15.1.1.2 Use Case Diagram

A *Use Case* diagram captures *Use Cases* and relationships between *Actors* and the subject (system). It describes the functional requirements of the system, the manner in which outside things (Actors) interact at the system boundary, and the response of the system.

In developing a Use Case diagram, also consider:

- *Use Case Extension Points*
- *Use Rectangle Notation*
- *Business Use Case* (stereotyped Use Case)

**Example Diagram**

The following diagram illustrates some features of Use Case diagrams:
Toolbox Elements and Connectors

Select Use Case diagram elements and connectors from the Use Case pages of the Enterprise Architect UML Toolbox.

Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Use Case Diagram Elements</th>
<th>Use Case Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Use</td>
</tr>
<tr>
<td>Use Case</td>
<td>Associate</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Generalize</td>
</tr>
<tr>
<td>Boundary</td>
<td>Include</td>
</tr>
<tr>
<td>Package</td>
<td>Extend</td>
</tr>
<tr>
<td></td>
<td>Realize</td>
</tr>
<tr>
<td></td>
<td>Invokes</td>
</tr>
<tr>
<td></td>
<td>Precedes</td>
</tr>
</tbody>
</table>

15.1.1.3 State Machine Diagram

Note:
State Machine diagrams were formerly known as State diagrams.

A State Machine diagram illustrates how an element (often a Class) can move between states, classifying its behavior according to transition triggers and constraining guards. Other aspects of State Machine diagrams further depict and explain movement and behavior. State Machine representations in UML are based on the
Harel State Chart Notation (see the OMG UML Superstructure Specification 2.1.1, section 15.1), and therefore are sometimes referred to as State Charts.

You can display a State Machine as a diagram (as below) or as a table in one of three relationship formats. In all formats, you use the same Enterprise Architect UML Toolbox elements and connectors.

To select the display format, follow the steps below:

1. Right-click on the diagram background to display the context menu.
2. Select the Statechart Editor option.
3. Select the appropriate display option:
   - Diagram
   - Table (State-Next State)
   - Table (State-Trigger)
   - Table (Trigger-State)

Example Diagram

The following diagram 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. Composite States and sub-states are both State elements, a Composite State being an expanded State element that encloses other State elements, which are then referred to as sub-states. Composite States and State Machines can also contain Regions.

Note:

State elements can display either with or without a line across them. The line - as shown below - displays when the element has features such as attributes (which could be hidden) or when the Show State Compartment checkbox is selected in the Options | Objects dialog.
Toolbox Elements and Connectors


Tip:

Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>State Machine Diagram Elements</th>
<th>State Machine Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>![State]</td>
<td>![Fork/Join]</td>
</tr>
<tr>
<td>![State Machine]</td>
<td>![Fork/Join]</td>
</tr>
<tr>
<td>![Initial]</td>
<td>![Transition]</td>
</tr>
<tr>
<td>![Final]</td>
<td>![Object Flow]</td>
</tr>
</tbody>
</table>
State Machine Diagram Elements | State Machine Diagram Connectors

- History
- Synch
- Object
- Choice
- Junction
- Entry
- Exit
- Terminate

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, Section 15.3.12, p. 560*) 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) state machine.

**15.1.1.3.1 Regions**

Regions can be created in Composite States or State Machines on a State Machine diagram. 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 or State Machine element, follow the steps below:

1. Right-click on the element, and select the Advanced | Define Concurrent Substates menu option. The State Regions dialog displays.
2. Create the Regions of a State, which can be named or anonymous.

3. Click on the **OK** button.

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 544*) states:

A region is an orthogonal part of either a composite state or a state machine. It contains states and transitions.

### 15.1.1.3.2 Pseudo-States

Pseudo-states are a UML 2.1.1 abstraction for various types of transient vertices used in State Machine diagrams. Pseudo-states are used to express complex transition paths. The following types of pseudo-state are available:

- **Initial**
- **Entry Point**
- **Exit Point**
- **Choice**
- **Junction**
- **History**
- **Terminate**
- **Final**
- **Fork**
- **Join**

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 536*) 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.

### 15.1.1.4 State Machine Table

A State Machine table is one of two variants of a State Machine (the other is the State Machine diagram). It displays the information of the State Machine in table form, and is a method of specifying the discrete behavior of a finite state-transition system; that is, what state the State Machine moves to and the conditions under which the transition takes place.

You can display the state transition as one of two different relationships:
- **State - Trigger**: The rows indicate the current states and the columns indicate trigger events (or the other way around if you prefer, in a **Trigger - State** format). The cell at the intersection of a row and column identifies the target state in the transition if the trigger occurs, and the condition (or guard) of the transition.

<table>
<thead>
<tr>
<th>State</th>
<th>Event1</th>
<th>Event2</th>
<th>Event3</th>
<th>Event4</th>
<th>Event5</th>
<th>Event6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>S0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State1</td>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S2</td>
</tr>
<tr>
<td>State2</td>
<td>S2</td>
<td>S8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubState1</td>
<td>S3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubState2</td>
<td>S4</td>
<td></td>
<td></td>
<td></td>
<td>[Cond]</td>
<td>S5</td>
</tr>
<tr>
<td>SubState3</td>
<td>S5</td>
<td></td>
<td>[guard]</td>
<td></td>
<td></td>
<td>S4</td>
</tr>
<tr>
<td>State3</td>
<td>S6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S7</td>
</tr>
<tr>
<td>Final</td>
<td>S7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **State - Next State**: The rows and columns both indicate states, and the cell at the intersection of a row and column indicates the event that triggers a transition from the current (row) state to the next (column) state, the condition (or guard) of the event, and the effect of the transition.
Select Format
You can display a State Machine as a diagram or table, and as a table in one of three relationship formats. To select the display format, follow the steps below:

1. Right-click on the diagram background to display the context menu.
2. Select the **Statechart Editor** option.
3. Select the appropriate display option:
   - Diagram
   - Table (State-Next State)
   - Table (State-Trigger)
   - Table (Trigger-State)

To define the State Machine Table further, see:
- **State Machine Table Options**
- **State Machine Table Operations**

15.1.1.4.1 **State Machine Table Options**
You can choose the **State Machine table** layout and set other options from the State Machine Diagram: Options dialog, which you display by either:
- Double-clicking on the State Machine table background or
- Right-clicking on the background and selecting the **State Table Options** context menu option.
Option | Use to
--- | ---
**Table Format** | Select the required table format:
- **State - Trigger**: rows represent States, each state name in a left edge cell; columns represent Triggers, each trigger name in a column header cell; the intersection of a row and column identifies the Transition (if there is one); the Transition cell displays information about the next State and the condition (guard) of the Transition.
- **Trigger - State**: as above, except that rows represent triggers and columns represent states.
- **State - Next State**: both rows and columns represent states; intersection of row and column defines the transition (if there is one) from the row state to the column state.

**Cell Size**
- **Transition Cell Width**: Specify the width of the transition cells (that is, the column width).
- **Transition Cell Height**: Specify the height of the transition cells (that is, the row height).
- **Left Edge Cell Width**: Specify the width of the left edge (row title) cells.
- **Top Edge Cell Height**: Specify the height of the top edge (column title) cells.

**Cell Color**
- **State/Trigger Cell**: Select the color of the row and column title cells.
- **State/Trigger Enumeration**: Select the color of the enumeration (row/column numbering) cells.
15.1.1.4.2 State Machine Table Operations

Overview
As a State Machine table is a variant of a UML State Machine diagram, most of the operations for manipulating the data are the same as for State Machine diagrams. These include operations to:

- Create new items by drag-and-dropping a specified object from the Enterprise Architect UML Toolbox to the current diagram
- Delete an item
- Apply to the diagram elements in the Project Browser
- Display or change the properties of the State, Trigger or Transition
- Apply to the diagram, such as Lock Diagram, Zoom, and in place editing of the element.

The operations specific to State Machine tables are described in the following topics:

- Change Position of State Machine Table
- Change Size of State Machine Table
- Insert New State (and Substate)
- Insert Trigger
15.1.1.4.2.1 Change State Machine Table Position

If necessary, you can move the State Machine table around in the Diagram View. To change the position of the State Machine table, follow the steps below:

1. Press [Ctrl]+[A] or double click on the top left cell to select the whole State Machine table.
2. Drag and drop the State Machine table to the required position. Alternatively, use [Shift]+[→], [←], [↑] or [↓] to move the State Machine table.

15.1.1.4.2.2 Change State Machine Table Size

There are three ways to change the size of the State Machine table:

- Change the cell size on the State Machine Diagram: Options dialog.
- Press [Ctrl]+[A] or double click on the top left cell to select the whole State Machine table, then press [Ctrl]+[→], [←], [↑] or [↓] to change the size.
- Select the State Machine table, then drag the shape handles to change the size.

15.1.1.4.2.3 Insert New State

You can insert a new State in the State Machine table, using one of the following methods:

- In the top left cell in the State Machine table, move the cursor to the word State to display a + at the end of the word; click on the + to create a new State.
- Right-click in the top left cell in the State Machine table to display the context menu, and select the Add State menu option.
- Right-click on an existing State cell in the State Machine table to display the context menu, and select the
  - Insert New State Before option to insert a new State before the current State, or
  - Insert New State After option to insert a new State after the current State.
- Click on an existing State cell in the State Machine table, and press [Insert] to create and insert a new State above the selected State.
- In the Enterprise Architect UML Toolbox, on the State Elements page, click on an element and then click on:
  - the diagram background to add a new State to the end of the table, or
  - an existing State cell to add the new State just above it.

Note:

From the State Elements page of the Enterprise Architect UML Toolbox you can insert State, Initial, Final, Entry, Exit and Terminate elements.

Add a Substate

To add a Substate to a selected State, follow the steps below:

1. Right-click on the required State cell in the State Machine table. The context menu displays.
2. Select the Add Substate menu option. Enterprise Architect adds the Substate to the State.

Note:

If the selected State does not allow a Substate, then the Add Substate menu option is grayed out.

You can also drag one existing State over another. If the second State allows Substates, the dragged State then becomes its Substate.

Similarly, you can change the parent State of a Substate by dragging the Substate from the original parent State to a different State.
Remove Parent Relation of a Substate

To remove the parent relation of a Substate and make it a separate State, follow the steps below:

1. Right-click on the Substate in the State Machine table. The context menu displays.
2. Select the Remove Parent Relation menu option. The Substate cell becomes a State cell.

You can also drag and drop the Substate onto the top left cell of the State Machine table. The dragged Substate again becomes a State cell.

15.1.1.4.2.4 Insert Trigger

If the State Machine table format is either State-Trigger or Trigger-State, you can use one of the following methods to insert a new Trigger:

- In the top left cell in the State Machine table, move the cursor to the word Event to display a + at the end of the word; click on the + to create a new Trigger
- In the top left cell in the State Machine table, right-click to display the context menu and select the Add Trigger menu option to create a new Trigger
- Select an existing Trigger in the State Machine table, then press [Insert] to insert a new Trigger before the existing Trigger
- Click on an existing Trigger in the State Machine table, right-click to display the context menu and select either the:
  - Insert New Trigger Before option to insert a new Trigger before the current Trigger, or
  - Insert New Trigger After option to insert a new Trigger after the current Trigger.

15.1.1.4.2.5 Insert/Change Transition

You can insert a new Transition using one of the following methods:

- Right-click on the cell in which to create a Transition, to display the context menu
  - If the State Machine table format is State-Trigger or Trigger-State, the context menu lists the States you can choose as the target of the Transition; click on the required State name to create the Transition
  - If the State Machine table format is State-Next State, click on the Insert Transition menu option to create the Transition.
- In the State Relationships page of the Enterprise Architect UML Toolbox, select the Transition element, then click on the cell in the State Machine table in which to create the Transition. Double-click on the Transition to define it in the Transition Properties dialog.

Change the Transition

As for the State Chart diagram, to change the properties of a Transition double-click the Transition cell and edit the details on the Transition Properties dialog.

Change Transition States

You can change the source and target of the Transition by right-clicking the Transition and selecting the Advanced | Set Source and Target context menu option.

Alternatively, you can change the Transition source, target or Trigger by clicking on the Transition and dragging it to a different cell.

If the State Machine table format is either State-Trigger or Trigger-State, you can change the target state of a transition by:

1. Highlighting the target state name in the Transition cell and clicking on it to display a list of the states in the table.
2. Clicking on the preferred target state name.

Highlight States and Trigger Related to Transition

You can select options to highlight the source State, target State and Trigger cells associated with a Transition, using the Highlight Options panel on the State Machine Diagram Options dialog. When you click on the Transition cell its associated State and Trigger cells are highlighted.

Alternatively, click on the Transition cell and press and hold [L].
15.1.1.4.2.6 Reposition State or Trigger Cells

You can change the position of a selected State or Trigger cell in one of the following ways:

- Right-click on the State or Trigger title cell and select the appropriate **Order | Move xxx** context menu option.
- Click on the cell and press `[Shift]+[→], [←], [↑] or [↓]`.

15.1.1.4.2.7 Add Legend

You can add a simple legend to any State Machine Table cell that has no transition. The two legend symbols are:

- **I** - Ignore
- **N** - Never Happen

To assign a legend symbol to a State Machine Table cell, follow the steps below:

1. Right-click on the cell to which to assign the legend. The context menu displays.
2. Select the appropriate menu option:
   - **Legend | Ignore**
   - **Legend | Never Happen**.
   The required symbol displays in the center of the cell.

To remove a legend symbol from a cell, right-click on the cell and select the **Legend | Remove Legend** context menu option.

15.1.1.4.2.8 Find Cell in State Machine Diagram

On the State Machine table you can select a State or Trigger element and locate it in a State Machine diagram, by selecting the **Find | Locate in State Chart** context menu option. Enterprise Architect switches to the State Machine diagram and highlights the selected element. You can locate a Transition relationship in a similar way, by selecting the **Locate in State Chart** context menu option.

**Note:**

A Trigger on a State Machine table might or might not exist on the corresponding State Machine diagram. If the Trigger does not exist on the State Machine diagram, the **Locate in State Chart** option is disabled.

Conversely, on the State Machine diagram, you can select a State or Trigger element and locate it on the corresponding State Machine table, by selecting the **Find | Locate in State Table** context menu option. Enterprise Architect switches to the State Machine table and highlights the selected element. You can locate a Transition relationship in a similar way, by selecting the **Locate in State Table** context menu option.

15.1.1.4.2.9 State Machine Table Conventions

**Trigger**

- Deleting a Trigger removes it completely from the model, therefore you cannot UNDO a Trigger deletion.
- There is a `<None>` column at the end of the **Event** heading row. This is for Transitions that have no Trigger information.

**State**

From the Enterprise Architect UML Toolbox you can insert the following **State** element types only (although the State Machine table might pick up and display other types, such as **Submachine State**):

- State
- Initial
- Final
- Entry
- Exit
- Terminate.
**Transition**

The Transition cell displays its properties in one of two ways, depending on the State Machine table format. If the State Machine table format is **State - Trigger** or **Trigger - State**, the Transition cell displays the **Guard** and **Target** as shown below:

![Transition cell example](image)

If the State Machine table format is **State - Next State**, then the Transition cell displays the **Trigger**, **Guard** and **Effect** as shown below:

![Transition cell example](image)

The State Machine table enables you to edit the **Guard** and **Effect** in place. If the **Guard** or **Effect** is empty for your selected Transition cell, the cell displays an ellipsis [ ... ] instead. Click twice (not double-click) on the ellipsis to type in the **Guard** and **Effect** names.

**15.1.1.4.2.10 Export State Table To CSV File**

To export a State Machine Table to a CSV file, follow the steps below:

1. Open the required State Machine Table.
2. Right-click on the diagram background and select the **Export Statechart to CSV file** context menu option.
3. The Save As browser dialog displays. Select the appropriate directory location and type in the .CSV filename.
4. Click on the **Save** button.

**15.1.1.5 Timing Diagram**

One of four types of **Interaction** diagram. (The other three are **Sequence Diagrams**, **Interaction Overview Diagrams** and **Communication Diagrams**.)

A **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.

You can use Timing diagrams to define hardware-driven or embedded software components; for example, those used in a fuel injection system or a microwave controller. You can also use Timing diagrams to specify time-driven business processes.

To create and edit a Timing diagram, see the following topics:

- **Create a Timing Diagram**
- **Set a Time Range**
- **State Lifeline**
- **Value Lifeline**
- **Edit a Timing Diagram**
- **Time Intervals**
- **Message (Timing Diagram)**

**Example Diagram**

An example of a Timing diagram is shown below:
Toolbox Elements and Message

Select Timing diagram elements and connectors from the Timing pages of the Enterprise Architect UML Toolbox.

Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Timing Diagram Elements</th>
<th>Timing Diagram Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Lifeline</td>
<td>Message</td>
</tr>
<tr>
<td>Value Lifeline</td>
<td></td>
</tr>
<tr>
<td>Message Label</td>
<td></td>
</tr>
<tr>
<td>Message Endpoint</td>
<td></td>
</tr>
<tr>
<td>Diagram Gate</td>
<td></td>
</tr>
</tbody>
</table>

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 517) 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 Superstructure Specification, v2.1.1, p. 519) 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.

15.1.1.5.1 Create a Timing Diagram

To create a Timing diagram, follow the steps below:

1. Right-click on a package in the Project Browser. The context menu displays.
2. Select the Add | Add Diagram menu option. The New Diagram dialog displays.
3. In the Select From panel, select UML Behavioral.
4. In the Diagram Types panel, select Timing.
5. Click on the OK button. The Diagram view displays, on which you create the Timing elements for the diagram. See Set a Time Range and Edit a Timing Diagram.

15.1.1.5.2 Set a Time Range

Before adding Lifeline elements to your Timing diagram, set a time range. To do this, follow the steps below:

1. Right-click on the diagram. The context menu displays.
2. Select the Set Timeline Range option. The Set Timeline Range dialog displays.
3. In the Start Time and End Time fields, type the numeric values for the start and end points of the timeline; for example, set the range 0 to 100.

**Note:**

The start time must be less than the end time.

4. In the Time Units field, type the unit in which the time is measured; for example, seconds or minutes.
5. If it is not necessary to show the time range on the diagram, select the Suppress In Diagram checkbox.
6. Click on the OK button. If you have not suppressed it, the time range displays underneath the Lifeline elements that you create on the diagram.

15.1.1.5.3 Edit a Timing Diagram

On a Timing Diagram, you can add State Lifeline elements and Value Lifeline elements. You can maintain the states and transitions on these Lifeline elements either on the diagram itself or via the Configure Timeline dialog. See the following topics:

- Add and Edit a State Lifeline Element
- Edit States in a State Lifeline Element
- Edit Transitions in a State Lifeline Element
- Add and Edit a Value Lifeline Element
- Add States in a Value Lifeline Element
- Edit Transitions in a Value Lifeline Element
- Configure Timeline dialog - States Tab
- Configure Timeline dialog - Transitions Tab
15.1.1.5.3.1 Add and Edit State Lifeline

From the Timing elements page of the Enterprise Architect UML Toolbox drag a State Lifeline element onto your diagram. The element displays on the diagram.

To define the name of the State Lifeline, follow the steps below:
1. Right-click on the element. The context menu displays.
2. Select the Other Properties option. The Timeline <name> dialog displays, showing the General tab.
3. Overtype the Name field.
4. Click on the Apply button and the OK button.

Sizing and Scale

In the top left corner of a selected Lifeline element are the left and right quick sizing buttons. These buttons increase or decrease the width of the Lifeline element, which in turn controls the scale width of each time unit. By increasing the width of the element you increase the resolution when adding transitions, which makes them easier to edit.

Note:
In order to edit the State Lifeline element, you must click on it to select it.

Set Timeline Start Position

You might require more space at the start of your timelines; for example, to use long state names. To insert this space in all the timelines on a diagram, follow the steps below:
1. Right-click on the diagram background to display the context menu.
2. Select the Set Timeline Start Position menu option. The Set Timeline Start Position dialog displays.
3. The Value 80 to 300 field defaults to 80 as the minimum distance in pixels between the start of the timeline element and the start of the timeline itself. Type a new value up to 300 pixels and click on the OK button to increase the space at the start of the timeline, as shown in the following diagrams.

You now edit the states and transitions in the State Lifeline.

15.1.1.5.3.2 Edit States in State Lifeline

Add States
1. Click on the State Lifeline element. The New State button and Edit States button display at the bottom left of the element.
2. Click on the New State button. The New State dialog displays.
3. In the State field, type the name of the state.
4. Click on the OK button.

**Note:**
You must add at least two states; for example, On and Off.

5. As you add states, increase the height of the element by dragging a handle-box (→) on the edge of the element.

**Note:**
You can also add states using the States tab of the Configure Timeline dialog. Add either:
- Discrete states to the Timeline as described in Add a New State or
- A continuous range of numeric states as described in Numeric Range Generator.

### Edit States
1. Click on the State Lifeline element and click on the required state. The Edit State dialog displays.
2. In the State field, change the name as required.
3. Click on the OK button.
4. If necessary, change the order of the states by either:
   - Clicking on the up or down arrows (↑ and ↓) beside each state name, or
   - Right-clicking on the state name and selecting the Move Up or Move Down context menu options.

**Note:**
You can also edit the states using the States tab of the Configure Timeline dialog.

### Delete States
1. Right-click on the state name. The context menu displays.
2. Select the Delete option.
   Alternatively:
   1. Click on the State Lifeline element.
   2. Hold down [Ctrl] and move the cursor over the state name. The cursor changes form (↑).
   3. Click the mouse button. The state name is deleted.

### 15.1.1.5.3.3 Edit Transitions In State Lifeline

#### Add and Move Transitions
After you have added states, you can add transitions via the diagram. As you move the cursor over the timeline, the cursor changes to one of three shapes:

- The move cursor (↑) displays when it is directly over the timeline. Hold down the mouse button and drag the line to move the timeline to a state above or below the current position. You can move the transition more than one state up or down, if necessary.

- The new transition up cursor (↑) displays when it is just below the timeline, and there is another state above the line. Press and hold [Alt]; the cursor changes (↑). Click to create a new transition to the state above the line. To push the transition up more than one state, then move the cursor onto the line and drag...
it up. The transition is for one interval unit; to make it longer, see *Change the Transition Time* below.

If you do not hold [Alt], the cursor does not change and the whole timeline from the transition onwards moves up.

- The *new transition down* cursor (📉) displays when it is just above the transition line, and there is another state below the line. Press and hold [Alt]; the cursor changes (📉). Click to create a new transition to the state below the line. To push the transition down more than one state, then move the cursor onto the line and drag it down. The transition is for one interval unit; to make it longer, see *Change the Transition Time* below.

If you do not hold [Alt], the cursor does not change and the whole timeline from the transition onwards moves down.

As you move the cursor over the vertical line of a transition, the time at which the transition occurs displays next to the line.

**Edit Transitions**

Follow the steps below:

1. Click directly on the appropriate transition line, after the transition begins. Alternatively, right-click on the transition line to display the context menu, and select the **Edit** menu option.

   The *Edit Transition* dialog displays. The fields in this dialog are all optional.

```
Edit Transition dialog

- At Time: 13
- Transition To: Full
- Event:
- Duration Constraint:
- Time Constraint:

OK  Cancel
```

2. In the **At Time** field, type the point on the timescale at which the transition occurs.
3. In the **Transition To** field, type the name of the state to which the transition occurs.
4. In the **Event** field, type the name of the event that the transition represents; this displays on the Timeline element just above the transition line.
5. In the **Duration Constraint** field, type any constraint on the duration of the transition; this displays on the Timeline element, along the top of the element over the transition.
6. In the **Time Constraint** field, type any constraint on the start of the transition. This displays on the Timeline element at the start of the transition.
7. Click on the **OK** button.

**Notes:**

- *Once Event, Duration Constraint* or *Time Constraint* are displayed on the diagram, you can edit them directly by clicking on them to display their specific dialog. You can also delete them by pressing and holding [Ctrl] as you click on them; the cursor changes form when you press [Ctrl].

- You can also edit transitions using the **Transitions** tab of the *Configure Timeline* dialog.

**Change the Transition Time**

Move the cursor over one or other of the vertical transition lines and drag the line left or right to change the time of the transition. While on the line, the cursor shape changes to the horizontal movement cursor (↔).
Merge Transitions
If necessary, you can 'push' a transition to merge it with the next or previous transition point on any Lifeline element on the diagram.

Position the cursor off the appropriate side of the transition line; the cursor changes form (← or →). Click the mouse button. The system locates the nearest transition in the required direction, on any element on the diagram, and merges the current transition with that transition.

Delete Transitions
Transitions are automatically deleted when you move the transition to the same state as the previous transition state, and release the cursor.
Alternatively, right-click on the transition line to display the context menu, and select the Delete menu option.

15.1.1.5.3.4 Add and Edit Value Lifeline
From the Enterprise Architect UML Toolbox drag a Value Lifeline element onto your diagram. The element displays on the diagram.

To edit the Value Lifeline name, follow the steps below:
1. Right-click on the element. The context menu displays.
2. Select the Other Properties option. The Timeline <name> dialog displays, showing the General tab.
3. Overtype the Name field.
4. Click on the Apply button and the OK button.

Sizing and Scale
In the top left corner of a selected Lifeline element are the left and right quick sizing buttons (← →). These buttons increase or decrease the width of the Lifeline element, which in turn controls the scale width of each time unit. By increasing the width of the element you increase the resolution when adding transitions, which makes them easier to edit.

Note:
In order to edit the Value Lifeline element, you must click on it to select it.

You now add states and edit transitions on the Value Lifeline.

15.1.1.5.3.5 Add States In Value Lifeline
Add States
This is similar to adding states to a State Lifeline element.

Notes:
- For a Value Lifeline, only the first state displays on the diagram. The other states are added to a list to access when creating transitions; they only display on the Lifeline element as you create transitions to those states.
- You can only edit or delete states in a Value Lifeline element using the States tab of the Configure Timeline dialog.

15.1.1.5.3.6 Edit Transitions In Value Lifeline
Add Transitions
After you have added states to the Value Lifeline element, you can add transitions via the diagram. To do this, follow the steps below:
1. Move the cursor above the transition line. The cursor changes form ( ).
2. Click the mouse button. The New Transition Event dialog displays.

![New Transition Event Dialog]

3. In the Transition To field, click on the drop-down arrow and select a state from the list of available states; this displays on the Lifeline element within the transition box. The remaining fields on the dialog are optional.

4. In the Event field, type the name of the event that the transition represents; this displays on the Lifeline element just below and at the start of the transition line.

5. In the Duration Constraint field, type any constraint on the duration of the transition; this displays on the Lifeline element, along the top of the element over the transition.

6. In the Time Constraint field, type any constraint on the start of the transition. This displays on the Lifeline element at the start of the transition, just after the Event name.

7. Click on the OK button to create the new transition.

**Edit Transitions**

To edit a transition, follow the steps below:

1. Click on the state name in the transition. Alternatively, right-click on the state name to display the context menu, and select the Edit menu option.

   The Edit Transition dialog displays; this is the same as the New Transition Event dialog, except that the At Time field is enabled.

2. If necessary, overtype the At Time field to define a different start point.

   **Note:**
   You cannot change the At Time field for the first state in the timeline; this is always 0.

3. Edit the remaining fields as necessary.

4. Click on the OK button to save the changes.

**Change the Transition Time**

To change the start or end time of a transition, click on the start or end point of the transition and drag it to the new position. While on the line, the cursor shape changes to the horizontal movement cursor (←→).

**Delete Transitions**

To delete a transition, press and hold [Ctrl] and click on the transition state name. While you hold [Ctrl] on the transition state name, the cursor changes form (△).

Alternatively, right-click on the state name to display the context menu, and select the Delete menu option.
15.1.1.5.3.7 Configure Timeline - States

You can also manage states using the States tab of the Configure Timeline dialog. To display this, either:

- Double-click on the Lifeline element
- Right click on the Lifeline element and, from the context menu, select the Properties option, or
- On a Value Lifeline, click on the Edit States button ( ).

The Configure Timeline dialog defaults to the States tab. All states currently defined for the Lifeline element are listed in the States panel.

Add a New State:
1. In the State Name field, type the name of the first new state in the Lifeline element; for example, WaitState.
2. Click on the Save button. The state is added to the States panel and (for a State Lifeline Element) to the diagram.
3. Click on the New button.
4. In the State Name field, type the name of the next state in the Lifeline element.
5. Repeat steps 2 to 5 until you have added all required states (you must add at least three to the Lifeline element).
6. When you have added all the required states, click on the OK button to close the Configure Timeline dialog.

**Edit an Existing State:**
1. Click on the state in the States: list.
2. In the State Name field, change the name of the state.
3. Click on the Save button.

**Delete an Existing State:**
1. Click on the state in the States: list.
2. Click on the Delete button.

**Change the Order of States:**
1. Click on the state in the States: list.
2. Click on the or buttons to move the state up or down the sequence.

**Numeric Range Generator**
You can also use the Configure Timeline dialog to create a range of states having numeric values to be applied to the Timeline.

**Important:**
This operation deletes all existing states and transitions for the Timeline element.

1. Display the Configure Timeline dialog.
2. Click on the Create Continuous Numeric States button. The Numeric Range Generator dialog displays.
3. In the High Value and Low Value fields, type the upper and lower values of the range.
4. In the Step Value field, type the increase interval.

**Note:**
Nonsense values do not parse; Low Value must be less than High Value, and Step Value must be a positive value smaller than the total range.

5. In the Units field, type the name of the measurement unit; for example, minutes.
6. Click on the OK button. Enterprise Architect displays a warning that existing states and transitions will be deleted.
7. Click on the Yes button. The Configure Timeline dialog redispays, with the defined range of states listed in the States panel.
8. Click on the OK button. For a:
   - Value Lifeline, the first state is shown on the Timeline for the full time range of the Timeline.
   - State Lifeline, the range of states is displayed as the y-axis of the Timeline.

**15.1.1.5.3.8 Configure Timeline - Transitions**
You can also manage transitions using the Transitions tab of the Configure Timeline dialog. To display this, either:

- Double-click on the Lifeline element
- Right click on the Lifeline element and, from the context menu, select the Properties option, or
- On a Value Lifeline, click on the Edit States button ( ).

The Configure Timeline dialog defaults to the States tab. Click on the Transitions tab.
All transitions defined for the Timeline element are listed in the Transition Points panel.

**Add a New Transition**
1. Click on the **New** button.
2. In the New Transition panel, type the details of the transition.
3. Click on the **Save** button.

**Edit a Transition**
1. Click on a transition in the list.
2. In the Edit Transition panel, edit the fields for the transition as required.
3. Click on the **Save** button.

**Delete a Transition**
1. Click on a transition in the list.
2. Click on the **Delete** button. The transition is removed from the dialog and the Lifeline.
3. Click on the **OK** button.
15.1.1.5.4 Time Intervals

You create and manage Time Intervals using the Interval Bar (the pale line along the top of each selected Lifeline element). Time Intervals enable you to perform various operations on transitions, such as copy and paste. They also enable you to compress sections of the timeline so that they are not visible.

Each Time Interval displays across all Timeline elements down to the last element on the diagram.

Create Time Intervals

To create a Time Interval, follow one of the three sets of steps below:

Interval Bar - Context Menu

1. Right-click on the Interval Bar at approximately the point at which to start or finish the Time Interval.

2. Select the Create Time Interval option. The Time Interval displays down all the timeline elements, as a narrow pale band with a blue compression box at the top.

3. Move the cursor to the edge of the Time Interval in the Interval Bar so that the cursor changes to the
drag form (↕️️️) and drag the edge to the correct start or end point.

**Interval Bar - [Shift] key**
1. Move the cursor over the Interval Bar and press [Shift]. The cursor changes shape (↕️️️).
2. Click to create the Time Interval.
3. Move the cursor to the edge of the Time Interval in the Interval Bar so that the cursor changes to the drag form (↕️️️) and drag the edge to the correct start or end point.

**Timeline - Context menu**
1. Right-click on the timeline just after a transition. The context menu displays.
2. Click on the Select menu option. Enterprise Architect creates a Time Interval covering the period from the selected transition up to the next transition.

**Notes:**
- If there are other Time Intervals in this period, Enterprise Architect replaces them with the single Time Interval for the transition state. You should consider this when creating the Time Interval, as it extends across the other Timeline Elements in the diagram.
- A value of this method is that it creates a Time Interval for a period in which no transitions occur, which could be lengthy. You can then compress this Time Interval (see below) to hide the period of inactivity. See also Compress Timeline.

**Compress Time Intervals**
You can compress Time Intervals to conserve space on long timelines.

**Uncompressed Time Intervals**
Compressed Time Intervals

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![Expand/Compress Icons] | The compression toggle boxes:  
- ![](Expand) is expanded, click on this to compress the selected time interval  
- ![](Compress) is compressed, click on this to expand the selected time interval again. |
| ![Timeline Section] | The compressed sections of the timelines themselves, in all elements.  
If there is space between the paired symbols, there are transitions within the compressed section.  
If the timeline continues through the paired symbols there are no transitions in the compressed section. |
| 25 ... 55 | The compressed sections in the time range underneath the elements. |

Notice:

You can also compress and expand Time Intervals using context menu options; see Time Interval Operations on Transitions.

Select Time Intervals

- To select a Time Interval across all elements on the diagram, click on the Interval Bar within the Time Interval.
- To select a number of individual Time Intervals, press and hold [Ctrl] while clicking on the Interval Bar within each Time Interval.
- To select all Time Intervals in a range, click on the Interval Bar within the first Time Interval in the range, then press and hold [Shift] and click on the Interval Bar within the last Time Interval in the range. All Time Intervals between the two are selected.

After you have selected one or more Time Intervals, you can modify the selection in the following ways:

- To exclude Lifeline elements from the selection, press and hold [Ctrl] and click on any part of the selection within that element. In the diagram below, the Value Lifeline is excluded from selection.
Repeat the step to toggle the selection and re-include the element. See also Toggle Interval Selection.

- To select only one Lifeline element and exclude all others, press and hold [Shift] and click on any part of the selection within that element.

**Note:**
Selection is useful for cutting, copying and pasting transitions.

### Move and Resize Time Intervals

To move a Time Interval, move the cursor over the Interval bar within the Time Interval, hold down the mouse button and drag the interval left or right.

To resize a Time Interval, move the cursor over the Interval Bar at the start or end edge of the Time Interval, hold down the mouse button and move the edge left or right.

**Note:**
Time Intervals can meet, but cannot overlap.

### Delete Time Intervals

Select each Time Interval to be deleted and press [Delete].

**Note:**
Deleting the Time Interval does not delete transitions within that interval.

#### 15.1.1.5.4.1 Time Interval Operations

You can operate on selected Time Intervals or all Time Intervals in the diagram.

### Selected Intervals

**Note:**
The Copy, Cut and Delete operations act on all selected Time Intervals over the whole diagram, not just the current one.

To select and update specific Time Intervals, right-click on the Interval Bar within an interval. The following context menu displays.
### Option

<table>
<thead>
<tr>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select Interval</strong></td>
</tr>
<tr>
<td><strong>Deselect Interval</strong></td>
</tr>
<tr>
<td><strong>Toggle Interval Selection</strong></td>
</tr>
<tr>
<td><strong>Compress Interval</strong></td>
</tr>
<tr>
<td><strong>Remove Interval</strong></td>
</tr>
<tr>
<td><strong>Copy</strong></td>
</tr>
<tr>
<td><strong>Cut</strong></td>
</tr>
<tr>
<td><strong>Cut and Remove Time</strong></td>
</tr>
<tr>
<td><strong>Delete</strong></td>
</tr>
<tr>
<td><strong>Delete and Remove Time</strong></td>
</tr>
<tr>
<td><strong>Insert Time</strong></td>
</tr>
</tbody>
</table>

### Compress Timeline

The Compression toggle boxes and **Compress Interval** menu option operate on the Time Interval and compress the timeline and all transitions within the Interval. You have an alternative option that operates on the timeline and compresses a single transition state.

1. Right-click on the timeline (rather than the Interval Bar) just after a transition. The context menu displays.
2. Click on the **Compress** menu option. Enterprise Architect creates a new Time Interval covering the...
period from the selected transition up to the next transition, and then compresses that Time Interval.

**Notes:**

- If there are other Time Intervals in this period, Enterprise Architect replaces them with the single Time Interval for the transition state. You should consider this when creating and compressing the Time Interval, as it extends across the other Timeline elements in the diagram.
- A value of this method is that it creates a Time Interval for a period in which no transitions occur, which could be lengthy, and then compresses this Time Interval to hide the period of inactivity.

### All Time Intervals in the Diagram

To create a new Time Interval or work across all Time Intervals in the diagram, right-click on the Interval Bar between Time Intervals. The following context menu displays.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Time Interval</td>
<td>Create a single Time Interval</td>
</tr>
<tr>
<td>Expand all Time Intervals</td>
<td>Expand all Time Intervals over the whole diagram.</td>
</tr>
<tr>
<td>Compress all Time Intervals</td>
<td>Compress all Time Intervals over the whole diagram.</td>
</tr>
<tr>
<td>Paste Combine</td>
<td>Paste copied transitions over any existing transitions within the copied time frame.</td>
</tr>
<tr>
<td>Paste Remove</td>
<td>Delete all the transitions and then pastes the copied transition within the copied time frame.</td>
</tr>
<tr>
<td>Paste Insert</td>
<td>Insert time, moving all transitions and Time Intervals to the right to make room to paste in the copied transitions.</td>
</tr>
<tr>
<td>Insert Time</td>
<td>Add time to the timeline and move all transitions and Time Intervals to the right. This option does not change the duration of any Time Interval.</td>
</tr>
</tbody>
</table>

**Note:**

The **Paste** menu options become active after transitions have been copied.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Time Interval</td>
<td>Create a single Time Interval</td>
</tr>
<tr>
<td>Expand all Time Intervals</td>
<td>Expand all Time Intervals over the whole diagram.</td>
</tr>
<tr>
<td>Compress all Time Intervals</td>
<td>Compress all Time Intervals over the whole diagram.</td>
</tr>
<tr>
<td>Paste Combine</td>
<td>Paste copied transitions over any existing transitions within the copied time frame.</td>
</tr>
<tr>
<td>Note:</td>
<td>The diagram does not allow two consecutive transitions to the same state, and removes the second transition automatically.</td>
</tr>
<tr>
<td>Paste Remove</td>
<td>Delete all the transitions and then pastes the copied transition within the copied time frame.</td>
</tr>
<tr>
<td>Paste Insert</td>
<td>Insert time, moving all transitions and Time Intervals to the right to make room to paste in the copied transitions.</td>
</tr>
<tr>
<td>Insert Time</td>
<td>Add time to the timeline and move all transitions and Time Intervals to the right. This option does not change the duration of any Time Interval.</td>
</tr>
</tbody>
</table>

### Copy and Paste Transitions From One Timeline Element to Another

A special mode enables you to copy transitions from one Timeline element to another. Any states that don’t exist in the Timeline element you are pasting to are created.

1. Press and hold `[Shift]` and select the Timeline element within a Time Interval to copy or cut.
2. Right-click on the Interval Bar (it doesn't matter which element you select). The context menu displays.
3. Copy or cut the transitions. You can also cut and remove time.
4. Select the timeline to paste transitions to and right-click on the Interval Bar. The context menu displays.
5. Select one of the paste operations. Note that states are created if they don't already exist in the timeline.

Note:
Any new states created might be in the wrong order. You can change the order via the diagram quick buttons.

Shift Transitions Left or Right
You can move transitions within a selected Time Interval or multiple selected Time Intervals.

1. Select all the Time Intervals containing the transitions to be shifted; see Select Time Intervals.
2. Press and hold [Shift] and click on the Interval Bar (it doesn't matter which Timeline element you select) and move the transition left or right.

Note:
You cannot drag transitions over other transitions; the move stops when the moved transition collides with a stationary transition.

Tip:
If having collision problems, use [Shift]+select to shift transitions for a single Timeline element.

15.1.1.6 Sequence Diagram
One of four types of Interaction diagram. (The other three are Timing Diagrams, Interaction Overview Diagrams and Communication Diagrams.)

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 can be used to represent the user initiating the flow of events.
- Stereotyped elements, such as Boundary, Control and Entity, can 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.

To configure a Sequence diagram, see the following topics:
- Denote the Lifecycle of an Element
- Layout of Sequence Diagrams
- Sequence Element Activation
- Lifeline Activation Levels
- Message Label Visibility
- Change the Top Margin
- Change the Timing Details

Robustness diagrams, used extensively in the ICONIX Process, can be created as Sequence diagrams.

Example Diagram
The following example Sequence diagram demonstrates several different elements:
Toolbox Elements and Connectors

Select Sequence diagram elements and connectors from the Interaction pages of the Enterprise Architect UML Toolbox.

Enterprise Architect also supports a number of stereotyped elements to represent various entities in business modeling.

Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Sequence Diagram Elements</th>
<th>Sequence Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>→ Message</td>
</tr>
<tr>
<td>Lifeline</td>
<td>▼ Self-Message</td>
</tr>
<tr>
<td>Boundary</td>
<td>▼ Recursion</td>
</tr>
<tr>
<td>Control</td>
<td>▼ Call</td>
</tr>
<tr>
<td>Entity</td>
<td></td>
</tr>
<tr>
<td>Fragment</td>
<td></td>
</tr>
</tbody>
</table>
OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 503) states:

A sequence diagram describes an Interaction by focusing on the sequence of Messages that are exchanged, along with their corresponding Occurrence Specifications on the Lifelines.

15.1.1.6.1 Denote Lifecycle of an Element

You can capture element lifetimes using messages that are denoted as New or Delete message types. To do this, follow the steps below:

1. Double-click on a message within a Sequence diagram to display the Message Properties dialog.
2. In the Lifecycle field, click on the drop-down arrow and select New or Delete.
3. Click on the OK button to save the 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 following example diagram, you must switch on garbage collection: Tools | Options | Diagram | Sequence | Garbage Collect.

15.1.1.6.2 Layout of Sequence Diagrams

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, Enterprise Architect interprets that as the requirement to swap positions, rather than simply offset a message position.

The example below shows an economical use of space in a Sequence diagram.

![Sequence Diagram Example](image)

### 15.1.1.6.3 Sequence Elements

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 a collaborating element.

**Note:**

A Sequence diagram is often attached directly under the 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 the **Advanced | Make Composite** menu option.

The example below shows some possible elements of Sequence diagrams and their stereotyped display.

- **Actor** - An instance of an actor at runtime.
- **Lifeline** - An Object element with the stereotype Lifeline.
- **Boundary** - Represents a user interface screen or input/output device.
- **Entity** - A persistent element - typically implemented as a database table or element.
- **Control** - 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 eventually become method calls in the Class model.

15.1.1.6.4 Sequence Element Activation

Sequence elements in a Sequence diagram have Activation rectangles drawn along their lifelines. These rectangles describe 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 the Suppress Activations menu option.

In general, Enterprise Architect calculates the period of activation for you, but in some cases you might want 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 menu option.

- **Start New Message Group**: Starts off a new round of processing in the current diagram. This enables you to describe more than one processing scenario in a single diagram.
- **Extend Source Activation Down**: Forces an element to stay active beyond the normal processing period. This could be used to express an element that continues its own processing concurrently with other processes.
- **Extend Source Activation Up**: Forces an element's activation upwards.
- **End Source Activation**: Truncates the activation of the source element after the current message. This is useful for expressing an asynchronous message after which the source element becomes idle.
- **End Target Activation**: Ends a Forced Activation started by the Extend Source Activation options.

The Raise Activation Level and Lower Activation Level options display on the context menu only where their use is appropriate. For example, after a self-message the next message starts by default at a lower
activation level but the **Raise Activation Level** command displays on the context menu to enable you to raise its level.

A more convenient way to change activation levels is directly on the diagram. Whenever appropriate, left and/or right arrows display on specific connectors. In the following diagram, see connector 1.3. Click on the arrow to raise or lower the activation level.

![Diagram showing activation levels](image)

**Note:**
Program flow can more accurately be depicted with nested activation levels for callback messages.

### 15.1.1.6.5 Lifeline Activation Levels

Complicated processing systems can be easily negotiated and reflected in Sequence diagrams, by adding activation layers on a single lifeline. For example, a Class invokes the method *Sample A*, which in turn calls *Sample A1*. To produce the arrangement in the diagram, select the **More tools** | **UML** | **Interaction** menu option, click on the **Self-message** icon in the **Interaction Relationships** panel and then click on the lifeline.

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.

15.1.1.6.6 Sequence Message Label Visibility

On Sequence messages, you can control label visibility using the message context menu. To hide and show the labels used in Sequence messages, follow the steps below:

1. Right-click on the message within the Sequence diagram. The message context menu displays.
2. Select the Set Label Visibility menu option. The Label Visibility dialog displays.
Clear check box to hide label

- Source bottom note
- Source top note
- Mid bottom note
- Mid top note: "Boot From CD (Windows 2..."
- Target bottom note
- Target top note

3. Select or clear the checkbox against each message label to display or hide, respectively.
4. Click on the OK button to save the settings.

15.1.1.6.7 Change the Top Margin

In order to change the top margin of a Sequence diagram from the default 50 units, right-click on the diagram to display the context menu and select the Set Top Margin menu option. You can set the top margin to any value between 30 and 250 units.

15.1.1.6.8 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, to display the context menu.
2. Select the Embedded Elements | Embedded Elements menu option.
3. Select the checkbox against each Part or Port to show, and click on the Close button.
15.1.1.7 Communication Diagram

One of four types of Interaction diagram. (The other three are Timing Diagrams, Sequence Diagrams and Interaction Overview Diagrams.)

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, and so on.

A new number segment begins for a new layer of processing, and would be equivalent to a method invocation.

Robustness diagrams are simplified Communication diagrams, but can be created in any diagram type that supports Boundary, Control and Entity elements.

Example Diagram

The example below illustrates a Communication diagram among cooperating object instances. Note the use of message levels to capture related flows, and the different colors of the messages.

Toolbox Elements and Connectors

Select Communication diagram elements and connectors from the Communication pages of the Enterprise Architect UML Toolbox.

Tip:

Click on the following elements and connectors for more information.
Communication Diagram Elements | Communication Diagram Connectors
--- | ---
Actor | Associate
Object | Nesting
Boundary | Realize
Control
Entity
Package

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 511) states:

*Communication Diagrams focus 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.*

*Communication Diagrams correspond to simple Sequence Diagrams that use none of the structuring mechanisms such as InteractionUses and CombinedFragments. It is also assumed that message overtaking (i.e., the order of the receptions are different from the order of sending of a given set of messages) will not take place or is irrelevant.*

**Note:**

Communication diagrams were known as Collaboration diagrams in UML 1.4.

**15.1.1.7.1 Communication Diagrams in Color**

Enterprise Architect enables 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. Select the Tools | Options | Communication Colors menu option. The Communication Message Coloring page of the Options dialog displays.
2. Select the Use Communication Color checkbox.
3. Click on the drop-down arrow of each Message field, and select the required color for each message group.
4. Click on the Close button. On your Communication diagram, each sequence group of messages displays in a different color as shown below.
15.1.1.8 Interaction Overview Diagram

One of four types of Interaction diagram. (The other three are Timing Diagrams, Sequence Diagrams and Communication Diagrams.)

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 the same, as is the process of 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 display an inline Interaction diagram, which can be any one of the four types
- **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 displays, encapsulating an instance of the Interaction diagram.

Example Diagram

The following example depicts a sample sale process, shown in an Interaction Overview diagram, with subprocesses 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, the respective interaction's process must be followed before the Interaction Overview's flow can advance.
Toolbox Elements and Connectors

Select Interaction Overview diagram elements and connectors from the Activity pages of the Enterprise Architect UML Toolbox.

Tip:
Click on the following elements and connectors for more information.
### Interaction Overview Diagram Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Fork/Join</td>
</tr>
<tr>
<td>Decision</td>
<td>Fork/Join</td>
</tr>
<tr>
<td>Send</td>
<td>Control Flow</td>
</tr>
<tr>
<td>Receive</td>
<td>Object Flow</td>
</tr>
<tr>
<td>Synch</td>
<td>Interrupt Flow</td>
</tr>
<tr>
<td>Initial</td>
<td></td>
</tr>
<tr>
<td>Final</td>
<td></td>
</tr>
<tr>
<td>Flow Final</td>
<td></td>
</tr>
<tr>
<td>Region</td>
<td></td>
</tr>
<tr>
<td>Exception</td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td></td>
</tr>
</tbody>
</table>

### OMG UML Specification

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 514*) 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 InteractionUses. The Lifelines and the Messages do not appear at this overview level.
15.1.2 Structural Diagrams

Structural diagrams depict the structural elements composing a system or function. These diagrams reflect the static relationships of a structure, such as Class or Package diagrams, or run-time architectures such as Object or Composite Structure diagrams.

Structural diagrams include the following diagram types:

Class Diagrams
Class diagrams capture the logical structure of the system, the Classes and objects that make up the model, describing what exists and what attributes and behavior it has.

Composite Structure Diagrams
Composite Structure diagrams reflect the internal collaboration of Classes, Interfaces and Components (and their properties) to describe a functionality.

Component Diagrams
Component diagrams illustrate the pieces of software, embedded controllers and such that make up a system, and their organization and dependencies.

Deployment Diagrams
Deployment diagrams show how and where the system is to be deployed; that is, its execution architecture.

Object Diagrams
Object diagrams depict object instances of Classes and their relationships at a point in time.

Package Diagrams
Package diagrams depict the organization of model elements into packages and the dependencies amongst them.

15.1.2.1 Package Diagram
Package diagrams depict the organization of model elements into packages and the dependencies amongst them, including package imports and package extensions. They also provide a visualization of the corresponding namespaces.

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 clicking on the diagram background to display the diagram's Properties dialog, selecting the Elements tab and selecting the Package Contents checkbox.

The «import» connector indicates that the elements within the target Integer package, which in this example is the single Class Integer, are imported into the package Controller. The Controller's namespace gains access to the Integer Class; the Integer namespace is not affected.

The «merge» connector indicates that the package Controller's elements are 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 are 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.

Notes:
- Private elements within a package cannot be imported or merged.
- If you click on an element listed in a package, and then double-click, you can display and edit the element properties.

Toolbox Elements and Connectors
Select Package diagram elements and connectors from the Class pages of the Enterprise Architect UML Toolbox.

Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Package Diagram Elements</th>
<th>Package Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Associate</td>
</tr>
<tr>
<td>Class</td>
<td>Generalize</td>
</tr>
<tr>
<td>Interface</td>
<td>Compose</td>
</tr>
<tr>
<td>Table</td>
<td>Aggregate</td>
</tr>
<tr>
<td>Enumeration</td>
<td>Association Class</td>
</tr>
</tbody>
</table>
15.1.2.2 Class Diagram

The Class diagram captures the logical structure of the system: the Classes - including Active and Parameterized (template) 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 pale Aggregation relationship indicates that the Class Account uses AddressBook, but does not necessarily contain AddressBook. The dark Composite Aggregation connectors indicate ownership or containment by the target Classes (at the diamond end) of the source Classes.

Toolbox Elements and Connectors

Select Class diagram elements and connectors from the Class pages of the Enterprise Architect UML Toolbox.

Enterprise Architect also supports a number of stereotyped Class elements to represent various entities in...
web page modeling.

**Tip:**
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Class Diagram Elements</th>
<th>Class Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Associate</td>
</tr>
<tr>
<td>Class</td>
<td>Generalize</td>
</tr>
<tr>
<td>Interface</td>
<td>Compose</td>
</tr>
<tr>
<td>Table</td>
<td>Aggregate</td>
</tr>
<tr>
<td>Enumeration</td>
<td>Association Class</td>
</tr>
<tr>
<td>Signal</td>
<td>Assembly</td>
</tr>
<tr>
<td>Association</td>
<td>Realize</td>
</tr>
<tr>
<td></td>
<td>Nesting</td>
</tr>
<tr>
<td></td>
<td>Package Merge</td>
</tr>
<tr>
<td></td>
<td>Package Import</td>
</tr>
</tbody>
</table>

### 15.1.2.3 Object Diagram

An **Object diagram** is closely related to a **Class diagram**, with the distinction that it depicts object instances of Classes and their relationships 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.

**Note:**
Communication diagrams were known as Collaboration diagrams in UML 1.4.

**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**

Select Object diagram elements and connectors from the Object [pages] of the Enterprise Architect UML Toolbox.

Enterprise Architect also supports a number of stereotyped Object [elements] to represent various entities in business modeling.

**Tip:**

Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Object Diagram Elements</th>
<th>Object Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Information Flow</td>
</tr>
<tr>
<td>Object</td>
<td>Associate</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Dependency</td>
</tr>
<tr>
<td>Information Item</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td></td>
</tr>
</tbody>
</table>
15.1.2.4 Composite Structure Diagram

A Composite Structure diagram reflects the internal collaboration of Classes, Interfaces, or Components (and their Properties) 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 of Composite Structure diagrams, see the Toolbox elements listed below.

Toolbox Elements and Connectors

Select Composite Structure diagram elements and connectors from the Composite pages of the Enterprise Architect UML Toolbox.

Enterprise Architect also supports a stereotyped Collaboration to represent a Business Use Case Realization in business modeling.
Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Composite Structure Diagram Elements</th>
<th>Composite Structure Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Class icon]</td>
<td>![Connector icon]</td>
</tr>
<tr>
<td>![Interface icon]</td>
<td>![Assembly icon]</td>
</tr>
<tr>
<td>![Part icon]</td>
<td>![Role Binding icon]</td>
</tr>
<tr>
<td>![Port icon]</td>
<td>![Represents icon]</td>
</tr>
<tr>
<td>![Collaboration icon]</td>
<td>![Occurrence icon]</td>
</tr>
<tr>
<td>![Expose Interface icon]</td>
<td>![Delegate icon]</td>
</tr>
</tbody>
</table>

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 193) states:

A composite structure diagram depicts the internal structure of a classifier, as well as the use of a collaboration in a collaboration use.

15.1.2.4.1 Properties

A property is a nested structure within a classifier, which is usually a Class or an Interface on a Composite Structure diagram. 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 of 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 Enterprise Architect UML Toolbox out to the workspace, right-click on a Part and select the **Advanced** | **Set Property Type** menu option to connect to a classifier.
Note:
If Parts disappear when dragged onto the Class, adjust the Z-order of the Class (right-click on it and select the Z-Order menu option).

The relationship between the two Parts is indicated by the connector, reflecting that communication between the Parts is 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 on the Part and select the Advanced | Custom Properties menu option. 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 alternative representation is shown in the following diagram. However, this depiction fails to express the ownership immediately reflected by containing properties within a classifier.

15.1.2.5 Component Diagram
A Component diagram illustrates the pieces of software, embedded controllers and such that make up a system, and their organization and dependencies. 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, built up so that eventually a component can encompass a large portion of a system.

Example Diagram
The following diagram demonstrates some components and their inter-relationships. Assembly connectors connect 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**

Select Component diagram elements and connectors from the Component pages of the Enterprise Architect UML Toolbox.

**Tip:**

Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Component Diagram Elements</th>
<th>Component Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Assembly</td>
</tr>
<tr>
<td>Component</td>
<td>Delegate</td>
</tr>
<tr>
<td>Class</td>
<td>Associate</td>
</tr>
<tr>
<td>Interface</td>
<td>Realize</td>
</tr>
<tr>
<td>Object</td>
<td>Generalize</td>
</tr>
<tr>
<td>Port</td>
<td></td>
</tr>
<tr>
<td>Expose Interface</td>
<td></td>
</tr>
<tr>
<td>Artifact</td>
<td></td>
</tr>
</tbody>
</table>
15.1.2.6 Deployment Diagram

A Deployment diagram shows how and where the system is to be deployed; that is, its execution architecture. Hardware devices, processors and software execution environments (system Artifacts) are reflected as Nodes, and the internal construction can be depicted by embedding or nesting Nodes. As Artifacts are allocated to Nodes to model the system’s deployment, the allocation is guided by the use of deployment specifications.

Example Diagram

An example Deployment diagram is shown below. 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 Manifest relationships reveal the physical implementation of components ReposCustomer and ReposInternalRecords.

Toolbox Elements and Connectors

Select Deployment diagram elements and connectors from the Deployment pages of the Enterprise Architect UML Toolbox.

Tip:

Click on the following elements and connectors for more information.
<table>
<thead>
<tr>
<th>Deployment Diagram Elements</th>
<th>Deployment Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Communication Path</td>
</tr>
<tr>
<td>Execution Environment</td>
<td>Association Class</td>
</tr>
<tr>
<td>Component</td>
<td>Generalize</td>
</tr>
<tr>
<td>Interface</td>
<td>Realize</td>
</tr>
<tr>
<td>Artifact</td>
<td>Deployment</td>
</tr>
<tr>
<td>Document Artifact</td>
<td>Manifest</td>
</tr>
<tr>
<td>Deployment Specification</td>
<td>Object Flow</td>
</tr>
<tr>
<td>Package</td>
<td>Nesting</td>
</tr>
</tbody>
</table>
15.1.3 Extended Diagrams

In addition to diagrams defined by the UML, Enterprise Architect provides some extended diagram platforms to model business processes or develop custom diagrams.

- Analysis Diagram
- Custom Diagram
- Requirements Diagram
- Maintenance Diagram
- User Interface Diagram
- Database Schema
- Documentation
- Business Modeling and Business Interaction

15.1.3.1 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 requirements.

Enterprise Architect supports some of the Eriksson-Penker Business Extensions that facilitate business process modeling. The complete Eriksson-Penker Business Extensions UML Profile can also be loaded into Enterprise Architect and used to create detailed process models.

Robustness diagrams, used extensively in the ICONIX Process can be created as Analysis diagrams.

Toolbox Elements and Connectors

Select Analysis diagram elements and connectors from the Analysis pages of the Enterprise Architect UML Toolbox.
Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Analysis Diagram Elements</th>
<th>Analysis Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Information Flow</td>
</tr>
<tr>
<td>Object</td>
<td>Object Flow</td>
</tr>
<tr>
<td>Process</td>
<td>Associate</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Realize</td>
</tr>
<tr>
<td>Send</td>
<td>Representation</td>
</tr>
<tr>
<td>Receive</td>
<td></td>
</tr>
<tr>
<td>Information Item</td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td></td>
</tr>
</tbody>
</table>

### 15.1.3.2 Custom Diagram

A Custom diagram is an extended Class diagram that is used to capture requirements, user interfaces or custom-design models.

The below example reflects a Requirements diagram, Requirement elements can be linked back to Use Cases and Components in the system to illustrate how a particular system requirement is met. Change and Defect (Issue) elements look the same as Requirement elements and can be coded and managed in the same way.

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 enable some exploratory and non-rigorous experimentation with model elements and diagrams.
Toolbox Elements and Connectors
Select Custom diagram elements and connectors from the Custom pages of the Enterprise Architect UML Toolbox.

Tip:
Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Custom Diagram Elements</th>
<th>Custom Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Associate</td>
</tr>
<tr>
<td>Requirement</td>
<td>Aggregate</td>
</tr>
<tr>
<td>Issue</td>
<td>Generalize</td>
</tr>
<tr>
<td>Change</td>
<td>Realize</td>
</tr>
<tr>
<td>Screen</td>
<td>Nesting</td>
</tr>
<tr>
<td>UI Control</td>
<td></td>
</tr>
<tr>
<td>Test Case</td>
<td></td>
</tr>
</tbody>
</table>
15.1.3.3 Requirements Diagram

A Requirements diagram is a custom diagram used to describe a system’s requirements or features as a visual model.

Requirements are defined using Requirement elements (Custom elements of type Requirement). To view the detailed description of a Requirement, double-click on the element to display its properties. Requirement elements can 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 enable traceability between specifications and design requirements, and the model elements that realize them.

Requirements can have relationships with other elements such as other Requirements and Use Cases. To view the traceability of a requirement, use the Hierarchy window, which you access using the View | Hierarchy menu option (or press [Ctrl]+[Shift]+[4]).

Toolbox Elements and Connectors

Select Requirements diagram elements and connectors from the Requirements pages of the Enterprise Architect UML Toolbox.

Tip:

Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Requirements Diagram Elements</th>
<th>Requirements Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Aggregate</td>
</tr>
<tr>
<td>Requirement</td>
<td>Inheritance</td>
</tr>
<tr>
<td>Feature</td>
<td>Associate</td>
</tr>
<tr>
<td>Object</td>
<td>Implements</td>
</tr>
</tbody>
</table>
**15.1.3.4 Maintenance Diagram**

A Maintenance diagram is a custom diagram used to describe change requests and issue items within a system model.

An example Maintenance diagram is shown below. Change, Task and Issue elements can be linked back to other model elements in the system to illustrate how they must be modified, fixed or updated.

Maintenance models provide extensions to the UML model and enable change management of change items, and of the model elements that require the changes to be made to 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.

**Toolbox Elements and Connectors**

Select Maintenance diagram elements and connectors from the Maintenance pages of the Enterprise Architect UML Toolbox.

**Tip:**

Click on the following elements and connectors for more information.

<table>
<thead>
<tr>
<th>Maintenance Diagram Elements</th>
<th>Maintenance Diagram Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Package" /></td>
<td><img src="image" alt="Aggregate" /></td>
</tr>
<tr>
<td><img src="image" alt="Issue" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Change" /></td>
<td></td>
</tr>
</tbody>
</table>
### 15.1.3.5 User Interface Diagram

*User Interface Diagrams* are custom diagrams used to visually mock-up a system's user interface using forms, controls and labels.

In the example *User Interface* diagram below, 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.

**Toolbox Elements and Connectors**

Select User Interface diagram elements and connectors from the *User Interface* pages of the Enterprise Architect UML Toolbox.

**Note:**

Click on the following elements and connectors for more information.
15.1.3.6 Database Schema

Below is an example Database Schema, used in Data Modeling.

Toolbox Elements and Connectors

Select Database Schema diagram elements from the Data Modeling pages of the Enterprise Architect UML Toolbox.
15.1.3.7 Business Modeling/Interaction

Business Modeling diagrams and Business Interaction diagrams enable you to model both the structure and behavior of a business system.

Business Modeling diagrams are based on a Class (UML Structural) diagram, whilst Business Interaction diagrams are based on a Sequence (UML Behavioral) diagram. Both diagram types have the same default Enterprise Architect UML Toolbox, which consists of a Business Modeling element page. The available elements include stereotyped Objects and a stereotyped Actor (Business Actor), Use Case (Business Use Case) and Collaboration (Business Use Case Realization).

The following diagram shows the appearance of the elements when dragged and dropped onto a Business Modeling diagram:

![Business Modeling Diagram]

The following diagram shows the appearance of the elements when dragged and dropped onto a Business Interaction diagram:

![Business Interaction Diagram]
15.2 UML Elements

Models in UML are constructed from elements such as Classes, Objects, Interfaces, Use Cases, Components and Nodes, each of which has a different purpose, different rules and different notation. Model elements are used at different stages of the design process for different purposes.

This topic provides an introduction to elements defined by UML, which together compose the backbone of modeling. Most conceivable modeling elements are stereotypes or extensions of the elements introduced in this topic.

- 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 are 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.
15.2.1 Behavioral Diagram Elements

The following figure illustrates the main UML elements that are used in Behavioral Diagrams. For more information on using each element, click on the element name in this list:

- Action
- Activity
- Actor
- Central Buffer Node
- Choice
- Collaboration
- Combined Fragment
- Datastore
- Decision
- Diagram Frame
- Diagram Gate
- Endpoint
- Entry Point
- Exception
- Expansion Region
- Exit Point
- Final
- Flow Final
- Fork
- History
- Initial
- Interaction Occurrence
- Interruptible Activity Region
- Join
- Junction
- Lifeline
- Note
- Object
- Package
- Partition
- Receive
- Region
- Send
- State
- State Lifeline
- State/Continuation
- Structured Activity
- State Machine
- Synch
- System Boundary
- Terminate
- Trigger
- Use Case
- Value Lifeline

**Note:**

Actor, Collaboration, Note, Object and Package elements are used in both Behavioral diagrams and Structural diagrams.
15.2.1.1 Action

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.
An Action can be further defined with **pre-condition and post-condition** notes, and certain properties can be **graphically depicted** on the Action. The data values passed out of and into an Action can be represented by **Action Pins**.

An Action can be depicted as an **Expansion Node** to indicate that the Action comprises an **Expansion Region**.

---

**Toolbox Icon**

- Action

---

**OMG UML Specification**

The OMG UML specification *(UML Superstructure Specification, v2.1.1, p. 241)* states:

*An action is a named element that is the fundamental unit of executable functionality. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise.*

The OMG UML specification *(UML Superstructure Specification, v2.1.1, p. 313)* also states:

*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.*

---

**15.2.1.1 Action Notation**

Some properties can be graphically depicted on an **Action** element, as shown below.

- **Action Notation Kind:** SendSignal
- **Action Notation Kind:** AcceptEvent
- **Action Notation Kind:** AcceptEventTimer
- **Action Notation Kind:** CallOperation
- **Action Notation Kind:** CallBehavior

These properties can be defined by right-clicking on the Activity and selecting the **Advanced | Custom Properties** menu option, which displays the **Custom Properties** dialog. Set the Action type by selecting a value from the **Kind** drop-down list.
Class Operations in Activity Diagrams

Operations from Classes can be displayed on Activity diagrams as Actions. When an operation is shown as an Action, the notation of the Action displays the name of the Class that features the operation. To add an operation to an Activity diagram follow the steps below:

1. Open an Activity diagram.
2. From the Project Browser open a Class and locate the operation 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 displays the name of the Class that features the operation.

If it becomes necessary to change the operation that this Action refers to, follow the steps below:

1. Right-click on the Action. The context menu displays.
2. Select the Advanced | Set Operation menu option. The Set Operation dialog displays.
3. If necessary, in the In Namespace field, select the model that contains the required operation.
4. Double-click on the required operation. The Action updates to show the new classifier and operation.
15.2.1.1.1 Set Feature Dialog

The Set Feature dialog is the Set Operation dialog used to change the operation represented by an Action on an Activity diagram.

As the Set Operation or Set Attribute dialog, it is also used to set the Value operation or attribute for Tagged Values of type RefGUID.

To use this dialog, follow the steps below:

1. The Set Operation (or Set Attribute) dialog displays.

2. In the In Namespace field, click on the drop-down arrow and select the model that contains the required operation or attribute. All operations or attributes in that model are listed on the dialog.

3. Double-click on the required item to set it.

15.2.1.1.2 Action Expansion Node

Representing an Action as an Expansion Node is a shorthand notation to indicate that the Action comprises an Expansion Region.

To specify an Action as an Expansion Node, right-click on the Action to display the context menu and select the Embedded Elements | Add Expansion Node menu option. After designating an Action as an Expansion Node, you can modify or delete it using the Embedded Elements | Embedded Elements menu option.
15.2.1.1.3 Action Pin

An Action Pin is used to define the data values passed 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:

See UML Superstructure Specification, v2.1.1, Figure 12.110, p. 391.

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 to display the context menu and select the Embedded Elements | Add Action Pin menu option. To change the type of an Action Pin, right-click on the pin and select the Advanced | Custom Properties menu option. The following properties can be set:

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>isReadonly</td>
<td>False</td>
</tr>
<tr>
<td>precondition</td>
<td></td>
</tr>
<tr>
<td>isSingleExecution</td>
<td>False</td>
</tr>
<tr>
<td>postcondition</td>
<td></td>
</tr>
<tr>
<td>parameterName</td>
<td></td>
</tr>
</tbody>
</table>

© 1998-2008 Sparx Systems Pty Ltd
15.2.1.4 Local Pre/Post Conditions

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.

See UML Superstructure Specification, v2.1.1, Figure 12.32, p. 316.

Create a Constraint

To attach a constraint to an Action follow the steps below:

1. Right-click on the Action. The context menu displays:
2. Select the Add | Constraint menu option. A Note is created on the diagram, connected to the Action.
3. Right-click on the Note. The context menu displays.
4. Select the Properties menu option. The Constraint dialog displays.
5. In the Constraint Type field, click on the drop-down arrow and select the required constraint type.
6. In the Constraint field, type the text for the constraint.
7. Click on the OK button to save the constraint.
15.2.1.2 Activity

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 in Activity diagrams for various modeling purposes, from procedural-type application development for system design, to business process modeling of organizational structures or work flow.

The following simple diagram of an Activity contains Action elements and includes input parameters and output parameters.

You can define an Activity as a composite element, either during creation or during later edits. However, when creating a composite Activity element you can also use the Structured Activity element, which is tailored to this purpose. If converting an Activity element, right-click on the element and select the Advanced | Make Composite context menu option. The New Structured Activity dialog displays; for information on this dialog, see the Structured Activity topic.

Certain properties can be graphically depicted on an Activity. The Actions in an Activity can be further organized by Activity Partitions.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 318) 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 that is implemented by an activity containing actions that execute before the call action completes.

### 15.2.1.2.1 Activity Notation

Certain properties can be graphically depicted on an Activity element, as shown below.

To define these properties, right-click on the Activity and select the **Advanced | Custom Properties** menu option. The following dialog displays:
15.2.1.2.2 Activity Parameter Nodes

An Activity Parameter Node accepts input to an Activity and provides output from an Activity. The following example depicts two entry parameters and one output parameter defined for the Activity.

To define an Activity Parameter Node for an Activity, follow the steps below:

1. Right-click on the element and select the **Embedded Elements | Add Activity Parameter** menu option.
2. The Properties dialog displays, which prompts for the Name and other properties of the embedded element.
3. After closing this dialog, you can further define the new activity parameter. Right-click on the Activity Parameter and select the **Advanced | Custom Properties** menu option. The following dialog displays:

   ![Properties Dialog](image)

   Similar to characterizing Action Pins, Activity Parameter Nodes 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 or not a parameter can accept or post values during the execution of the Activity.

The following example uses the above settings:
OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 338) states:

An activity parameter node is an object node for inputs and outputs to activities.

... Activity parameter nodes are object nodes at the beginning and end of flows that provide a means to accept inputs to an activity and provide outputs from the activity, through the activity parameters.

Activity parameters inherit support for streaming and exceptions from Parameter.

15.2.1.2.3 Activity Partition

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 is shown below:

To define Partitions:

1. Right-click on the Activity element. The context menu displays.
2. Select the Advanced | Partition Activity menu option. The Activity Partitions dialog displays.
3. In the **Name** field, type the name of a partition. Click on the **Save** button.
4. Repeat step 3 for each partition to be created.

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 341*) 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.

### 15.2.1.3 Actor

An **Actor** is a user of the system; *user* can mean a human user, a machine, or even another system or subsystem in the model. Anything that interacts with the system from the outside or system boundary is termed an Actor. Actors are typically associated with **Use Cases**.

Actors can 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, which details the functions a system must provide to satisfy the user requirements.

Actors also represent the role of a user in **Sequence Diagrams**. Enterprise Architect supports a stereotyped Actor element for **business modeling**. The business modeling elements also represent Actors as stereotyped Objects.

**Toolbox Icon**

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 584*) states:

An actor models a type of role played by an entity that interacts with the subject (e.g. by exchanging signals...
and data), but which is external to the subject. ... Actors may represent roles played by human users, external hardware, or other subjects. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity that is relevant to the specification of its associated Use Cases. Thus, a single physical instance may play the role of several different actors and, conversely, a given actor may be played by multiple different instances.

### 15.2.1.4 Central Buffer Node

A **Central Buffer Node** is an object node for managing flows from multiple sources and destinations, represented in an Activity diagram. It acts as a buffer for multiple in-flows and out-flows from other object nodes, but does not connect directly to Actions.

**Toolbox Icon**

![Central Buffer Node Icon](image)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 352) states:

A central buffer node is an object node for managing flows from multiple sources and destinations. ... A central buffer node accepts tokens from upstream object nodes and passes them along to downstream object nodes.

#### 15.2.1.5 Choice

The **Choice** pseudo-state is used to compose complex transitional paths in, for example, a State Machine diagram 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 fires 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 in deciding transition paths at run-time.
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 538) 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 enables 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.

15.2.1.6 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 displays as a transparent window, divided by horizontal dashed lines for each operand.

The following diagram illustrates the use of Combined Fragments, with a Sequence diagram modeling a simplified purchasing process. A loop fragment is created to iterate through an unknown number 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 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 must click near the inside edge or drag a selection rectangle around the fragment. This prevents accidental selection when moving connectors inside the interaction fragment.

**Tip:**
Press and hold [Alt] to move a combined fragment independently of its contents.
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 467) 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.

15.2.1.6.1 Create a Combined Fragment

Use the following guidelines to create a Combined Fragment in Enterprise Architect.

1. Drag the Fragment element from the Interaction Elements page of the Enterprise Architect UML Toolbox. The following dialog displays:
2. In the **Type** field, click on the drop-down arrow and select the interaction operator. See the [Interaction Operators](#) topic for an explanation of the various types of Combined Fragments.

3. In the **Condition** field, specify a condition or interaction constraint for each operand.

4. A rectangular frame displays, partitioned by dashed lines into segments for each operand.

5. Adjust the frame to encompass the required event occurrences for each operand.

### 15.2.1.6.2 Interaction Operators

When creating Combined Fragments, you must apply an appropriate interaction operator to characterize the fragment. The following table provides guidance on the various operators, and their corresponding descriptions.

<table>
<thead>
<tr>
<th>Interaction Operator</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>alt</strong></td>
<td>Divide up interaction fragments based on Boolean conditions.</td>
</tr>
<tr>
<td><strong>opt</strong></td>
<td>Enclose an optional fragment of interaction.</td>
</tr>
<tr>
<td><strong>par</strong></td>
<td>Indicate that operands operate in parallel.</td>
</tr>
<tr>
<td><strong>loop</strong></td>
<td>Indicate that the operand repeats a number of times, as specified by interaction constraints.</td>
</tr>
<tr>
<td><strong>critical</strong></td>
<td>Indicate a sequence that cannot be interrupted by other processing.</td>
</tr>
<tr>
<td><strong>neg</strong></td>
<td>Assert that a fragment is invalid, and implies that all other interaction is valid.</td>
</tr>
<tr>
<td><strong>assert</strong></td>
<td>Specify the only valid fragment to occur. Often enclosed within a consider or ignore operand.</td>
</tr>
<tr>
<td><strong>strict</strong></td>
<td>Indicate that the behaviors of the operands must be processed in strict sequence.</td>
</tr>
<tr>
<td><strong>seq</strong></td>
<td>Indicate that 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.</td>
</tr>
<tr>
<td><strong>ignore</strong></td>
<td>Indicate which messages should be ignored during execution, or can appear anywhere in the execution trace.</td>
</tr>
</tbody>
</table>
Interaction Operator | Use to
--- | ---
consider | Specify 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 | Provide a reference to another diagram.

**Note:**
The ref fragment is not created using the method described in the **Create a Combined Fragment** topic. To create a ref fragment, simply drag an existing diagram from the Project Browser onto the current diagram.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 468-471) 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 be chosen. The chosen operand 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.

If none of the operands has a guard that evaluates to true, none of the operands are executed and the remainder of the enclosing InteractionFragment is executed.

**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. A **break** operator with a guard is chosen when the guard is true and the rest of the enclosing Interaction Fragment is ignored. When the guard of the **break** operand is false, the **break** operand is ignored and the rest of the enclosing InteractionFragment is chosen. The choice between a **break** operand without a guard and the rest of the enclosing InteractionFragment is done non-deterministically.

A CombinedFragment with interactionOperator **break** should cover all Lifelines of the enclosing InteractionFragment.

**Parallel**

The interactionOperator **par** designates that the CombinedFragment represents a parallel merge between the behaviors of the operands. The OccurrenceSpecifications 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 OccurrenceSpecifications of the operands may be interleaved without obstructing the order of the OccurrenceSpecifications 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 OccurrenceSpecifications within each of the operands is maintained in the result.
2. OccurrenceSpecifications on different lifelines from different operands may come in any order.

3. OccurrenceSpecifications on the same lifeline from different operands are ordered such that an OccurrenceSpecification 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 strict sequencing defines a strict ordering of the operands on the first level within the CombinedFragment with interactionOperator `strict`. Therefore OccurrenceSpecifications within contained CombinedFragment will not directly be compared with other OccurrenceSpecifications of the enclosing CombinedFragment.

**Negative**

The interactionOperator `neg` designates that the CombinedFragment represents traces that are defined to be invalid.

The set of traces that defined a CombinedFragment with interactionOperator negative 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 OccurrenceSpecifications (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 OccurrenceSpecifications may interleave into the region, such as 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**

(p. 473) 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 implicitly 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.

**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**

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.
15.2.1.7 Datastore

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 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 connect elements (such as Activities) 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 indicates other flows in the Activity diagram to which you can attach the note (if the behavior applies to multiple flows). To comply with UML 2, preface behavior with the notation «selection» or «transformation».

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 360) 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.

15.2.1.8 Decision

A Decision is an element of an Activity diagram or Interaction Overview 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 alternative flows can be merged (but not synchronized) to form one flow. The following examples show both of these modes of using the decision element.

Used as a decision:

**Used as a merge:**

![Diagram of a merge node](image)


**Note:**

Moving a diagram generally does not affect the location of elements in packages. If you move a diagram out of one package into another, all the elements in the diagram remain in the original package. However, Decision elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram. Therefore, if you move a diagram containing these elements, they are moved to the new parent package with the diagram.

**Toolbox Icon**

- ![Decision Icon](image)

**OMG UML Specification**

The OMG UML specification *(UML Superstructure Specification, v2.1.1, p. 361 (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 Superstructure Specification, v2.1.1, p. 387 (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.
15.2.1.9 Diagram Frame

A Diagram Frame element is a rendition of a diagram dropped from the Project Browser into another diagram. It is a type of Combined Fragment with the Interaction Operator `ref`. However, it can be created on any type of diagram, and is not created in the same way as other Combined Fragments.

When you drop the diagram from the Project Browser onto the open diagram, the following prompt displays:

If you click on the Diagram Frame radio button, a Diagram Frame is inserted into the diagram, containing an image of the dropped diagram.

If you select the Diagram Reference option, an empty frame is inserted with the name of the dropped diagram in the frame label. If you select the Hyperlink radio button, a diagram icon is inserted with no frame, and with the parent package and diagram name next to it.

In all three cases, the object acts as a hyperlink to the real referenced diagram. You can also define properties for the objects, as for other elements, by right-clicking on the object and selecting the element Properties context menu option.
15.2.1.10 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. A fragment might be required to receive or deliver a message; internally, an ordered message reflects this requirement, 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 Interaction diagrams (Sequence, Timing, Communication or Interaction Overview) and interaction occurrences and combined fragments (to specify the expression).

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 480) 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 OccurrenceSpecification that is not in the same scope as the Gate. Gates play different roles: we have formal gates on Interactions, actual gates on InteractionUses, expression gates on CombinedFragments.
### 15.2.1.11 Endpoint

An *Endpoint* is used in Interaction diagrams (Sequence, Timing, Communication, or Interaction Overview) 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 requires some timing specifications to draw the connection.

The following example depicts a lost message in a Sequence diagram.

![Sequence Diagram Example](image)

**Toolbox Icon**

- **Endpoint**

**OMG UML Specification**

The OMG UML specification *(UML Superstructure Specification, v2.1.1, p. 492)* 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.
15.2.1.12 Entry Point

Entry Point pseudo-states are used to define the beginning of a State Machine. An Entry Point exists for each region, directing the initial concurrent state configuration.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 471) states:

An entry point pseudostate is an entry point of a state machine or composite state. In each region of the state machine or composite state it has a single transition to a vertex within the same region.

15.2.1.13 Exception

The Exception Handler element defines the group of operations to carry out when an exception occurs. In an Activity diagram the protected element can contain a set of operations and is connected 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.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 364) states:

An exception handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node.

15.2.1.14 Expansion Region

You create an Expansion Region as one variant of a Region (the other is an Interruptible Activity Region).

On an Activity diagram, 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.

To modify the mode of an Expansion Region, right-click on it and select the Advanced | Custom Properties menu option.

See UML Superstructure Specification, v2.1.1, figure 12.87, p. 372.
Toolbox Icon

OMG UML Specification
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 367) states:

An expansion region is a structured activity region that executes multiple times corresponding to elements of an input collection.

15.2.1.14.1 Add Expansion Region
When you add a Region element to a diagram, the following prompt displays:

The Select type defaults to InterruptibleActivityRegion.

1. Select the type ExpansionRegion.
2. In the Kind field, click on the drop-down arrow and select the concurrency attribute.

15.2.1.15 Exit Point

Exit Points are used in Submachine states and State Machines to denote the point where the machine is exited and the transition sourcing this exit point, for Submachines, is triggered. Exit points are a type of pseudo-state used in the State Machine diagram.

Toolbox Icon

OMG UML Specification
The OMG UML specification (UML Superstructure Specification, v2.1.1 p. 538) states:

An exit point pseudostate is an exit point of a state machine or composite state. Entering an exit point within any region of the composite state or state machine referenced by a submachine state implies the exit of this composite state or submachine state and the triggering of the transition that has this exit point as source in the state machine enclosing the submachine or composite state.

15.2.1.16 Final
There are two nodes used to define a Final state in an Activity, both defined in UML 2.1 as of type Final Node. The Activity 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 that 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 (that is, stop installing this component, but keep on installing other components). It is only after the Deliver Application activity is completed, after the control flow reaches the Final node, that all flows stop.

The node that initiates a flow is the Initial node.


Note:
Moving a diagram generally does not affect the location of elements in packages. If you move a diagram out of one package into another, all the elements in the diagram remain in the original package. However, Final elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram. Therefore, if you move a diagram containing these elements, they are moved to the new parent package with the diagram.

Toolbox Icon

OMG UML Specification
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 332) states:

An activity may have more than one activity final node. The first one reached stops all flows in the activity.

15.2.1.17 Flow Final

There are two nodes used to define a final state in an Activity, both defined in UML 2.1 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 Activity Diagram 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 (that is,
stop installing this component, but keep on installing other components). It is only after the Deliver Application activity is completed, after the control flow reaches the Final node, that all flows stop.


Note:
Moving a diagram generally does not affect the location of elements in packages. If you move a diagram out of one package into another, all the elements in the diagram remain in the original package. However, Flow Final elements are used only within one diagram, have no meaning outside that diagram, and are never reused in any other diagram. Therefore, if you move a diagram containing these elements, they are moved to the new parent package with the diagram.

Toolbox Icon

OMG UML Specification
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 375) states:

A flow final destroys all tokens that arrive at it. It has no effect on other flows in the activity.

15.2.1.18 Fork/Join

The Fork/Join elements have different modes of use, 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.

These elements are used in both Activity and State Machine diagrams. With respect to State Machine diagrams, Forks and Joins are used 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 see their specific topics.

Some examples of Fork/Join nodes include:

Fork or split the flow into a number of concurrent flows:
Join the flow of a number of concurrent flows:

Join and Fork a number of incoming flows to a number of outgoing flows:

**Toolbox Icon**

**OMG UML Specification**

**Fork**
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 376) 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 Superstructure Specification, v2.1.1, p. 381-382) states:

A join node is a control node that synchronizes multiple flows... A join node has multiple incoming edges and one outgoing edge.
15.2.1.18.1 Fork


These elements are used in both Activity and State Machine diagrams. With respect to State Machine diagrams, a Fork pseudo-state signifies that its incoming transition comes from a single state, and it has 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 must not have triggers or guards. The following diagram demonstrates a Fork pseudo-state dividing into two concurrent regions, which then return to the End State via the Join pseudo-state.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 538) 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.
15.2.1.18.2 Join

The Join element is used by 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 must not have triggers or guards. The following diagram demonstrates a Fork pseudo-state dividing into two concurrent Regions, which then return to the End State via the Join.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 538) 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.
15.2.1.19 **History**

There are two types of History pseudo-state 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.

---

**Toolbox Icon**

[Image of History icon]

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 537) states:

---

**Enterprise Architect User Guide**
... 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.

15.2.1.20 Initial

The Initial element is used by 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.

See UML Superstructure Specification, v2.1.1, Figure 12.97, p. 378.

The activity flow is completed by a Final or Flow Final node.

Note:

Moving a diagram generally does not affect the location of elements in packages. If you move a diagram out of one package into another, all the elements in the diagram remain in the original package. However, Initial elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram. Therefore, if you move a diagram containing these elements, they are moved to the new parent package with the diagram.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 537) 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 outgoing transition from the initial vertex may have a behavior, but not a trigger or guard.

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 378) also states:

An initial node is a control node at which flow starts when the activity is invoked.
15.2.1.21 Interaction Occurrence

An Interaction Occurrence is a reference to an existing Interaction diagram (Sequence, Timing, Communication or Interaction Overview). 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 open an Interaction diagram and drag another Interaction diagram into its workspace. A dialog displays, providing configuration options.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 423) refers to an Interaction Occurrence as an InteractionUse, and states:

An InteractionUse refers to an Interaction. The InteractionUse is a shorthand for copying the contents of the referred Interaction where the InteractionUse 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 InteractionUse allows multiple interactions to reference an interaction that represents a common portion of their specification.

15.2.1.22 Interruptible Activity Region

You create an Interruptible Activity Region as one variant of a Region (the other is an Expansion Region).

In an Activity diagram, 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 kills any processing of the order at the receipt, filling or shipping stage.
See UML Superstructure Specification, v2.1.1, figure 12.100, p. 381.

To create an Interruptible Activity Region, click on the Add an Interruptible Activity Region link.

**Toolbox Icon**

![Region Icon]

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 380) 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.

### 15.2.1.22.1 Add Interruptible Activity Region

When you add a **Region** element to an Activity diagram, the following prompt displays:

- Select type:
  - InterruptibleActivityRegion
  - ExpansionRegion

- **Kind**: iterative

The **Select type** panel defaults to **InterruptibleActivityRegion** and the **Kind** field is disabled. Click on the **OK** button.

### 15.2.1.23 Junction

**Junction** pseudo-states are used to design complex transitional paths in State Machine diagrams. 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.
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 538) 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 a 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.

15.2.1.24 Lifeline

A Lifeline is an individual participant in an interaction (i.e. Lifelines cannot have multiplicity). A Lifeline
represents a distinct connectable element. To specify that representation within Enterprise Architect, right-click on the Lifeline and select the Advanced | Instance Classifier menu option. A dialog displays containing a selectable list of all project classifiers.

Lifelines are available in Sequence diagrams. There are different Lifeline elements for Timing diagrams (State Lifeline and Value Lifeline); however, although the representation differs between the two diagram types, the meaning of the Lifeline is the same.

Toolbox Icon

OMG UML Specification
The OMG UML specification (UML Superstructure Specification, v2.1.1, p.489) 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.

15.2.1.25 Merge

A Merge Node brings together a number of alternative flow paths in Activity, Analysis and Interaction Overview diagrams. For example, if a Decision is used after a Fork, the two flows coming out of the Decision must be merged into one before going to a Join; otherwise, the Join waits for both flows, only one of which arrives.

A Merge Node has multiple incoming edges and a single outgoing edge. The edges coming into and out of a Merge Node must be either all object flows or all control flows.

Toolbox Icon

OMG UML Specification
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 387) 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.

15.2.1.26 Message Endpoint

A Message Endpoint element defines the termination of a State or Value Lifeline in a Timing diagram.
15.2.1.27 Message Label

A Message Label is an alternative way of denoting 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 following diagram illustrates how Message Labels are used to construct a message between Lifelines.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 518) 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.

15.2.1.28 Note

A Note element is a textual annotation that can be attached to a set of elements of any other type. The attachment is created separately, using a Notelink connector. Both Note and Notelink are available in any
Enterprise Architect diagram, through the Common pages of the Enterprise Architect UML Toolbox.

A Note is also called a Comment.

A Constraint is a form of Note, identifying a constraint on other elements. As for a Note, you can connect the Constraint element to other elements using a Notelink connector. This element is just a means of documenting the fact that there are constraints; it has no impact on the other elements. You define the types of constraint in the project reference data, apply them to the element in the element Properties dialog, and manage them through the Rules & Scenarios window.

**Toolbox Icon**

- **Note**
- **Constraint**

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 59) states:

A comment gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.

A comment can be owned by any element.

15.2.1.29 Partition

Activity Partitions are used to logically organize an Activity. They do not affect the token flow of an Activity diagram, but help structure all or parts of the view.

The following example depicts the partitioning between the Classes Process Payment and Order Processor.
The partition orientation defaults to horizontal. To turn it into a vertical partition, right-click on it and select the **Advanced | Vertical Partition** menu option.

**Toolbox Icon**

![Partition Icon](image)

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 341*) 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.*

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 341*) also states:

*An activity partition is a kind of activity group for identifying actions that have some characteristic in common.*

**15.2.1.30 Receive**

A **Receive** element is used to define the acceptance or receipt of a request, in an **Activity diagram**. 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 (pennant shape)
- **Accept Time Event Action** element (hourglass shape)

The following example reflects a payment process on an order. Upon receiving the payment (from **Request Payments**, a **Send** element), the payment is confirmed and the flow continues to ship the order.

[Activity Diagram]

See *UML Superstructure Specification, v2.1.1, figure 12.26, p. 312*.

To depict an Accept Time Event, use the standard Receive element from the Enterprise Architect UML Toolbox. Right-click on this element, and select the **Advanced | Accept Time Event** menu option. The following example shows the hourglass-shaped Accept Time Event Action:

[Activity Diagram]

See *UML Superstructure Specification, v2.1.1, figure 12.27, p. 312*. 
**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 239) states:

*AcceptEventAction is an action that waits for the occurrence of an event meeting specified conditions.*

15.2.1.31 **Region**

Enterprise Architect supports two types of Region element:

- **Expansion Region**
- **Interruptible Activity Region**

When you add a Region element to an Activity diagram, the following prompt appears. You use this to select the Region type.

15.2.1.32 **Send**

The Send element is used to depict the action of sending a signal, in an Activity diagram. It is the opposite of a Receive element.

The following example shows an order being processed, where a signal is sent to fill the processed order and, upon creation of the resulting invoice, a notification is sent to the customer.
See UML Superstructure Specification, v2.1.1, figure 12.132, p. 408.

**Toolbox Icon**

![Send](image1)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 285) 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.

### 15.2.1.33 State

A State represents a situation where some invariant condition holds; this condition can be static (waiting for an event) or dynamic (performing a set of activities). State modeling is usually related to Classes and describes the enableable states a Class or element can be in and the transitions that enable the element to move there. There are two types of State: Simple States and Composite States, both created from the State element from the Enterprise Architect UML Toolbox.

Furthermore, there are pseudo-states resembling some aspect of a State but with a pre-defined implication. Pseudo-states model complex transitional paths, and classify common State Machine behavior.

You can define entry, internal and exit actions for a State using operations.

If a State element has features such as attributes or operations, the depiction of the element in a diagram has a line under the element name. This line persists if the features are hidden. The line also displays if the Show State Compartment checkbox is selected on the Objects page of the Options dialog (Tools | Options | Objects).

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 546) 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).
15.2.1.33.1 Composite State

Composite States are composed within the State Machine diagram by expanding a State element, adding Regions if applicable, and dragging further State elements, related elements and connectors within its boundaries. The internal State elements are then referred to as Sub-states.

(You can also define a State element, as with many other types of element, as a composite element; this then has a hyperlink to a child diagram that can be another State Machine diagram or other type of diagram elsewhere in the model.)

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 use, generates a situation of multiple States concurrently active; this situation is referred to as the active State configuration.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, 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.

15.2.1.34 State Lifeline

A Lifeline is the path an object takes across a measure of time, as indicated by the x-axis. There are two sorts: State Lifelines (defined here) and Value Lifelines, both used in Timing diagrams.

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 shown below:

See UML Superstructure Specification, v2.1.1, figure 14.29, p. 519.

A State Lifeline consists of a set of transition points. Each transition point can be defined with the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>At time</td>
<td>Specifies the starting time for a change of state.</td>
</tr>
<tr>
<td>Transition to</td>
<td>Indicates the state to which the lifeline changes.</td>
</tr>
<tr>
<td>Event</td>
<td>Describes the occurring event.</td>
</tr>
<tr>
<td>Timing constraints</td>
<td>Refers to the time taken for a state to change within a lifeline, or the time taken to transmit a message (e.g. t..t+3).</td>
</tr>
<tr>
<td>Timing observations</td>
<td>Provides information on the time of a state change or sent message.</td>
</tr>
<tr>
<td>Duration constraints</td>
<td>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.</td>
</tr>
<tr>
<td>Duration observations</td>
<td>Indicates the interval of a lifeline at a particular state, begun from a change in state or message receipt.</td>
</tr>
</tbody>
</table>

In the example diagram above, the OK transition point has these properties:
15.2.1.35  State/Continuation

The State/Continuation element serves two different purposes for Interaction (Sequence) diagrams, as State Invariants and Continuations. Enterprise Architect prompts you to identify the purpose when you create the element.

Toolbox Icon

15.2.1.35.1  Continuation

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 a matching Continuation (same name) preceding the Interaction Fragment.

You create a Continuation by dragging the State/Continuation element onto the diagram from the Interaction Elements page of the Enterprise Architect UML Toolbox.

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.
The following diagram shows the interaction referenced by the *Interaction Occurrence*. 
OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 474) states:

A Continuation is a syntactic way to define continuations of different branches of an Alternative CombinedFragment. Continuation is intuitively similar to labels representing intermediate points in a flow of control.

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 474) also 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.

15.2.1.35.2 State Invariant

A State Invariant is a condition applied to a Lifeline which must be fulfilled for the Lifeline to exist. You create a State Invariant by dragging the State/Continuation element onto the diagram from the Interaction Elements page of the Enterprise Architect UML Toolbox.

The following diagram illustrates a State Invariant.

![State Invariant Diagram](image)

When a State Invariant is moved near to a Lifeline, it snaps to the center. If the sequence object is dragged left or right, the State Invariant moves with it.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 502) states:

A StateInvariant is a runtime constraint on the participants of the interaction. It may be used to specify a variety of different kinds of constraints, such as values of attributes or variables, internal or external states, and so on.

A StateInvariant is an InteractionFragment and it is placed on a Lifeline.

15.2.1.36 Structured Activity

You can add a Structured Activity element to an Activity diagram. A Structured Activity element is a composite of an element, a connector and a child Activity diagram, which is represented by the small symbol in the bottom right-hand corner of the element.

There are five types of Structured Activity, each representing a particular structure of activity events. The five types are represented below:
Simple Composite - represents an arrangement of activities that take place independent of each other

Structured Activity node - represents an ordered arrangement of activities; that is, activities that have relationships

Conditional node - represents an arrangement of activities where choice determines which activities are performed

Sequential node - represents a sequential arrangement of activities

Loop node - represents a sequence of activities that are - or can be - repeated on the same object.

To create a Structured Activity, drag the Structured Activity element from the Activity Elements page in the Enterprise Architect UML Toolbox onto the diagram. The New Structured Activity dialog displays, on which you specify which type of Structured Activity to create.

To display the Activity diagram represented by the Structured Activity element, double-click on the element.

Nested Structured Activities

Structured Activity elements can point to child diagrams that themselves contain or consist of Structured Activity elements; that is, the Structured Activity elements are nested. When you create nested Structured Activity elements, they are shown as nested in the Project Browser; see the example below.
OMG UML Specification

Structured Activity Node
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 409) states:

A structured activity node is an executable activity node that may have an expansion into subordinate nodes as an ActivityGroup. The subordinate nodes must belong to only one structured activity node, although they may be nested.

A structured activity node represents a structured portion of the activity that is not shared with any other structured node, except for nesting.

Loop Node
The OMG UML specification (UML Superstructure Specification, v2.1.1, pp. 384-385) states:

A loop node is a structured activity node that represents a loop with setup, test, and body sections.

Each section is a well-nested subregion of the activity whose nodes follow any predecessors of the loop and precede any successors of the loop. The test section may precede or follow the body section. The setup section is executed once on entry to the loop, and the test and body sections are executed repeatedly until the test produces a false value. The results of the final execution of the test or body are available after completion of execution of the loop.

Sequential Node
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 408) states:

A sequence node is a structured activity node that executes its actions in order.

Conditional Node
The OMG UML specification (UML Superstructure Specification, v2.1.1, p.355) states:

A conditional node is a structured activity node that represents an exclusive choice among some number of alternatives.

A conditional node consists of one or more clauses. Each clause consists of a test section and a body section. When the conditional node begins execution, the test sections of the clauses are executed. If one or more test sections yield a true value, one of the corresponding body sections will be executed. If more than one test section yields a true value, only one body section will be executed. The choice is nondeterministic unless the test sequence of clauses is specified. If no test section yields a true value, then no body section is executed; this may be a semantic error if output values are expected from the conditional node.

15.2.1.37 State Machine

A State Machine element is a container for groups of related State elements. You can create sections of a State Machine diagram, showing the organization of the inter-related State elements, and enclose each section in a State Machine element. You can also create Regions on a State Machine Element.
15.2.1.38 Synch

A Synch state is useful for indicating that concurrent paths of a State Machine are synchronized. After bringing the paths to a synch state, the emerging transition indicates unison.

15.2.1.39 System Boundary

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 System Boundary can be set: the name, the border style, and the number of horizontal or vertical swim lanes.
A System Boundary element can be marked as Selectable, using the element's context menu. When not selectable, you can click within the System Boundary space without activating or selecting the Boundary itself. This is useful when you have many elements within the Boundary and the Boundary makes their selection difficult.

**Note:**
A System Boundary can have an associated image that it displays instead of its default format. Use the **Appearance | Alternate Image** menu option to select an image.

**OMG UML Specification**
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 594) 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.*

**15.2.1.40 Terminate**

The **Terminate** pseudo-state indicates that upon entry of its pseudo-state, the State Machine's execution ends.

**Toolbox Icon**

[Terminator icon]

**15.2.1.41 Trigger**

A Trigger indicates an event that initiates an action (and might arise from completion of a previous action). You initially define a Trigger in one of two ways:
As a property of a Transition relationship
As an event in a State Machine Table.

When you save the Trigger, it is added to the list of elements for the parent package in the Project Browser. You can then right-click on it and, if required, edit its properties. You can also drag the Trigger element onto another diagram, although there are limited uses for the element in that context.

This element is not the same as a Trigger Operation, which is an operation automatically executed as a result of the modification of data in a database.

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 456) states:

"Events may cause execution of behavior (e.g., the execution of the effect activity of a transition in a state machine). A trigger specifies the event that may trigger a behavior execution as well as any constraints on the event to filter out events not of interest."

### 15.2.1.42 Use Case

A Use Case is a UML modeling element that describes how a user of the proposed system interacts 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: the interaction either completed or 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
- Can have an associated Sequence diagram illustrating behavior over time; who does what to whom, and when
- Typically has scenarios associated with it that describe the work flow over time that produces the end result; alternative work flows (for example, to capture exceptions) are also enabled.

**Note:**

Use a Use Case diagram and model to build up the functional requirements and implementation details of the system.

The following is an example Use Case model:
If extending a Use Case, you can specify the points of extension with Use Case Extension Points. To display the attributes, operations or constraints of a Use Case on a diagram, use Rectangle Notation.

Enterprise Architect also provides two stereotyped Use Cases - the Test Case and the Business Use Case.

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 592) states:

*A UseCase is a kind of behaviored classifier that represents a declaration of an offered behavior. Each Use Case specifies some behavior, possibly including variants, that the subject can perform in collaboration with one or more actors.*

**15.2.1.42.1 Use Case Extension Points**

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:**

Conditions to apply the extending Use Case, and the extension point to use, should be attached as a note to the extend relationship.
To work with extension points, follow the steps below:

1. Right-click on the Use Case element. The context menu displays.
2. Select the **Advanced | Edit Extension Points...** menu option. The **Use Case Extension Points** dialog displays, listing defined points for that Use Case.
3. Select an extension point in the list and click on the appropriate button to add, edit or remove an extension point.

### 15.2.1.42.2 Rectangle Notation

You can display a **Use Case** using rectangle notation. This displays the Use Case in a rectangle, with an oval in the top right-hand corner. Any attributes, operations or constraints 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 the **Advanced | Use Rectangle Notation** context menu option. This setting only applies to the selected Use Case, and can be toggled on and off.

**Note:**

**Actor** elements can also be displayed using rectangle notation.

### 15.2.1.43 Value Lifeline

A **Lifeline** is the path an object takes across a measure of time, indicated by the x-axis. There are two sorts: **Value Lifelines** (defined here) and **State Lifelines**, both used in **Timing diagrams**.
A Value Lifeline shows the Lifeline's state across the diagram, with 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 below:

![Value Lifeline Diagram](image)

See UML Superstructure Specification, v2.1.1, Figure 14.30, p. 520.

A Value Lifeline consists of a set of transition points. Each transition point can be defined with the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>At time</td>
<td>Specifies the starting time for a change of state.</td>
</tr>
<tr>
<td>Transition to</td>
<td>Indicates the state to which the Lifeline will change.</td>
</tr>
<tr>
<td>Event</td>
<td>Describes the occurring event.</td>
</tr>
<tr>
<td>Timing constraints</td>
<td>Refers to the time taken for a state to change within a Lifeline, or the time taken to transmit a message.</td>
</tr>
<tr>
<td>Timing observations</td>
<td>Provides information on the time of a state change or sent message.</td>
</tr>
<tr>
<td>Duration constraints</td>
<td>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.</td>
</tr>
<tr>
<td>Duration observations</td>
<td>Indicates the interval of a Lifeline at a particular state, begun from a change in state or message receipt.</td>
</tr>
</tbody>
</table>

In the example diagram above, the 10ms transition point has these properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>At Time</td>
<td>10ms</td>
</tr>
<tr>
<td>Transition to</td>
<td>Waitcard</td>
</tr>
<tr>
<td>Event</td>
<td>Switch</td>
</tr>
<tr>
<td>Timing constraints</td>
<td></td>
</tr>
<tr>
<td>Timing observations</td>
<td></td>
</tr>
<tr>
<td>Duration constraints</td>
<td>$d..3^*d$</td>
</tr>
<tr>
<td>Duration observations</td>
<td></td>
</tr>
</tbody>
</table>

**Toolbox Icon**

![Value Lifeline Icon](image)

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 518*) 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.*
15.2.2 Structural Diagram Elements

The following figure illustrates the main UML elements that are used in Structural Diagrams. For more information on using each element, click on the element name in this list:

- Actor
- Artifact
- Class
- Collaboration
- Collaboration Occurrence
- Component
- Deployment Specification
- Document Artifact
- Enumeration
- Execution Environment
- Expose Interface
- Information Item
- Interface
- Node
- Note
- Object
- Package
- Part
- Port
- Primitive
- Qualifiers
- Signal

Note:
Actor, Collaboration, Note, Object and Package elements are used in both Behavioral diagrams and Structural diagrams.
15.2.2.1 Artifact

An Artifact is any physical piece of information used or produced by a system, represented in a Deployment Diagram. 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.
You can also create an Artifact element on a diagram for an external file, by clicking on the file in a file list (such as Windows Explorer) or on your Desktop and dragging it onto the diagram. A short context menu displays with two options - Hyperlink and Artifact.

Click on the Artifact option to create the element on the diagram. The Properties dialog displays, and you can define the name or other properties as required. Click on the OK button, and then open the Properties dialog again and click on the Files tab. The file pathname is listed in the Files panel.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 201) 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.

15.2.2.2 Class

A Class is a representation of objects that reflects their structure and behavior within the system. It is a template from which actual running instances are created, although a Class can be defined either to control its own execution or as a template or parameterized Class that specifies parameters that must be defined by any binding Class.

A Class can have attributes (data) and methods (operations or behavior). Classes can 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 a Class, as illustrated below, shows the attributes (or data elements) associated with the 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).

Class elements are generally used in Class diagrams and Composite Structure diagrams.

Enterprise Architect also supports a number of stereotyped Class elements to represent various entities in web-page modeling.
OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, pp. 52-53) 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.

15.2.2.2.1 Active Classes

An Active Class indicates that, when instantiated, the Class controls 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 Enterprise Architect, follow the steps below:

1. Highlight a Class, and display its Properties dialog.
2. Click on the Advanced button.
3. Select the Is Active checkbox.
4. Click on the OK button to save the details.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 438) states:

An active object is an object that, as a direct consequence of its creation, commences to execute its classifier behavior, and does not cease until either the complete behavior is executed or the object is terminated by some external object. (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 is determined solely by the behavior of the active object and not by the invoking object. If the classifier behavior of an active object completes, the object is terminated.

15.2.2.2.2 Parameterized Classes (Templates)

Enterprise Architect supports template or parameterized Classes, which specify parameters that must be defined by any binding Class. A template Class enables 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++.
Enterprise Architect imports and generates 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, follow the steps below:

1. Display the Properties dialog for a Class.
2. Select the Detail tab.
3. In the Type field, click on the drop-down arrow and select **Parameterized**.
4. Click on the Add button and define the required parameters in the **Class Parameter** dialog.
Notation Example

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 622) 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.

15.2.2.3 Collaboration

A Collaboration defines a set of cooperating roles and their connectors. These are used to collectively illustrate a specific functionality, in a Composite Structure diagram. A Collaboration should specify only the roles and attributes required 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 (Objects) connected as shown. The process for this Collaboration can be demonstrated by attaching an Interaction diagram (Sequence, Timing, Communication or Interaction Overview).
To understand referencing a Collaboration in a specific situation, see the Collaboration Occurrence topic. Enterprise Architect supports a stereotyped Collaboration to represent a Business Use Case Realization in business modeling.

**Toolbox Icon**

![Collaboration](image)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 171) states:

*A collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplish some desired functionality. Its primary purpose is to explain how a system works and, therefore, it typically only incorporates those aspects of reality that are deemed relevant to the explanation.*

### 15.2.2.4 Collaboration Occurrence

Use a Collaboration Occurrence to apply a pattern defined by a Collaboration to a specific situation, in a Composite Structure diagram.

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. (See the Collaboration topic for a representation of the Install Collaboration.)
To create a Collaboration Occurrence, drag the required Collaboration from the Project Browser onto the diagram and, on the Paste Element dialog, select the Paste as Link radio button.

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 173) refers to a Collaboration Occurrence as a Collaboration Use, and states:

A collaboration use represents one particular use of a collaboration to explain the relationships between the properties of a classifier. A collaboration use shows how the pattern described by a collaboration is applied in a given context, by binding specific entities from that context to the roles of the collaboration. Depending on the context, these entities could be structural features of a classifier, instance specifications, or even roles in some containing collaboration. There may be multiple occurrences of a given collaboration within a classifier, 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.

15.2.2.5 **Component**

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 can be manifested in Components.

A Component can 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, in Component diagrams. By building the system in discrete Components, localization of data and behavior enables decreased dependency between Classes and Objects providing a more robust and maintainable design.
A component represents 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).

### 15.2.2.6 Deployment Spec

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, as represented in a Deployment diagram. 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, as per the specifications of RepositoryApp, instantiated from the Deployment Specification SystemSpec.
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.

15.2.2.6.1 Device

A Device is a physical electronic resource with processing capability upon which Artifacts can be deployed for execution, as represented in a Deployment diagram. Complex Devices can consist of other devices; that is, a Device can be a nested element, where a physical machine is decomposed into its elements either through namespace ownership or through attributes that are typed by Devices.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, 10.3.7, v2.1.1, p. 207) states:

In the metamodel, a Device is a subclass of Node.

15.2.2.7 Document Artifact

A Document Artifact is an artifact having a stereotype of «document». You create the Document Artifact on a Component, Documentation, or Deployment diagram, and associate it with an RTF document. Double-click on the element to display the Linked Document Editor. See Linked Documents.

15.2.2.8 Enumeration

An Enumeration is a data type, whose instances can be any of a number of user-defined enumeration literals. It is possible to extend the set of applicable enumeration literals in other packages or profiles. You create
Enumerations in Class or Package diagrams and in diagrams developed from the Metamodel pages of the Enterprise Architect UML Toolbox.

**Toolbox Icon**

![Enumeration](image)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 69) states:

An enumeration is a data type whose values are enumerated in the model as enumeration literals.

**15.2.2.9 Execution Environment**

An Execution Environment is a node that offers an execution environment for specific types of components that are deployed on it in the form of executable artifacts. This is depicted in a Deployment diagram.

Execution Environments can be nested; for example, a database Execution Environment can be nested in an operating system Execution Environment. Components of the appropriate type are then deployed to specific Execution Environment nodes.

**Toolbox Icon**

![Execution Environment](image)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 210 states:

... an ExecutionEnvironment is ... usually part of a general Node, representing the physical hardware environment on which the ExecutionEnvironment resides. In that environment, the ExecutionEnvironment implements a standard set of services that Components require at execution time (at the modeling level these services are usually implicit). For each component Deployment, aspects of these services may be determined by properties in a DeploymentSpecification for a particular kind of ExecutionEnvironment.

**15.2.2.10 Expose Interface**

...
The **Expose Interface** element is a graphical method of depicting the required or supplied interfaces of a **Component**, **Class** or **Part** in a **Component** or **Composite Structure** diagram. It just identifies the fact that the element provides or requires an interface; to depict the fact that the provided interface is used, or the required interface provided, by another element use the **Assembly** connector.

The Expose Interface element must be attached to the Class or Component element, and it becomes a child element of that Class or Component; it cannot exist independently. You can attach more than one Expose Element to another element.

When you create the Expose Interface element, a dialog displays in which you enter a name for the element and specify whether it represents a required interface or a provided interface.

**Toolbox Icon**

![Expose Interface Icon](image)

### 15.2.2.11 Information Item

An **Information Item** represents an abstraction of data. It is used in **Activity**, **Analysis** and **Object** diagrams. An Information Item is also represented by an **Information Flow** connector.

**Toolbox Icon**

![Information Item Icon](image)

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 608*) 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.
15.2.2.12 Interface

An Interface is a specification of behavior (or contract) that implementers agree to meet. By implementing an Interface, Classes are guaranteed to support a required behavior, which enables the system to treat non-related elements in the same way; i.e. through the common interface. You also use Interfaces in a Composite Structure diagram.

Interfaces are drawn in a similar way to a Class with operations specified, as shown below. They can 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. Realization connectors to an Interface drawn as a circle are drawn as a solid line without target arrows.

Note:

An Interface cannot be instantiated (i.e. 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.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 88) states:

An interface is a kind of classifier that represents a declaration of a set of coherent public features and obligations. An interface specifies a contract; any instance of a classifier that realizes the interface must fulfill that contract. 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 instantiable. Instead, an interface specification is implemented by an instance of an instantiable classifier, which means that the instantiable classifier presents a public facade that conforms to the interface specification. Note that a given classifier may implement more than one interface and that an interface may be implemented by a number of different classifiers.
15.2.2.13 Node

A Node is a physical piece of equipment on which the system is deployed, such as a workgroup server or workstation. A Node usually hosts components and other executable pieces of code, which again can be connected to particular processes or execution spaces. Typical Nodes are client workstations, application servers, mainframes, routers and terminal servers.

Nodes are used in Deployment diagrams to model the deployment of a system, and to illustrate the physical allocation of implemented artifacts. They are also used in web modeling, from dedicated web modeling pages in the Enterprise Architect UML Toolbox.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 213) 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.

15.2.2.14 Object

An Object is a particular instance of a Class at run time. For example a 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, such as pieces of paper, faxes and information. To model the varying behavior of Objects at run-time, use run-time states.

Early in analysis, Objects can be used to quickly capture all the things that are of relevance within the system domain, in an Object, Composite Structure or Communication diagram. As the model progresses these analysis Objects are refined into generic Classes from which instances can be derived to represent common business items. Once Classes are defined, Objects can be typed; that is they can have a classifier set that indicates their base type. See Object Classifiers.

Enterprise Architect also supports a number of stereotyped Object elements to represent various entities in business modeling.
15.2.2.14.1 Run-time State

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, use instance values and run-time states or run-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 propagate to a list where the run-time state for the Object can be defined. To do this, see the following topics:

- Define a Run-Time Variable
- Remove a Defined Variable
- Define an Object State

The following example defines run-time values for the listed variables, which are attributes of the instances’ classifier AccountItem.

15.2.2.14.1.1 Define a Run-time Variable

To add run-time state instance variables to an Object, follow the steps below:

1. Right-click on the Object. The context menu displays.
2. If Instance Variables are supported, select the Advanced | Set Run State menu option (or press [Ctrl]+[Shift]+[R]). The Set Run State dialog displays.

Variable | Operator | Value | Note
--- | --- | --- | ---
| | | | |
3. In the **Variable** field, click on the drop-down arrow and select the variable, or type in the new variable name.

4. Set the **Operator**, the **Value** and optionally type in a **Note**.

5. Click on the **OK** button to save the variable.

### 15.2.2.14.1.2 Remove a Defined Variable

To delete a **run-time state** variable for an Object:

1. Right-click on the required Object. The context menu displays.
2. Select the **Set Run State** option. The **Run State** dialog displays.
3. In the **Variable** field, click on the drop-down arrow and select the variable to delete.
4. Clear the **Value** field.
5. Click on the **OK** button.

### 15.2.2.14.1.3 Define an Object State

To set the Object state for a Class instance, follow the steps below:

1. Right-click on the required Object and select the **Advanced | Set Object State** menu option. The **Set Instance State** dialog displays.

2. In the **State** field, type the required state (e.g. **Awaiting Approval**).

    **Note:**

    If the associated classifier has a child State Machine element, those states propagate into the drop-down list for this field, and you can select one of them instead.

3. Click on the **OK** button to apply the state. The object now shows the run-time state in square brackets below the object name.

### 15.2.2.15 Package

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, you can drag these Packages onto a diagram workspace (most diagram types, both standard and extended) for structural or relational depictions, including Package imports or merges.
OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 109) 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.

15.2.2.16 Part

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 is not composed within it.

You can also set property values for Parts.

15.2.2.16.1 Add Property Value

To add property value variables to a Part, follow the steps below:

1. Right-click on the Part. The context menu displays.
2. Select the Advanced | Set Property Values menu option (or press [Ctrl]+[Shift]+[R]). The Set Property Values dialog displays.

3. In the Variable field, click on the drop-down arrow and select the variable, or type in the new variable name.
4. Set the **Operator**, the **Value** and optionally type in a **Note**.
5. Click on the **OK** button to save the variable.

A Part with a property value resembles the following figure.

![Part Beta](image)

**15.2.2.17 Port**

Ports define the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted using the **interface element**. Any connector to a Port must provide the required interface, if defined. Ports can appear on a contained Part, a **Class**, or the boundary of a **Composite element**. A Port is a **typed** structural feature or property of its containing classifier. Ports are typically **created** in **Class diagrams**, **Object diagrams**, and **Composite Structure diagrams**.

You can **expose an inherited Port**, or **redefine a Port**.

**Toolbox Icon**

![Port Icon](image)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 182) states:

A port is a property 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.

**15.2.2.17.1 Add a Port to an Element**

To add a new Port to an element, use one of the following steps:

1. Click on the **Port** symbol in the Composite Elements page of the Enterprise Architect UML Toolbox and drag it to (or click on) the target host element. This creates an untyped, simple Port on the boundary, near the cursor position.
2. On the context menu of a suitable Class, Part or **Composite element** select the Embedded Elements | **Add Port** menu option to add a new Port at the cursor position.
3. Drag a suitable classifier from the Project Browser onto a Class or Part. Enterprise Architect prompts you to add a typed Port or Part at the cursor position. The new Port is typed by the original dragged classifier.
4. Use the Embedded Elements dialog to add a new Port to the currently selected element.
15.2.2.17.2 Inherited and Redefined Ports

A Port is a redefinable and re-useable property of a composite classifier. So, as for attributes, any Class can inherit 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 window the inherited Ports and their named owners are listed there.

It is possible to expose, for design purposes, an inherited Port (i.e. the child Class is re-using the parent Port). In this case, Enterprise Architect creates a clone of the re-used Port and marks 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 parent elements.

It is also possible to redefine a Port in a child Class, so that the name is the same but the child is a modifiable clone of the original. This is useful where a child Class places additional restrictions or behavior on the Port. The Embedded Elements window enables you to highlight an inherited Port and mark it as redefined; this creates a new Port on the child Class, which is editable but still logically related to the initial Port.

The Embedded Elements window 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 Port firewire has been added to PC. If any of the inherited Ports are made visible, they are considered re-use Ports and appear on the child in read-only format. Using the Redefine button, the inherited Port can be copied down and made writeable.

15.2.2.18 Primitive

A Primitive element identifies a predefined data type, without any relevant substructure (i.e. it has no parts in the context of UML).
OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 124) states:

A primitive data type may have an algebra and operations defined outside of UML, for example, mathematically ... The run-time instances of a primitive type are data values. The values are in many-to-one correspondence to mathematical elements defined outside of UML (for example, the various integers). Instances of primitive types do not have identity. If two instances have the same representation, then they are indistinguishable.

Primitive for MOF Diagrams

Drag and drop a Primitive element from the Metamodel Elements page of the Enterprise Architect UML Toolbox; the following dialog displays.

Enter the Primitive name and type, and any notes required. The following Types are available, as defined in the MOF specification.

15.2.2.19 Qualifiers

A Qualifier is a property of an Association that limits the nature of the relationship between two classifiers or objects.

Some examples of qualified associations are shown in the following diagram:
Qualifiers are set in the Source Role tabs and Target Role tabs of the Connector Properties dialog.

**Notes:**

- Separate multiple Qualifiers with a semi-colon; each Qualifier then displays on a separate line. For example, in the diagram the Qualifier 'rank:Rank;file:File' has been rendered in two lines, with a line break at the ; character.
- You can enable or disable Qualifier rectangles in the Diagram page of the Options dialog (select the Tools | Options | Diagram menu option). 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.
- You can enable or disable a mild shading on the Qualifier rectangles in the Links page of the Options dialog.

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 129) 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.
15.2.2.20 Signal

A Signal is a specification of Send request instances communicated between objects, typically in a Class or Package diagram. The receiving object handles the Received request instances as specified by its receptions. The data carried by a Send request is represented as attributes of the Signal. A Signal is defined independently of the classifiers handling the signal occurrence.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 450) states:

A signal triggers a reaction in the receiver in an asynchronous way and without a reply. The sender of a signal will not block waiting for a reply but continue execution immediately. By declaring a reception associated to a given signal, a classifier specifies that its instances will be able to receive that signal, or a subtype thereof, and will respond to it with the designated behavior.
15.2.3 Inbuilt and Extension Stereotypes

There are many other UML elements that you can also work with in Enterprise Architect, most of which are basic elements extended by the use of stereotypes. This topic gives a brief introduction to some of these elements.

- Analysis Stereotypes
- Boundary Element
- Composite Elements
- Control Element
- Entity Element
- Event Elements
- Hyperlinks
- N-Ary Association
- Process
- Requirements
- Screen
- Table
- UI Control Element
- Web Stereotypes

For more information on the use of stereotypes in Enterprise Architecture, see the UML Stereotypes topic.

15.2.3.1 Analysis Stereotypes

Enterprise Architect has some built in stereotypes that you can assign to an element during analysis. The effect of these stereotypes is to display a different icon from the normal element icon, providing a visual key to the element purpose. The Robustness diagram below illustrates the main types of inbuilt icons for elements:

- **Boundary** - for a system boundary (e.g. a Login screen)
- **Control** - to specify an element is a controller of some process (as in the Model-View-Controller pattern)
- **Entity** - the element is a persistent or data element

Also see the Business Modeling elements, used in Business Modeling and Business Interaction diagrams.
15.2.3.2 **Boundary**

A **Boundary** is a stereotyped **Object** that models some system boundary, typically a user interface screen. You can also create a Boundary as a stereotyped **Class**. See the [Create a Boundary](#) topic.

A Boundary 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').

---

### Toolbox Icon

![Boundary Icon](#) or ![Boundary Icon](#)

#### 15.2.3.2.1 Create a Boundary

**Using the Toolbox**

To create a **Boundary** element on a diagram as an Object, follow the steps below:

1. In the Enterprise Architect UML Toolbox, select the **More Tools | UML | Analysis** menu option.
2. From the **Analysis Elements** page, drag the **Boundary** element onto the diagram.

**Using the Properties Dialog**

To create a Boundary element as a stereotyped Class, using the Class **Properties** dialog, follow the steps below:

1. Insert a new Class.
2. Right-click on the element and select the **Properties** menu option; the Properties dialog displays.
3. In the **Stereotype** field, type the value **boundary**.
4. Click on the **Apply** and **OK** buttons.
5. Save the diagram ([Ctrl]+[S]).

The following illustration shows an **Actor** interacting with a Boundary (in this case, a Login screen).
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.

15.2.3.3 Composite Elements

Enterprise Architect supports Composite elements for Classes, Objects, Use Cases and such. A Composite element is a pointer to a child diagram.

Create a Composite Element

To set Composite elements from the element context menu, follow the steps below:

1. Create the element to set as a Composite element.
2. Right-click on the element in the diagram and select the Advanced | Make Composite context menu option.

**Note:**

If the Make Composite option is not listed in the context menu, the option is not available for the type of element you have selected.

The element displays as follows:

```
Class1
```

Note the small icon in the bottom right hand corner indicating that this is now a Composite element.

3. Double-click on 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:

```
Class1
└── ClassX
    └── ClassY
    └── ClassZ
```

Note that ClassX, ClassY and ClassZ are elements in the child diagram.
**Alternative Notation**

Composite elements might show their contents instead of their usual notation. To enable this notation, right-click on the element to open the context menu, then select the **Advanced | Show Composite Diagram** option.

**The Automation Interface**

Automation support is available for Composite elements. *Element* has an *Elements* collection and a *Diagrams* collection.

### 15.2.3.4 Control

A **Control** is a stereotyped **Object** that models a controlling entity or manager. A Control organizes and schedules other activities and elements, typically in **Analysis** (including Robustness), **Sequence** and **Communication** diagrams. It is the controller of the **Model-View-Controller** pattern.

You can also create a Control as a stereotyped **Class**. See the **Create a Control Element** topic.

**Toolbox Icon**

![Control Icon](image)

**15.2.3.4.1 Create a Control Element**

**Using the Toolbox**

To create a **Control** element on a diagram as an Object, follow the steps below:

1. In the Enterprise Architect UML Toolbox, select the **More Tools | UML | Analysis** menu option.
2. From the **Analysis Elements** page, drag the **Control** element onto the diagram.

**Using the Properties Dialog**

To create a Control element as a stereotyped Class, using the Class Properties dialog, follow the steps below:

1. Insert a new Class.
2. Right-click on the element and select the **Properties** menu option; the Properties dialog displays.
3. In the **Stereotype** field, type the value **control**.
4. Click on the **Apply** and **OK** buttons.
5. Save the diagram (**Ctrl**+**S**).

The appearance changes as illustrated in the following diagram (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.

15.2.3.5 Entity

An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system. It is the Model in the Model-View-Controller pattern. You can also create an Entity as a stereotyped Class. See the Create an Entity topic.

Toolbox Icon

Using the Toolbox
To create an Entity element on a diagram as an Object, follow the steps below:
1. In the Enterprise Architect UML Toolbox, select the More Tools | UML | Analysis menu option.
2. From the Analysis Elements page, drag the Entity element onto the diagram.

Using the Properties Dialog
To create an Entity element as a stereotyped Class, using the Class Properties dialog, follow the steps below:
1. Insert a new Class.
2. Right-click on the element and select the Properties menu option; the Properties dialog displays.
3. In the Stereotype field, type the value entity.
4. Click on the **Apply** and **OK** buttons.
5. Save the diagram ([Ctrl]+[S]).

### 15.2.3.6 Event

The UML includes two elements that are used to model **Events**. The first 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, either within the system or external to the system.

![Send and Receive Events Diagram]

The second element is the **Receive Event**, which 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 invokes further activities and processing.

Send and Receive Events can be added from the **Analysis** and **Activity** Element pages of the Enterprise Architect UML Toolbox.

If you should select the wrong type of event, or otherwise want to change the type, right-click on the Event and select the **Advanced | Make Sender** or **Advanced | Make Receiver** menu option, as appropriate.

#### Toolbox Icons

- ![Send Event Icon]
- ![Receive Event Icon]

### 15.2.3.7 Feature

**Feature** is a small, granular function or characteristic expressed in client-valued terms as a satisfaction of a requirement; for example: 'context-sensitive Help', or 'ability to reverse-engineer VB.Net'.

Features are the primary requirements-gathering artifact of the **Feature-Driven Design (FDD)** methodology. They define the product feature that satisfies what a **Requirement** element has formalized as a contractual, testable, expected deliverable (for example: requirement - 'every element must provide context-sensitive Help'; feature - 'every element provides context-sensitive Help'). One Feature might realize one or more Requirements, and one Requirement might be realized by more than one Feature.

Features also have relationships with **Use Cases**. A Use Case defines the interaction a user has with the system in order to satisfy one or more Requirements. The Feature identifies the facility that provides the means for that interaction.

Feature elements are non-UML and are not related to UML-defined features, which are either **BehavioralFeatures** (operations or methods) or **StructuralFeatures** (Ports, Parts, and attributes).

Feature elements are available from the **Requirements** page of the Enterprise Architect UML Toolbox.

**Note:**

Feature elements can be created with or without an identifying **F** in the top right corner of the element. To toggle the display of this letter, select or deselect the **Show stereotype icon for requirements** checkbox on the Options dialog, **Objects** page.
15.2.3.8 Hyperlinks

You can place a Hyperlink element onto a diagram. This element is a type of text element, but one that can contain a pointer to a document file or web address. When you double-click on the element, Enterprise Architect executes the related file or address. You can connect diagrams to associated files, web pages, Help and even other Enterprise Architect model files. To add a Hyperlink element, click on the Hyperlink icon in the Elements toolbar and then click on the diagram.

Note:
Once you have created the hyperlink, as described in the sections below, you can edit the hyperlink text on the diagram by clicking once on the field, once on the text, then right-clicking and selecting the Edit Selected context menu option.

Configure the Hyperlink

Create Hyperlinks to Enterprise Architect Views

Hyperlinks can also be added as a Hyperlink object in a diagram with various other targets, such as a Matrix Profile or Help topic. The target opens within the Enterprise Architect work area, as a view. In the Hyperlink Address field:

- For a Matrix Profile, use $matrix:// followed by the name of the profile (e.g. $matrix://MyProfile).
- For a Help topic, use $help:// followed by the name of the Help topic file (e.g. $help://ShipDiag.htm)
- For a search, use $search://Name=Searchtype;Term=Searchterm; (e.g. $search://Name=Changes - In Progress;Term=High Priority.)
- To open the discussion forum, use $forum://
• To access an internet facility use $inet://$ followed by the http address (e.g. $inet://http://www.google.com/)

**Note:**

The difference between this $inet Hyperlink and the direct (http://) link is that whilst the $inet link opens the target as an Enterprise Architect view, the direct link opens the target as a new window separate from Enterprise Architect.

Create Hyperlink To File

You can also create a hyperlink on a diagram to an external file very simply, by clicking on the file in a file list (such as Windows Explorer) or on your Desktop and dragging it onto the diagram. A short context menu displays with two options - Hyperlink and Artifact. Click on the Hyperlink option to create the hyperlink on the diagram. The link is effective immediately, and you can right-click on it to define its properties as shown above.

Create Hyperlinks Between Diagrams

To create Hyperlinks between diagrams, follow the steps below.

**Note:**

If the Hyperlinks appear as Sub Activities, select the Tools | Options | Diagram | Behavior menu option and deselect the Use Automatic SubActivities checkbox.

1. Open the diagram in which to display a Hyperlink to another diagram. From the Project Browser select the diagram you want to create a Hyperlink to.

2. Drag the diagram on to the current diagram. The Select Type dialog displays.
3. Select the **Hyperlink** option and click on the **OK** button. The final hyperlinked diagram should resemble the diagram below, where the *Class* diagram is the diagram to which the *Product Order* diagram hyperlinks.

![Hyperlinked Diagram](image)

### 15.2.3.9 N-Ary Association

An *n*-Ary *Association* element is used to model complex relationships between three or more elements, typically in a *Class diagram*. It is not a commonly-employed device, but can be used to good effect where there is a dependant relationship between several elements. It is generally used with the *Associate* connector, but the relationships can include other types of connector.
In the example above there is a relationship between a Company, an Employee and a Salary.

**Toolbox Icon**

![Association](image)

**15.2.3.10 Process**

A Process is an Activity element with the stereotype process, which expresses the concept of a business process. Typically this involves inputs, outputs, work flows, goals and connections with other Processes. The Process element is typically used in Analysis diagrams.

Business processes typically range across many parts of the organization and span one or more systems.

**Toolbox Icon**

![Process](image)

**15.2.3.11 Requirements**

As an analysis step, often it is desirable to capture simple system requirements. These are eventually realized by Use Cases.

In the initial requirement gathering phase, cataloging requirements can be achieved using the Requirement extension on a Custom diagram.
Requirements can 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 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 easily model quite detailed and complex dependencies and implementation relationships.

Notes:
- External requirements can be created with or without an identifying E in the top right corner of the element. To toggle the display of this letter, select or deselect the Show stereotype icon for requirements checkbox on the Options dialog, Objects page.
- The colors on Requirement elements identify the status of the requirement. You change the status - and hence color - on the element Properties dialog. You set the color for each status on the Status Types dialog.
15.2.3.12 Screen

A Screen is used to prototype User Interface screen flow. By using UML features such as requirements, constraints, and scenarios against User Interface diagram 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 has from a user perspective, and in conjunction with the Use Case model, defines exactly how a user gets work done.

Web pages can also be prototyped and specified rigorously using Enterprise Architect's custom interface extensions.

The example diagram below illustrates some features of Enterprise Architect'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:

Enterprise Architect displays UI Controls as a range of special icons, depending on the stereotype used; for example, a Control stereotyped as a «list» displays with a vertical scroll bar.

---

Toolbox Icon

- Requirement

- Screen
15.2.3.13 Table

A Table is a stereotyped Class element. It is drawn with a small table icon in the upper right corner. You typically use this element in Data Modeling and Class diagrams.

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 do not have any data types to choose from when creating columns.

Toolbox Icon

Table

15.2.3.14 Test Case

A Test Case is a stereotyped Use Case element. You might use it to extend the facilities of the Testing window by applying element properties and capabilities to the tests of a feature represented by another element or - more appropriately - set of elements. That is, you can define once - in the Testing window for the Test Case element - the details of the tests that apply to each of several elements, instead of recording the details separately in each element.

Within the Test Case element properties you can define test requirements and constraints, and associate the test with test files. You can also link the element to Document Artifacts or (in the Corporate edition) directly to linked documents, such as a Test Plan.

The Test Case element enables you to give greater visibility to tests, in the Project Browser, Element List, Model Search, Relationship Matrix, Hierarchy window and reports.

15.2.3.15 UI Control Element

A UI Control element represents any user interface control element (e.g. edit box). It is used for capturing the components of a screen layout and requirements in a Custom or User Interface diagram.

UI Controls can be drawn in a variety of ways depending on the element stereotype:
### Toolbox Icon

![UI Control Icon](image)

#### 15.2.3.15.1 Create A UI Control Element

Create a UI Control element with the required representation, follow the steps below:

1. Create a User Interface diagram.
2. Drag a UI Control element from the User Interface Elements page of the Enterprise Architect UML Toolbox onto the diagram. The GUIElement: UI Control dialog displays.
3. In the **Stereotype** field, type the required stereotype name or click on the drop-down arrow and select the appropriate stereotype.

**Note:**
Initially, not every possible stereotype name is listed in the **Stereotype** field drop-down list; they are added as you type them into the field.

4. In the **Name** field, type an appropriate name for the element, then click on the **Apply** button.

### 15.2.3.16 Web Stereotypes

Enterprise Architect supports a number of stereotypes for web page modeling, the graphical elements for which display with a **graphical icon** instead of the usual « **stereotype** » format. These stereotypes are only supported for **Class** elements. The image below indicates the various graphical icons and their associated stereotypes.
A similar set of web modeling elements and their relationships are also available through dedicated Web Modeling pages in the Enterprise Architect UML Toolbox.

**Set a Web Icon**

To set a web icon, follow the steps below:

1. Create a new Class element in a diagram.
2. Display the Class Properties dialog.
3. In the **Stereotype** field, either type in the required stereotype name or click on the drop-down arrow and select the required stereotype (as named above).
4. Click on the **OK** button. The Class displays as in one of the examples above.
## 15.3 UML Connectors

### What is a Connector?

A connector is a logical or functional relationship between model elements. There are several different connector types, each having a particular purpose and syntax. Enterprise Architect supports all of the UML connectors as well as some custom ones of its own. Together with the UML Elements, these form the basis of UML models.

For more information on using these connectors, consult the appropriate topic by clicking on the required connector icon in the table below.

<table>
<thead>
<tr>
<th>Behavioral Diagram Connectors</th>
<th>Structural Diagram Connectors</th>
<th>Inbuilt and Extended Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Activity Diagrams</strong></td>
<td><strong>Composite Structure Diagrams</strong></td>
<td><strong>Analysis Diagrams</strong></td>
</tr>
<tr>
<td>Control Flow</td>
<td>Connector</td>
<td>Information Flow</td>
</tr>
<tr>
<td>Object Flow</td>
<td>Assembly</td>
<td>Object Flow</td>
</tr>
<tr>
<td>Interrupt Flow</td>
<td>Delegate</td>
<td>Associate</td>
</tr>
<tr>
<td></td>
<td>Role Binding</td>
<td>Realize</td>
</tr>
<tr>
<td>Use Case Diagrams</td>
<td></td>
<td>Representation</td>
</tr>
<tr>
<td>Use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Include</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extend</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Realize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invokes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precedes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Diagrams</td>
<td>Package, Class and Object Diagrams</td>
<td></td>
</tr>
<tr>
<td>Transition</td>
<td>Associate</td>
<td>Dependency</td>
</tr>
<tr>
<td>Object Flow</td>
<td>Generalize</td>
<td>Realize</td>
</tr>
<tr>
<td></td>
<td>Compose</td>
<td>Information Flow</td>
</tr>
<tr>
<td></td>
<td>Aggregate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Association Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assembly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dependency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Realize</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Information Flow</td>
<td></td>
</tr>
</tbody>
</table>

**Common Connectors**

- Dependency
- Realize
- Trace
- Information Flow
- Note Link
- Profile
- Extension
- Generalize
- Application
<table>
<thead>
<tr>
<th>Behavioral Diagram Connectors</th>
<th>Structural Diagram Connectors</th>
<th>Inbuilt and Extended Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timing Diagrams</strong></td>
<td><img src="image" alt="Nesting" /></td>
<td><img src="image" alt="Tagged Value" /></td>
</tr>
<tr>
<td><img src="image" alt="Message" /></td>
<td><img src="image" alt="Package Merge" /></td>
<td><img src="image" alt="Redefinition" /></td>
</tr>
<tr>
<td><img src="image" alt="Package Import" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sequence Diagrams</strong></td>
<td><img src="image" alt="Nesting" /></td>
<td><img src="image" alt="Metamodel" /></td>
</tr>
<tr>
<td><img src="image" alt="Message" /></td>
<td><img src="image" alt="Assembly" /></td>
<td><img src="image" alt="Associate" /></td>
</tr>
<tr>
<td><img src="image" alt="Self-Message" /></td>
<td><img src="image" alt="Delegate" /></td>
<td><img src="image" alt="Compose" /></td>
</tr>
<tr>
<td><img src="image" alt="Recursion" /></td>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="Aggregate" /></td>
</tr>
<tr>
<td><img src="image" alt="Call" /></td>
<td><img src="image" alt="Realize" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Generalize" /></td>
<td><strong>Component Diagrams</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Communication Diagrams**" /></td>
<td><img src="image" alt="Assembly" /></td>
<td><img src="image" alt="Custom" /></td>
</tr>
<tr>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="Delegate" /></td>
<td><img src="image" alt="Associate" /></td>
</tr>
<tr>
<td><img src="image" alt="Realize" /></td>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="Aggregate" /></td>
</tr>
<tr>
<td><img src="image" alt="Nesting" /></td>
<td><img src="image" alt="Generalize" /></td>
<td><img src="image" alt="Generalize" /></td>
</tr>
<tr>
<td><img src="image" alt="Realize" /></td>
<td><img src="image" alt="Realize" /></td>
<td><img src="image" alt="Realize" /></td>
</tr>
<tr>
<td><img src="image" alt="Nesting" /></td>
<td><img src="image" alt="Deployment Diagrams" /></td>
<td><img src="image" alt="Requirements" /></td>
</tr>
<tr>
<td><img src="image" alt="Control Flow" /></td>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="Aggregate" /></td>
</tr>
<tr>
<td><img src="image" alt="Object Flow" /></td>
<td><img src="image" alt="Generalize" /></td>
<td><img src="image" alt="Inheritance" /></td>
</tr>
<tr>
<td><img src="image" alt="Interrupt Flow" /></td>
<td><img src="image" alt="Realize" /></td>
<td><img src="image" alt="Associate" /></td>
</tr>
<tr>
<td><img src="image" alt="Fork/Join" /></td>
<td><img src="image" alt="Deployment" /></td>
<td><img src="image" alt="Implements" /></td>
</tr>
<tr>
<td><img src="image" alt="Fork/Join" /></td>
<td><img src="image" alt="Manifest" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Object Flow" /></td>
<td><img src="image" alt="Nesting" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Nesting" /></td>
<td><strong>Deployment Diagrams</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Maintenance" /></td>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="WSDL" /></td>
</tr>
<tr>
<td><img src="image" alt="Aggregate" /></td>
<td><img src="image" alt="Aggregate" /></td>
<td>No special connectors</td>
</tr>
<tr>
<td><img src="image" alt="Generalize" /></td>
<td><img src="image" alt="Realize" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="Realize" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Data Modeling" /></td>
<td><img src="image" alt="Generalize" /></td>
<td>No special connectors</td>
</tr>
<tr>
<td><img src="image" alt="Associate" /></td>
<td><img src="image" alt="Realize" /></td>
<td></td>
</tr>
</tbody>
</table>
15.3.1 Aggregate

An Aggregation connector is a type of association that shows that an element contains or is composed of other elements. It is used in Class models, Package models, and Object models to show how more complex elements (aggregates) are built from a collection of simpler elements (component parts; e.g. a car from wheels, tires, motor and so on).

A stronger form of aggregation, known as Composite Aggregation, is used to indicate ownership of the whole over its parts. The part can belong to only one Composite Aggregation at a time. If the composite is deleted, all of its parts are deleted with it.

After drawing an Aggregation association, its form can be changed.

Toolbox Icon

15.3.1.1 Change Aggregation Connector Form

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 connector from weak to strong, follow the steps below.

1. Right-click on an Aggregation connector to display the context menu.
2. Select Set Aggregation to Composite. The diamond is shown as filled.

Note:
If the connector is already a Strong (Composition) connector, the context menu option changes to Set Aggregation to Shared.
15.3.2 Assembly

An Assembly connector bridges a component's required interface (Component1) with the provided interface of another component (Component2), typically in a Component diagram.

**Toolbox Icon**

Assembly

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 156) 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.
15.3.3 **Associate**

An **Association** implies two model elements have a relationship, usually implemented as an instance variable in one **Class**. This connector can include named roles at each end, multiplicity, direction and constraints. **Association** is the general relationship type between elements. To connect more than two elements in an association, you can use the **N-Ary Association** element.

When code is generated for **Class diagrams**, Associations become instance variables in the target Class. The relationship is also used in **Package**, **Object**, **Communication** and **Deployment** diagrams.

**Toolbox Icon**

![Associate Icon](image)

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 41*) 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.*

An *end property of an association that is owned by an end class or that is a navigable owned end of the association indicates that the association is navigable from the opposite ends; otherwise, the association is not navigable from the opposite ends.*
15.3.4 Association Class

An Association Class connector is a UML construct that enables an Association connector to have attributes and operations (features). This results in a hybrid relation with the characteristics of a connection and a Class. It is used to model particular types of connections in UML (see the OMG UML Specification for more details).

When you add an Association Class connection, Enterprise Architect also creates a Class that is automatically connected to the Association. When you hide or delete the Association, the Class is also hidden or deleted.

To add an Association Class to a Class or Deployment diagram, click on the Association Class icon in the Enterprise Architect UML Toolbox. Click and hold on the source object in the diagram while you drag the line to the target element, then release the mouse button. Enterprise Architect draws the connector and adds the Class, then prompts you to add the Class name. Note that the names of the Class and the connector are the same. You can also connect a new Class to an existing Association.

The following diagram illustrates an Association Class between model elements. Note the dotted line from the Class to the Association. You cannot move or delete this line.

**Toolbox Icon**

![Association Class Icon]

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 49) 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.

15.3.4.1 Connect New Class to Association

To connect a new Class to an existing Association, follow the steps below:

1. Create a Class in the diagram containing the Association to connect.
2. Right-click on the new Class. The context menu displays.
3. Select the Advanced | Association Class menu option. The Create Association Class dialog displays.

Select Association to Link To:
- Association from Customer to CustomerAccount
- Association from Customer to CustomerPreferences
- Association from CustomerPreferences to ShippingDetails

4. Select the connector to connect to.
5. Click on the OK button.
15.3.5 Communication Path

A Communication Path defines the path through which two Deployment Targets are able to exchange signals and messages. Communication Path is a specialization of Association. A Deployment Target is the target for a deployed Artifact and can be a Node, Property or Instance Specification in a Deployment diagram.

Toolbox Icon

Communication Path
15.3.6 Compose

A Composite Aggregation is used to depict an element that is made up of smaller components, typically in a Class or Package diagram. 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 the example below.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 43) 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.
15.3.7 **Connector**

Connectors illustrate communication links between parts to fulfill the structure's purpose, typically in a Composite Structure diagram. 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.

**Toolbox Icon**

![Connector Icon](image)

**OMG UML Specification**

The OMG UML specification *(UML Superstructure Specification, v2.1.1, p. 177)* 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 or slots, 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.*
15.3.8 Control Flow

The Control Flow is a connector connecting 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 can define a Guard and a Weight condition. A Guard defines a condition that must be true before control passes 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 that is exclusive of the other and defines which edge is taken under what conditions. The Control Flow Properties dialog enables you to set up Guard conditions on Control Flows and on Object Flows.

A Weight defines the number of tokens that can flow along a Control or Object Flow connection when that edge is traversed. Weight can also be defined on the Control Flow and Object Flow Properties dialogs.

OMG UML specification:

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 356) states:

A control flow is an edge that starts an activity node after the previous one is finished.
15.3.9 Delegate

A Delegate connector defines the internal assembly of a component's external Ports and Interfaces on a Component diagram. Using a Delegate connector wires the internal workings of the system to the outside world, by a delegation of the external interfaces' connections.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 156) 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.
15.3.10 Dependency

Dependency relationships are used to model a wide range of dependent relationships between model elements in Use Case, Activity, and Structural diagrams, and even between models themselves. You can create the Dependency from the Common page of the Enterprise Architect UML Toolbox. The Dependencies package as defined in UML 2.1 has many derivatives, such as Realization, Deployment, and Use. Once you create a Dependency you can further refine its meaning by applying a specialized stereotype.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 64) 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).

15.3.10.1 Apply a Stereotype

To apply a stereotype to a Dependency relationship, follow the steps below:

1. Select the Dependency relationship to change.
2. Right-click on the connector and, from the context menu, select the Dependency Properties option. The Dependency Properties dialog displays.
3. In the Stereotype field, either type in the required stereotype name or click on the drop-down arrow and select the stereotype from the list.
4. Click on the OK button.

Alternatively, you can right-click on the Dependency relationship and select the Advanced | Dependency Stereotypes menu option, then select from a shorter list of standard stereotypes.
15.3.11 Deployment

A Deployment is a type of Dependency relationship that indicates the deployment of an artifact onto a node or executable target, typically in a Deployment diagram. 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.

Toolbox Icon

Deployment
15.3.12 Extend

An Extend connection is used to indicate that an element extends the behavior of another. Extensions are used in Use Case models to indicate that one Use Case (optionally) extends the behavior of another. An extending Use Case often expresses alternative flows.

Toolbox Icon

![Extend Toolbox Icon](image)

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 587) 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.
15.3.13 Generalize

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. It is used typically in Class, Component, Object, Package, Use Case, and Requirements diagrams.

Toolbox Icon

A Generalization icon 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. It is used typically in Class, Component, Object, Package, Use Case, and Requirements diagrams.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 73) 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.
15.3.14 Include

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.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 591) states:

Include is a DirectedRelationship between two Use Cases, implying that the behavior of the included Use Case is inserted into the behavior of the including Use Case. It is also a kind of NamedElement so that it can have a name in the context of its owning 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.
15.3.15 Information Flow

An Information Flow represents information items or classifiers flowing between two elements in any diagram. The connector is available from the Common toolbox page and from every Quick Link menu.

You can have more than one Information Flow connector between the same two elements, identifying which items flow between the two under differing conditions.

Example of Use

1. Open a diagram and add two elements (for example, Nodes on a Deployment diagram).
2. Click on the Information Flow connector in the Common toolbox page and drag between the two elements. The Information Items Conveyed dialog displays.
3. Add the classifier or information item element(s) to the Information Flow. The diagram now resembles the following.

4. Add another connector between the same two elements (for example, a Communication Path connector).
5. Right-click the connector and select the Advanced | Information Flows Realized context menu option. The Information Flows Realized dialog displays.
6. Tick the checkbox against the required classifier element and click on the OK button. The combined connector now resembles the following:

Notes:

- Once the connectors are combined, you cannot access the Information Items Conveyed dialog directly. You add or delete information items on the connector using the Information Items Realized dialog. If you have more than one Information Flow connector between the elements, they form part of the same combined connector; you can again work on them separately through the Information Items Realized dialog.
- If you have information flows in a diagram that you use as the source for a Pattern, the Information Items Conveyed and Information Flows Realized data is not copied into the Pattern.
The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 606) states:

An InformationFlow specifies that one or more information items circulates from its sources to its targets.

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 607) also 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.

### 15.3.15.1 Convey Information on a Flow

As you create an Information Flow connector between two elements, you can specify which Information Items or classifiers are conveyed on this flow.

To specify these Information Items or classifiers, right-click on the connection and select Advanced | Information Item Conveyed. The Information Items Conveyed dialog displays.

#### Button | Use to
--- | ---
Add | Display the Set Element Classifier dialog, from which you select the required Information or Classifier element(s).
Remove | Remove the selected item.

**Note:**

If you select more than one element, they are listed in one entry for the Information Flow connector.
15.3.15.2 Realize an Information Flow

The Information Flows Realized dialog displays all flows that can be realized on the selected connector. To realize an Information Flow on this connector, select the corresponding checkbox and click on the OK button.

If you want to change the information items conveyed on an information flow, click on the flow text (as for Use Case2, above) and click on the Items button. The Information Items Conveyed dialog displays, and you can add or remove items as required. When you click on the OK button, the Information Items Realized dialog redisplays and you can realize the selected flow or flows as above.
15.3.16 Interrupt Flow

The Interrupt Flow is a connection 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*. It is typically used in an *Activity diagram*.

**Toolbox Icon**

![Interrupt Flow Icon](image)

**OMG UML Specification**

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 327*) states:

*An activity edge is an abstract class for directed connections between two activity nodes.*
15.3.17 Manifest

A Manifest relationship indicates that the Artifact source embodies the target model element, typically in Component and Deployment diagrams. Stereotypes can be added to Enterprise Architect to classify the type of manifestation of the model element.

Toolbox Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 212) 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.
### 15.3.18 Message

*Messages* indicate a flow of information or transition of control between elements. Messages can be used by *Timing Diagrams*, *Sequence Diagrams*, and *Communication Diagrams* but not *Interaction Overview* diagrams to reflect system behavior. If between *Classes* or classifier instances, the associated list of operations is available to specify the event.

Moving a Message can disrupt the organization of other features on the diagram. To avoid this, and move *only* the Message, press `[Alt]` while you move the Message.

#### Toolbox Icon

![Message Icon]

#### OMG UML Specification

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 491*) 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, for example, raising a signal, invoking an Operation, creating or destroying an Instance. The Message specifies not only the kind of communication given by the dispatching ExecutionSpecification, but also the sender and the receiver.*


#### Note:

Communication diagrams were known as Collaboration diagrams in UML 1.4.

### 15.3.18.1 Message (Sequence Diagram)

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, and can be of many different *types*. 

![Sequence Diagram Example]
To create a Message on a Sequence diagram, follow the steps below:

2. In the Interaction Relationships page, click on the Message icon, click on the source object and drag the cursor to the destination (target) object. The Message Properties dialog displays (if not, right-click on the Message and select the Message Properties menu option).

3. In the Message field, type the Message name.

   **Note:**

   If the Message flow is towards a Class element (dropped in from a Class diagram) or a Lifeline element having a classifier, and the destination Class has defined operations, you can click on the drop-down arrow and select an appropriate operation name. The Message then reflects the destination Class operations.

   If the available operations are not appropriate, you can click on the Operations button and define a new operation in the target element, using the Operations dialog.

   If you create a Message without making reference to the target Class operations, no new operation is added to the target Class.

4. In the Parameters field, type any parameters that the Message has, as a comma-separated list. If required, in the Parameter Values field type the actual value for each parameter, again as a comma-separated list.

5. If the Message is a return message, in the Return Value field enter the returned value or type.
Note:
It is possible to depict returns from a Self Message. Simply create a second Self Message at the end of execution and select the Is Return checkbox in the Control Flow Type panel.

6. If the Message flow is from a Class element or Lifeline element with classifier that has defined attributes, click on the drop-down arrow in the Assign to field and select an appropriate attribute name. The Message reflects the attributes from the source Class. You cannot add further attributes to the source Class here - if no appropriate attribute is listed, open the element Properties dialog and add the required attribute.

Otherwise, if required, type the name of the object to assign the message flow to.

7. In the Stereotype field, type or select an optional stereotype for the connector (this is displayed on the diagram, if entered).

8. If required, in the Alias field type an alias for the name of the Message.

Note:
On the diagram, the alias displays if the Use Alias if Available checkbox is selected on the Diagram tab of the Diagram Properties dialog. The Alias displays instead of or as well as the Message name, depending on the setting selected in the Alias Usage panel of the Diagram Behavior page of the Options dialog.

9. In the Condition field, type any conditions that must be true in order for the message to be sent.

10. In the Synch: field in the Control Flow Type panel, select Synchronous or Asynchronous as appropriate.

11. In the Lifecycle field, select New to create a new element at the end of the Message, or Delete to terminate the message flow at the end of the Message. If neither case applies, leave the field at the default of <none>.

12. If required, in the Notes field type any explanatory notes. You can format the notes using the Rich Text Notes toolbar at the top of the field.

13. Click on the OK button to save the Message definition.

Note:
You can change the timing details of a message on the Timing Details dialog, and emphasize the sequence of closely-ordered messages using General Ordering.

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 menu option. There are four sub-options available:

- Start at head
- End at head
- Start at tail
- End at tail

Toolbox Icon

Message
15.3.18.1.1 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 typically in a Sequence diagram.

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. Double-click on the Message name to open the Message Properties dialog.
4. Raise the Activation level of the return.

Self-Message Calls indicate a nested invocation; new activation levels are added with each Call.

Toolbox Icon

🔗 Self-Message
15.3.18.1.2 Call

A Call is a type of Message connector 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.

Toolbox Icon

15.3.18.1.3 Change the Timing Details

It is possible to change the timing details of a Message in a Sequence diagram by right-clicking on the Message connector and selecting the Timing Details menu option. The Timing Details dialog displays.

Complete the fields on this dialog as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration Constraint</td>
<td>Indicate the minimum and maximum limits on how long a message can last.</td>
</tr>
<tr>
<td>Duration Constraint Between Messages</td>
<td>Indicate the minimum and maximum interval between sending or receipt of the previous message at the current message's source Lifeline, and sending the current message.</td>
</tr>
<tr>
<td>Duration Observation</td>
<td></td>
</tr>
<tr>
<td>Timing Constraint</td>
<td></td>
</tr>
<tr>
<td>Timing Observation</td>
<td></td>
</tr>
</tbody>
</table>

Enterprise Architect User Guide
See the OMG UML specification (UML Superstructure Specification, v2.1.1, p. 511).

In the diagram below, on the Open Order Message:

- **Duration Constraint** has been set to 0...13.

On the Get Cart Message:

- **Duration Constraint Between Messages** has been set to d...d^3
- **Duration Observation** has been set to d=duration
- **Timing Constraint** has been set to t...t+3
- **Timing Observation** has been set to t=now.

By typing a value in the **Duration Constraint** field, you enable the Message angle to be adjusted. After clicking on the **OK** button on the **Timing Details** dialog, click on the head of the Message connector and drag the connector up or down to change the angle. You cannot extend the angle beyond the life line of the connecting sequence object or create an angle of less than 5 degrees.

You can also create the **Duration Constraint Between Messages** line by dragging the **General Ordering** arrow up to the point at which the previous message joins the source Lifeline for the current message. A dialog displays on which you enter the value for the constraint. Having created the line, you can move it to any point within half way along the current message and half way along the previous message, to avoid overlap with other message timing details. You can edit or delete the value either through the **Timing Details** dialog or by right-clicking on the line itself and selecting the appropriate context menu option.
15.3.18.1.4 General Ordering

In a Sequence diagram, the workflow is represented by the sequence of Messages down the diagram. Messages near the top of the diagram are passed before Messages lower down the diagram.

Consider the following diagram.

Message 1 is earlier than Message 2. However, in a complex diagram, or when representing finely timed operations or parallel processing, this might not be apparent. You can reinforce the sequence using a General Ordering arrow.

Click on the Message arrow. A small arrow displays at the source anchor point.

Click on this arrow and drag it to the start of the next Message in sequence (Message 2 in the example). The General Ordering arrow displays, indicating that the second Message follows the first.

The General Ordering arrow is exaggerated in the above figure. You would normally have the arrow running almost horizontal across the diagram.
You can have more than one General Ordering arrow issuing from or targeting a Message, if necessary.

15.3.18.1.5 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.
15.3.18.2 Message (Communication Diagram)

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 connector between object instances. Always create the Association connector first, then add Messages to the connector.

Messages can 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 includes the exposed operations for the base type.

15.3.18.2.1 Create a Communication Message

To create a Communication Message, follow the steps below:

1. Open a diagram (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 communicate.
4. Right-click on an Association to display the context menu.
5. Select the option to add a message from one object to another.
6. When the Message Properties dialog displays, type in a name and any other required details.
7. Click on the **OK** button. The message is added, connected to the Association and Object instances.
8. Move the message to the required position.

### 15.3.18.2.2 Re-Order Messages

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.

#### Organize Message Groups

If you have several Messages in the form 1.1, 1.2, 1.3, 1.4, for example, but would like to start a new numbering group on, say, the third Message (i.e. 1.1, 1.2, **2.1**, 2.2, 2.3), you can change a Message in the series to a **Start Group** message.

To reorganize **message groups**, follow the steps below:

1. Double-click on a Message name. The **Message Properties** dialog displays.
2. To make the selected Message the start of a new group, select the Start New Group checkbox.
3. If required, in the Notes field, type an explanatory note. You can format the text using the Rich Text Notes toolbar at the top of the field.
4. Click on the OK button to save changes.

**Sequence Messages**

To re-order messages, follow the steps below:

1. Select the Diagram | Sequence Messages menu option. The Communication Messages dialog displays.
2. Click on the Message to move, and click on the 'hand' buttons at the bottom of the dialog to move the messages up or down the sequence. Repeat until the sequence matches your requirements.

3. Click on the OK button to save changes.

**Note:**

Communication diagrams were known as Collaboration diagrams in UML 1.4.

### 15.3.18.3 Message (Timing Diagram)

Messages are the communication links between Lifelines in a Timing diagram. In the case of a Timeline, a Message is a connection between two Timeline objects.
Toolbox Icon

15.3.18.3.1 Create a Timing Message

To create a Message in a Timing diagram, at least two Lifeline objects (State or Value) must be created first, each with existing transition points. To create a Message between lifelines, follow the steps below:

1. Click on one of the Lifelines in the Timing diagram.
3. Drag the cursor onto the Lifeline at the point at which the Message originates. The Timing Message dialog displays. (If not, double-click on the Message.)

The dialog consists of a set of transition points. Each transition point can be defined with the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Defines the lifeline where the message originates.</td>
</tr>
<tr>
<td>End</td>
<td>Defines the lifeline where the message terminates.</td>
</tr>
</tbody>
</table>

These are set by default when a Message is created by dragging the cursor between two Lifelines.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Time</td>
<td>Specifies the start time for a message.</td>
</tr>
<tr>
<td>End Time</td>
<td>Specifies the end time for a message.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the message.</td>
</tr>
<tr>
<td>Time Observation</td>
<td>Provides information on the time of a sent message.</td>
</tr>
<tr>
<td>Duration Observation</td>
<td>Indicates the interval of a Lifeline at a particular state, begun from a</td>
</tr>
<tr>
<td></td>
<td>message receipt.</td>
</tr>
<tr>
<td>Transition To</td>
<td>The state in the target Lifeline that the Message points to.</td>
</tr>
<tr>
<td>Event</td>
<td>The occurring event.</td>
</tr>
<tr>
<td>Time Constraint</td>
<td>The time taken to transmit a message.</td>
</tr>
<tr>
<td>Duration Constraint</td>
<td>Pertains to a lifeline's period at a particular state. The constraint could</td>
</tr>
<tr>
<td></td>
<td>be instigated by that Lifeline's receipt of a message.</td>
</tr>
</tbody>
</table>

The following diagram shows the Message configured by the above dialog snapshot.

**Note:**
You can move the source end of the Message freely along the source timeline. However, the target end (arrow head) must attach to a transition. If you create a new Message and do not give it a target transition, it automatically finds and attaches to the nearest transition. If you move the target end, it drags the transition with it.
15.3.19 Nesting

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 in a Package diagram.

Toolbox Icon

Nesting
A Notelink connector connects a Note to one or more other elements of any other type.

Both Note and Notelink are available in any category of the Enterprise Architect UML Toolbox, in the Common pages. You can also select them from the UML Elements toolbar.

**Toolbox Icon**

![Notelink Icon]
15.3.21 **Object Flow**

Object Flows are used in *Activity diagrams* and *State Machine diagrams*. When used in an Activity diagram, an Object Flow connects two elements, with specific data passing through it. To view sample Activity diagrams using Object Flows, see the *Object Flows in Activity Diagrams* topic.

In State Machine diagrams, 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 pages of the Enterprise Architect UML Toolbox, or from the drop-down list of all relationships located in the header toolbar. You can also modify a transition connection to an Object Flow by selecting the *ObjectFlow* checkbox on the connection Properties dialog.

See the *Control Flow* topic for information on setting up Guards and Weights on Object Flows.

### Toolbox Icon

![Object Flow](image)

### OMG UML Specification

The OMG UML specification (*UML Superstructure Specification, v2.1.1, p. 389*) states:

> An object flow is an activity edge that only passes object and data tokens.

#### 15.3.21.1 Object Flows in Activity Diagrams

In *Activity diagrams*, there are several ways to define the flow of data between objects.

The following diagram depicts a simple *Object Flow* between two actions, *Fill Order* and *Ship Order*, both accessing order information.

![Object Flow Diagram](image)

See *UML Superstructure Specification, v2.1.1, figure 12.110, p. 391.*

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 used to reflect the order.

![Refined Diagram](image)

See *UML Superstructure Specification, v2.1.1, figure 12.110, p. 391.*

The following diagram is an example of multiple Object Flows exchanging data between two actions.
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 the **Attach Note or Constraint** context menu option. A dialog lists 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 the behavior with the notation «selection» or «transformation».


15.3.22 Occurrence

An Occurrence relationship indicates that a Collaboration represents a classifier, in a Composite Structure diagram. An Occurrence connector is drawn from the Collaboration to the classifier.

Toolbox Icon

Occurrence
15.3.23 Package Import

A Package Import relationship is drawn from a source Package to a Package whose contents are to be imported. Private members of a target Package cannot be imported. The relationship is typically used in a Package diagram.

Toolbar Icon

Package Import

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 112) 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.
15.3.24 Package Merge

In a Package diagram, 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 where multiple packages contain identically-named elements, representing the same thing. A Package Merge merges 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 XMI exports or Enterprise Architect Automation Interface scripts for code generation or other Model Driven Architecture (MDA) interests.

Package Merge relationships are useful to reflect situations where existing architectures contain functionalities involving like elements, which are merged in a developing architecture. Merging doesn't affect the merged objects, and supports the common situation of product progression.

**Toolbar Icon**

![Package Merge Icon]

**OMG UML Specification**

The OMG UML specification ([UML Superstructure Specification, v2.1.1, p. 113-114](https://www.omg.org/spec/UML/2.1.1/)) states:

A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined. It is very similar to Generalization in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both.

This mechanism should be used when elements defined in different packages have the same name and are intended to represent the same concept. Most often it is used to provide different definitions of a given concept for different purposes, starting from a common base definition. A given base concept is extended in increments, with each increment defined in a separate merged package. By selecting which increments to merge, it is possible to obtain a custom definition of a concept for a specific end. Package merge is particularly useful in meta-modeling and is extensively used in the definition of the UML metamodel.

Conceptually, a package merge can be viewed as an operation that takes the contents of two packages and produces a new package that combines the contents of the packages involved in the merge. In terms of model semantics, there is no difference between a model with explicit package merges, and a model in which all the merges have been performed.
15.3.25 Realize

A source object implements or Realizes its destination object. Realize is used in a Use Case, Component or Requirements diagram to express traceability and completeness in the model. A business process or Requirement is realized by one or more Use Cases, which in turn are realized by some Classes, which in turn are realized by a Component, and so on. Mapping Requirements, Classes and such 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.

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 131) 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.
15.3.26 Recursion

A Recursion is a type of Message used in Sequence diagrams to indicate a recursive function.

Toolbox Icon

Recursion
15.3.27 Role Binding

Role Binding is the mapping between a Collaboration Occurrence's internal roles and the respective Parts required to implement a specific situation, typically in a Composite Structure diagram. 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.

**Toolbar Icon**

![Role Binding Icon]

**OMG UML Specification**

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 174) 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).
15.3.28 Represents

The Represents connector indicates that a Collaboration is used in a classifier, typically in a Composite Structure diagram. The connector is drawn from the Collaboration to its owning classifier.

Toolbar Icon

![Represents Icon]
15.3.29 Representation

The Representation relationship is a specialization of a Dependency, connecting Information Item elements that represent the same idea across models, typically in an Analysis diagram. For example, Bonus and Salary are both a representation of the Information Item Wage.

Toolbar Icon

![Representation Icon]
15.3.30 Trace

The Trace relationship is a specialization of a Dependency, connecting model elements or sets of elements that represent the same concept across models. Traces are often used to track requirements and model changes, typically in a Traceability diagram, or in a Class, Use Case, Object or Composite Structure diagram.

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.

Toolbar Icon

- Trace
15.3.31 Transition

A Transition defines the logical movement from one State to another, in a State Machine diagram. The Transition can be controlled through the following connector Properties dialog:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guard</td>
<td>Type in 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.</td>
</tr>
<tr>
<td>Effect is a Behavior</td>
<td>Convert the Effect field from a free-text field to the definition of a specific Activity or behavior. Enterprise Architect displays the Set Element Classifier dialog to prompt you to select the Activity or behavior element from the model.</td>
</tr>
<tr>
<td>Effect</td>
<td>Either:</td>
</tr>
<tr>
<td></td>
<td>• Type a description of the effect of the Transition, or</td>
</tr>
<tr>
<td></td>
<td>• If you have selected the Effect is a Behavior check box, select an Activity or behavior to be performed during the Transition (to change this subsequently, click on the [...] button to redisplay the Set Element Classifier dialog).</td>
</tr>
</tbody>
</table>
### OMN UML Specification

The OMG UML specification *(UML Superstructure Specification, v2.1.1, p. 568)* states:

*A transition is a directed relationship between a source vertex and a target vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to an occurrence of an event of a particular type.*

---

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Specify the name of the trigger.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of trigger: <em>Call, Change, Signal</em> or <em>Time</em>.</td>
</tr>
<tr>
<td>Specification</td>
<td>Specify the event instigating the Transition.</td>
</tr>
<tr>
<td>New</td>
<td>Create a new trigger.</td>
</tr>
<tr>
<td>Save</td>
<td>Save the current trigger.</td>
</tr>
<tr>
<td>Remove</td>
<td>Remove the selected trigger from the list.</td>
</tr>
<tr>
<td>Triggers</td>
<td>List the current triggers for the Transition.</td>
</tr>
</tbody>
</table>
15.3.32 Use

A Use relationship indicates that one element requires another to perform some interaction. The Use (or Usage) relationship does not specify how the target supplier is used, other than that the source client uses it in definition or implementation. A Use relationship is a sub-typed Dependency relationship. You typically use the Use relationship in Use Case diagrams to model how Actors use system functionality (Use Cases), or to illustrate usage dependencies between Classes or Components.

Notes:
- It is more usual (and correct UML) to have an Association Connector between an Actor and a Use Case.
- To depict a usage dependency on a Class or Component diagram, draw a Dependency connector. Right-click on the Dependency, and select the Dependency Stereotypes | Use menu option.

Toolbar Icon

OMG UML Specification

The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 138) 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.
16 SDK for Enterprise Architect

Introduction

Welcome to the Enterprise Architect Software Developers Kit (SDK). This is a special section of the Enterprise Architect User Guide, covering the more advanced aspects of extending and customizing Enterprise Architect.

In describing aspects of extending Enterprise Architect, it is expected that you are familiar with the concepts introduced in the main body of the Enterprise Architect User Guide. Wherever appropriate, linked cross-references to these concepts are provided in the text.

Contents

- UML Profiles (incorporating UML Stereotypes)
- MDG Technologies
- Shape Scripts
- Tagged Value Types
- Code Template Framework
- Enterprise Architect Object Model (Automation Interface)
- Enterprise Architect Add-In Model
16.1 Developing Profiles

Introduction

UML Profiles provide a means of extending the UML Language, which enables you to build UML models in particular domains. They are based on additional stereotypes and Tagged Values that are applied to UML elements, connectors and their components. A Profile is a collection of such extensions that together describe some particular modeling problem and facilitate modeling constructs in that domain. UML Profiles for Enterprise Architect are specified in XML files, with a specific format. These XML files are imported into Enterprise Architect through the Resources window.

The imported Profile also automatically generates a page of elements and relationships in the Enterprise Architect UML Toolbox.

The Resources window 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 Enterprise Architect from supplied XML files.

Items in the Profile represent stereotypes. UML supports a large number of stereotypes, which are an inbuilt mechanism for logically extending or altering the meaning, display, appearance and syntax of a model element. Different model elements have different stereotypes associated with them.

For more information on the use of Profiles in Enterprise Architect, see the UML Profiles topic.

For information on developing your own Profiles, see the following topics:

- Custom Stereotypes
- Create Profiles
- Quick Linker
- Toolbox Profiles
- Diagram Profiles
- Task Pane Profiles
16.1.1 Custom 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. For more information on the use of stereotypes in Enterprise Architect, see the UML Stereotypes topic.

In Enterprise Architect you can create new stereotypes with their own custom appearance. The stereotypes can be altered to make use of metafiles (image files) and customized colors, or you can make use of the Enterprise Architect 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. The UML Types dialog displays, defaulted to the Stereotypes tab.

2. Type or select a Stereotype name.
3. Select a Base Class from the drop-down list.
4. To associate a Metafile with this stereotype, click on the Metafile radio button and the Assign button, and locate the required .emf or .wmf file.
5. Enter optional Notes and select Default Colors for this stereotype.
6. Click on the Save button to save the stereotype.

The table below describes the functionality of the Stereotypes tab.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stereotype</td>
<td>Specify the name of the stereotype.</td>
</tr>
<tr>
<td>Option</td>
<td>Use to</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Group name</td>
<td>Enable grouping of stereotype features by a plural name, for attributes and operations, and is shown on diagrams in the attribute and operations compartments.</td>
</tr>
<tr>
<td>Base Class</td>
<td>Enable the stereotyped element to inherit the base characteristics from a pre-existing element type.</td>
</tr>
<tr>
<td>Notes</td>
<td>Add any stereotype notes.</td>
</tr>
<tr>
<td>Override Appearance</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Switch to the default element appearance.</td>
</tr>
<tr>
<td>Metafile</td>
<td>Enable an image file to be used for the appearance of the stereotype.</td>
</tr>
<tr>
<td>Shape Script</td>
<td>Specify custom shapes for the stereotype using the Enterprise Architect Shape Scripting language. For more information see the Shape Scripting topic.</td>
</tr>
<tr>
<td>Assign</td>
<td>Add the associated metafile or shape script from the stereotyped element.</td>
</tr>
<tr>
<td>Remove</td>
<td>Remove the associated metafile or shape script from the stereotyped element.</td>
</tr>
<tr>
<td>Default Colors</td>
<td></td>
</tr>
<tr>
<td>Fill</td>
<td>Set the default background color of the element.</td>
</tr>
<tr>
<td>Border</td>
<td>Control the border color.</td>
</tr>
<tr>
<td>Font</td>
<td>Control the color of the stereotype font.</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset the appearance of the element to the default element appearance.</td>
</tr>
</tbody>
</table>

**Note:**

You can transport these custom stereotype definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu.
16.1.2 Create Profiles

This topic describes how to create profiles and profile items. These creation tasks include creating the profile stereotypes, defining the metaclasses they apply to, and defining Tagged Values and constraints. This topic also describes how to export a profile for use in UML modeling.

To create a UML Profile, follow the steps below:

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. Add Shape Scripts
7. Set Default Appearance
8. Export the Profile

16.1.2.1 Create a Profile Package

In Enterprise Architect, you must create a UML Profile in a Package that has the stereotype «profile». To create a Profile Package, follow the steps below.

1. Open or create the appropriate Class diagram.
2. Open the Profile page of the Enterprise Architect UML Toolbox (More tools | UML | Profile).
3. Drag the Profile item onto the Class diagram. The New Package dialog displays.
4. In the Package Name field, type a name for the Profile.
5. Click on the OK button. Enterprise Architect creates a package with the stereotype «profile» and with a child diagram.
6. Double-click on the Profile Package on the diagram to open the child diagram.

You now use this child diagram to add stereotypes to the Profile.

16.1.2.2 Add Stereotypes and Metaclasses

To add metaclasses and stereotypes to a Profile, follow the steps below for as many stereotypes and metaclasses as you require:

1. Open the child diagram of the Profile Package.
2. Drag the Metaclass element from the Profile page of the Enterprise Architect UML Toolbox onto the diagram. The Create New Metaclass dialog displays, in which you can tick multiple metaclasses for
dropping onto the diagram.

3. Scroll down the Element list and select the checkbox for **Class**.
4. Click on the **OK** button, and in the Class Name dialog type a name for the element. Click on the **OK** button again.
5. Drag a **Stereotype** element from the Toolbox onto the diagram. If the Properties dialog does not display, double-click on the element on the diagram.
6. In the **Name** field, type a name for the stereotype.
7. Click on the **OK** button and, if it displays, **Close** the Generate Code dialog.
8. Click on the **Extension** relationship in the Toolbox and drag the connection from the stereotype element to the metaclass element.
9. Your diagram should now resemble the one below:

You can now add **stereotype Tags**, **Constraints**, **Enumerations**, and/or **Shape Scripts** to your Profile, and define the default appearance of the elements or connectors as required.
16.1.2.3 Define Stereotype Tags

Stereotypes within a UML Profile can have one or more associated Tagged Values. When creating a UML Profile, you define these Tagged Values as attributes of the stereotyped Class.

You can also:

- Define Stereotype Tags with Predefined Tag Types
- Define Stereotype Tags with Supported Attributes
- Use the Tagged Value Connector

To define Tagged Values for a stereotype, follow the steps below:

1. Open the Attributes dialog for the stereotyped element.

2. Click on the New button to create a new attribute.
3. In the Name field, type the name of the stereotype tag.
4. In the Type field, click on the drop-down arrow and select the attribute type.
5. In the Initial field, type the initial value of the tag. (See Add Enumerations to UML Profiles for the steps for creating enumerated types for Tagged Values.)
6. In the Notes field, type a description of the tag.
7. Click on the Save button.

16.1.2.3.1 With Predefined Tag Types

Define Predefined Tag Types

To define a stereotype tag with a predefined tag type, you must first create the predefined tag type. For full instructions on how to do this, see the Create Structured Tags topic.
Assign Predefined Tag Types to Stereotypes

To assign a predefined tag type to a stereotype, just create an attribute with the same name. For example, to make the Tagged Value Handicap 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.

16.1.2.3.2 With Supported Attributes

Supported stereotype attribute tags are special tags that 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. For a list of supported attributes, see the Supported Attributes topic.

To define tags for a stereotype with supported attributes, follow the steps below:

1. Open the Attributes dialog for the stereotyped element.
2. In the **Name** field, type the name of the stereotype tag.

3. In the **Initial** field, type the initial value of the tag.

**Note:**
For supported attributes you set only the **Name** (which must match the attributes listed in the supported attributes section) and the **Initial** value; do not set the other values.

4. Click on the **Save** button.

### 16.1.2.3.3 Use the Tagged Value Connector

In a Profile, you can use the **Tagged Value** connector to define a Tagged Value that has as its value the name of an element containing the stereotype pointed to. You select the Tagged Value connector from the Profile pages of the Enterprise Architect UML Toolbox.

The following diagram demonstrates how you might use the connector. It shows a (saved and imported) profile that defines two stereotypes: «Collection» and «Node». The «Collection» stereotype has a Tagged Value connector named **rootNode**, pointing to the «Node» stereotype.
In the Tagged Values window for the connector, against rootNode, you click on the selection button. This displays the Set Element Classifier dialog, which lists the elements in the current model with the «Node» stereotype. You can then select one of these elements as the value of the tag.

### 16.1.2.4 Define Stereotype Constraints

Defining constraints for stereotypes uses the same procedure as defining constraints for any Class. To define constraints for a stereotype, follow the steps below:

1. Open the Class Properties dialog of the stereotype element in a diagram.
2. Click on the Constraints tab and click on the New button to create a new constraint.
3. In the **Constraint** field, type the value of the constraint.
4. In the **Type** field, click on the drop-down arrow and select the appropriate type.
5. In the **Status** field, click on the drop-down arrow and select the appropriate status.
6. In the **Notes** field, type any additional information required.
7. Click on the **Save** button.
16.1.2.5 Add Enumeration Elements

Enumerations can 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.

To add an Enumeration element, follow the steps below:

1. Open your Profile Package child Class diagram.
2. In the Enterprise Architect UML Toolbox, select More tools | UML | Profile. The contents of the Profile page of the Toolbox display.
3. Drag an Enumeration item from the toolbox onto the diagram. If the Properties dialog does not display, double-click on the element on the diagram.
4. In the Name field, type the name of the new Enumeration.
5. Click on the Details tab and click on the Attributes button. The Attributes Properties dialog displays.
6. In the **Name** field, type the name of the Enumeration attribute.
7. In the **Type** field, click on the drop-down arrow and select the appropriate type.
8. In the **Initial** field, type the initial value of the attribute.
9. Click on the **Save** button, and repeat steps 6 to 9 for additional attributes.
10. When you are finished, click on the **Close** button.
11. Right-click on the **Stereotype** element and select the **Attributes** context menu option. The **Attributes Properties** dialog displays for the stereotype.
12. In the **Name** field type a name for the attribute.
13. In the **Type** field type the name of the Enumeration element.
14. In the **Initial** field type the name of the first enumeration attribute you defined.
15. Click on the **Save** and **Close** buttons.

You have now generated a drop-down list for setting the value of the tag in the **Tagged Values** window.

### 16.1.2.6 Add Shape Scripts

To add a **Shape Script** to a stereotype in a UML Profile, follow the steps below:

1. On the Profile Package child diagram, select a **Stereotype** element.
2. Right-click on the element and select the **Attributes** context menu option.
3. In the **Attributes Properties** dialog, in the **Name** field, type `_image`. 
4. Click on the [...] button next to the Initial field. The Shape Editor dialog displays.
5. Enter the shape script in the Shape Editor dialog, and click on the OK and Close buttons.
   The Stereotype element now resembles the example below:
16.1.2.7 Set Default Appearance

You can define the appearance of stereotyped elements and connectors as you create or edit the stereotypes using the Override Appearance and Default Colors panels of the UML Types dialog. However, an easier way is to review your completed profile diagram and set the default appearance of the elements and connectors in place.

Simply click on the required element or connector and press [F4], then define the background, font and border colors and border thickness as appropriate, on the Default Appearance dialog.

When you save the profile containing the stereotyped elements and connectors, make sure that you select the Color and Appearance checkbox on the Save UML Profile dialog.

16.1.2.8 Export a UML Profile

Once you have created a Profile and defined the elements and metaclasses, you can save (export) the Profile to disk for future UML models.

To save a Profile, follow the steps below:

1. If your profile is
   - a single profile spread over multiple diagrams within the same Profile package, find the Profile package in the Project Browser window, right-click it and select the Save Package as UML Profile menu option
   - one of multiple profiles within the same Profile package, right-click anywhere in the background of the Profile diagram and select the Save as Profile menu option
   - a single diagram within the Profile package, choose either the Save Package as UML Profile menu option or the Save as Profile menu option.

   **Note:**
   The two menu options give slightly different results. See Save Profile Options.

2. The Save UML Profile dialog displays.

3. Click on the [ ... ] (Browse) button, and select the destination for the XML Profile file to be exported. If necessary, edit the profile filename, but do not delete the .xml extension.

4. Against the Profile Type, select the UML 2.0 radio button.

   **Note:**
   You can also create stereotypes using the same format as for pre-4.0 versions of Enterprise Architect, by clicking on the EA Pre-4.0 radio button. This has a reduced feature set and is provided for legacy use.

5. Set the required export options for all stereotypes defined in the profile:
• **Element Size** - select the checkbox to export the element size attributes
• **Color and Appearance** - select the checkbox to export the color (background, border and font) and appearance (border thickness) attributes
• **Alternate Image** - select the checkbox to export the metafile images
• **Code Templates** - select the checkbox to export the code templates, if they exist.

6. Click on the **Save** button to save the profile to disk.

### 16.1.2.8.1 Save Profile Options

When you save a UML Profile, you can save it either from the package or from the diagram, depending on whether the Profile is:

- a single profile spread over multiple diagrams within the same Profile package (find the Profile package in the Project Browser, right-click it and select the **Save Package as UML Profile** menu option), which is typically the case for a stereotypes profile
- one of multiple profiles within the same Profile package (right-click anywhere in the background of the Profile diagram and select the **Save as Profile** menu option); for example, when creating multiple toolbox profiles
- a single diagram within the Profile Package (choose *either* the **Save Package as UML Profile** menu option or the **Save as Profile** menu option).

The two context menu options produce slightly different results. You should take these into consideration, especially in the third instance where you could choose either option.

<table>
<thead>
<tr>
<th>Save From Diagram</th>
<th>Save From Package</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>The profile takes the diagram name.</td>
<td>The profile takes the package name.</td>
<td>Package and diagram names are not necessarily the same, although you can save a lot of confusion if you make them the same or very similar. For example: package GL with diagrams GL1, GL2, GL3.</td>
</tr>
<tr>
<td>The profile takes the diagram's notes.</td>
<td>The profile takes the package's notes.</td>
<td></td>
</tr>
<tr>
<td>You can take the default size and appearance (including alternate image) from the diagram object.</td>
<td>You cannot take the default size and appearance from the diagram object.</td>
<td>You can use the _sizeX, _sizeY and _image properties, but there is no equivalent for default colors.</td>
</tr>
<tr>
<td>Can be much faster.</td>
<td>Can be much slower.</td>
<td>Because diagram objects are kept in memory and project browser elements aren't. This is only likely to be an issue if the profile is a large one and you are using a slow network connection to a remote repository.</td>
</tr>
</tbody>
</table>

### 16.1.2.9 Supported Attributes

#### Supported Stereotype Attributes in UML Profiles

The following attributes can be applied to stereotypes in UML Profiles:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sizeX</td>
<td>Initial width of the element, in pixels at 100% zoom.</td>
</tr>
<tr>
<td>_sizeY</td>
<td>Initial height of the element, in pixels at 100% zoom.</td>
</tr>
<tr>
<td>_image</td>
<td>Shape script definition.</td>
</tr>
<tr>
<td>_metatype</td>
<td>Used for defining stereotypes as metatypes.</td>
</tr>
</tbody>
</table>
Supported Metatype Attributes in UML Profiles

The following attributes can be applied to metatype Classes in UML Profiles, and refer to the stereotypes that extend them:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_defaultDiagramType</td>
<td>Used for defining child diagram types</td>
</tr>
<tr>
<td>_makeComposite</td>
<td>Used for creating composite elements</td>
</tr>
<tr>
<td>_AttPri</td>
<td>If set to 1, switches on the Attribute Visibility: Private setting.</td>
</tr>
<tr>
<td>_AttPro</td>
<td>If set to 1, switches on the Attribute Visibility: Protected setting.</td>
</tr>
<tr>
<td>_AttPub</td>
<td>If set to 1, switches on the Attribute Visibility: Public setting.</td>
</tr>
<tr>
<td>_AttPkg</td>
<td>If set to 1, switches on the Attribute Visibility: Package setting.</td>
</tr>
<tr>
<td>_OpPri</td>
<td>If set to 1, switches on the Operation Visibility: Private setting.</td>
</tr>
<tr>
<td>_OpPro</td>
<td>If set to 1, switches on the Operation Visibility: Protected setting.</td>
</tr>
<tr>
<td>_OpPub</td>
<td>If set to 1, switches on the Operation Visibility: Public setting.</td>
</tr>
<tr>
<td>_OpPkg</td>
<td>If set to 1, switches on the Operation Visibility: Package setting.</td>
</tr>
<tr>
<td>_AttInh</td>
<td>If set to 1, switches on the Inherited Features: Show Attributes setting.</td>
</tr>
<tr>
<td>_OpInh</td>
<td>If set to 1, switches on the Inherited Features: Show Operations setting.</td>
</tr>
<tr>
<td>_Constraint</td>
<td>If set to 1, switches on the Show Element Compartments: Constraints setting.</td>
</tr>
<tr>
<td>_Responsibility</td>
<td>If set to 1, switches on the Show Element Compartments: Responsibilities setting.</td>
</tr>
<tr>
<td>_ResInh</td>
<td>If set to 1, switches on the Show Element Compartments: Inherited Responsibilities setting.</td>
</tr>
<tr>
<td>_Tag</td>
<td>If set to 1, switches on the Show Element Compartments: Tags setting.</td>
</tr>
<tr>
<td>_TagInh</td>
<td>If set to 1, switches on the Show Element Compartments: Inherited Tags setting.</td>
</tr>
<tr>
<td>_PType</td>
<td>If set to 1, switches on the Show element type (Port and Part only) setting.</td>
</tr>
<tr>
<td>_Runstate</td>
<td>If set to 1, switches on the Hide Object Runstate in current diagram setting.</td>
</tr>
<tr>
<td>_HideStype</td>
<td>If set to a comma-separated list of stereotypes, sets the Hide Stereotyped Features filter.</td>
</tr>
</tbody>
</table>

16.1.2.9.1 Define a Stereotype as a Metatype

The _metatype attribute is applied to a stereotype element. This is used where users want to hide the identity of an element as a stereotyped UML element. It is also a method of getting custom types to appear in contexts where only Enterprise Architect's inbuilt types would normally appear; for example in the lists of element types in the Relationship Matrix.

In the following example from SysML, block is defined as a stereotype that extends a UML Class.
However, a SysML user isn’t interested in UML Classes, only in SysML Blocks. An element created from a stereotype defined this way, while behaving like a stereotyped Class in most contexts:

- Shows *Block Properties* rather than *Class Properties* as the title of its *Properties* dialog
- Is auto-numbered as *Block1* not *Class1* on creation, and
- Appears as *Block* not *Class* in many other contexts throughout Enterprise Architect.

### 16.1.2.9.2 Define Multiple-Stereotype Level

The *_strictness* attribute is applied to a stereotype element. It defines to what level multiple stereotypes can be applied to an element. The type of the attribute is *StereotypeStrictnessKind* and it can have one of four values:

- *profile*, which states that an element of this type cannot be given more than one different stereotype from the same profile
- *technology*, which states that an element of this type cannot be given more than one different stereotype from the same technology
- *all*, which states that an element of this type cannot have multiple stereotypes at all, or
- *none*, which is the default Enterprise Architect behaviour and states that there are no restrictions on the use of multiple stereotypes.

The following example is from SysML and shows that a «flowPort» cannot have any other stereotype applied to it.
16.1.2.9.3 Define Creation of Instance

The _instanceType attribute is applied to a stereotype element and defines what kind of element is created as an instance of this element type. The value corresponds to the metatype given to a stereotype using the _metatype attribute. It is shown on the Paste Element dialog and is translated if it matches an Enterprise Architect element type.

The _instanceMode attribute is applied to a stereotype element and controls the text in the Paste Element dialog after being translated. Valid values are instance and property, with the default being instance.

The _instanceOwner attribute is applied to a stereotype element and controls the text in the Paste Element dialog. It is translated if it matches an Enterprise Architect element type. The default value is Element.

The following example from SysML shows that when an instance of a Block is created, it is created as a BlockProperty element.

16.1.2.9.4 Create Composite Elements

The _makeComposite attribute is applied to a metaclass element, not a stereotype element. It defines whether an element is always made composite when created.

Notes:

- A stereotyped package is not by default created with a child diagram, so you should use the _makeComposite attribute to ensure the child diagram is created.

- Unless you also use the _defaultDiagramType attribute to define the child diagram type, the child diagram created is a Package diagram.

The following example from BPMN shows that a BusinessProcess element is always created as a Composite element with a BPMN custom child diagram.
16.1.2.9.5 Define Child Diagram Types

The _defaultDiagramType attribute is applied to a metaclass element, not a stereotype element. It defines the type of diagram created when an element is made composite. This attribute can take as its name any of the inbuilt diagram types of Enterprise Architect (these are listed below). Alternatively, if a custom diagram type is required, it should be prefixed with the technology name and '::'.

The following examples show a «BusinessProcess» Activity that, when made a composite element, automatically creates an Analysis diagram, and a «block» stereotype that creates a SysML InternalBlock custom diagram.

You can also use the _defaultDiagramType attribute for packages, extending the Package metaclass.

Values For _defaultDiagramType

The initial value of the _defaultDiagramType attribute must be set to one of the following:

- UML Behavioral::Use Case
- UML Behavioral::Activity
- UML Behavioral::State Machine
- UML Behavioral::Communication
- UML Behavioral::Sequence
- UML Behavioral::Timing
- UML Behavioral::Interaction Overview
16.1.2.10 Stereotypes Profiles

In Enterprise Architect 6, an MDG Technology could consist of many UML Profiles, each representing an Enterprise Architect UML Toolbox page. Each profile could include stereotype definitions alongside redefinitions of standard UML elements. This meant that to define a stereotype in a technology, it had to appear in one of the Toolbox pages.

Enterprise Architect 7 takes a different approach, splitting the task of defining the stereotypes and the task of defining the Toolbox pages into separate profiles. The MDG Technology's stereotypes are contained in one or more profiles. The stereotypes within each profile use the profile name as the namespace.

Create one or more packages with the «profile» stereotype, each package name being the namespace. Within each package create profile diagrams defining all the stereotypes in the namespace. You can use multiple diagrams to do this, but do not use nested packages.

Give the «profile» package a description in the Notes field (eg MDG Technology for BPMN). When all of the stereotypes are defined (make sure that every stereotype extends at least one Metaclass) right-click the profile package in the Project Browser and select the Save Package as UML Profile context menu option, then proceed as usual.
16.1.3 Quick Linker

Introduction
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 an element. Simply clicking and dragging the icon enables you to create new connectors and elements. The philosophy behind the built-in Quick Linker definitions has always been to provide not a complete list of valid or legal connections, but a short and convenient list of the commonest connections for the given context. As part of a UML Profile, you can add to or replace the built-in Quick Linker definitions; the following sections of this document explain how:

- Quick Linker Definition Format
- Quick Linker Example
- Hide Default Quick Linker Settings
- Quick Linker Object Names

Customized Quick Linker Settings
A Quick Linker definition is a Comma Separated Value (CSV) format file. It is best manipulated in a spreadsheet which should be set up to save the CSV file as comma-separated text without quotation marks around text fields.

To add a Quick Linker definition file to a profile or technology, simply place a DocumentArtifact element onto the Profile diagram. Give it the name 'QuickLink' then double-click on it. Open your CSV file in a text editor such as Notepad and copy and paste the contents into the DocumentArtifact element. The definitions are saved with the profile and are processed and applied when the profile is imported. The same applies if a profile is included within a technology, with the proviso that the QuickLink element must be in the same profile as the link stereotype definitions. This means that a technology could have a set of Quick Link definitions for each profile.

16.1.3.1 Quick Linker Definition Format
A Quick Linker definition is a text file consisting of records terminated by new-line characters. Each record must consist of 23 comma-separated fields, as defined by the table below. The values of each field must not be in quotes (" "). A Quick Linker definition can include comments: all lines that begin with // are ignored by Enterprise Architect.

Each record of the Quick Linker definition represents a single entry on the Quick Linker menu. Some fields define the menu command; some fields can be thought of as filters, with the entry being ignored if the filter condition isn't met.

A Quick Linker definition has the following fields.

<table>
<thead>
<tr>
<th>Column</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Source Element Type</td>
<td>The row is ignored unless a connector is being dragged away from this type of element.</td>
</tr>
<tr>
<td>B</td>
<td>Source Stereotype Filter</td>
<td>If set, the row is ignored unless a connector is being dragged away from an element with this stereotype.</td>
</tr>
<tr>
<td>C</td>
<td>Target Element Type</td>
<td>If set, the row is ignored unless a connector is being dragged onto this type of element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If blank, the row is ignored unless a connector is being dragged onto an empty piece of diagram.</td>
</tr>
<tr>
<td>D</td>
<td>Target Stereotype Filter</td>
<td>If set and Target Element Type also set, the row is ignored unless a connector is being dragged onto an element with this stereotype.</td>
</tr>
<tr>
<td>E</td>
<td>Diagram Filter</td>
<td>Contains either an inclusive or exclusive list of diagrams, which limits the diagrams the given kind of connector can be included on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each diagram name is terminated by a semi-colon. Excluded diagram names are preceded by an exclamation mark.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example of an inclusive list: Collaboration;Object;Custom;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example of an Exclusive list: !Sequence;</td>
</tr>
</tbody>
</table>

© 1998-2008 Sparx Systems Pty Ltd
<table>
<thead>
<tr>
<th>Column</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>New Element Type</td>
<td>If set and Create Element also set, results in the creation of an element of this type.</td>
</tr>
<tr>
<td>G</td>
<td>New Element Stereotype</td>
<td>If set and Create Element also set, results in the creation of an element with this stereotype.</td>
</tr>
<tr>
<td>H</td>
<td>New Link Type</td>
<td>If set and Create Link also set, results in the creation of a connector of this type.</td>
</tr>
<tr>
<td>I</td>
<td>New Link Stereotype</td>
<td>If set and Create Link also set, results in the creation of a connector with this stereotype.</td>
</tr>
</tbody>
</table>
| J      | New Link Direction            | Can be:  
- directed (always creates an association from source to target)  
- from (always creates an association from target to source)  
- undirected (always creates an association with unspecified direction)  
- bidirectional (always creates a bi-directional association), or  
- to (creates either a directed or undirected association, depending on the value of the Association Direction option). |

**Note:** Not all of the above work with all connector types; for example, you cannot create a bi-directional Generalization.

| K      | New Link Caption              | If a new connector is being created but not a new element, then this is the text that appears on the context menu.                                                                                              |
| L      | New Link & Element Caption    | If a new connector AND a new element are being created, then this is the text that appears on the context menu.                                                                                              |
| M      | Create Link                   | If set to TRUE, results in creation of a new connector; otherwise should be left blank.                                                                                                                    |
| N      | Create Element                | If set to TRUE the row is ignored unless a connector is being dragged onto an empty piece of diagram and results in creation of a new element; otherwise should be left blank.  
This overrides the values of Target Element Type and Target Stereotype Filter. |
| O      | Disallow Self connector       | Should be set to TRUE if self connectors are invalid for this kind of connector; otherwise should be left blank.                                                                                          |
| P      | Exclusive to ST Filter + No inherit from Metatype | If set to TRUE, indicates that elements of type Source Element Type with the stereotype Source Stereotype Filter do not display the Quick Linker definitions of the equivalent unstereotyped element. |
| Q      | Menu Group                    | If set, indicates the name of a sub-menu in which a menu item is created.                                                                                                                                   |
| R      | Complexity Level              | Not implemented, always set to 0.                                                                                                                                                                           |
| S      | Target Must Be Parent         | If set to TRUE this menu item only appears when dragging from a child element to its parent; for example from a port to its containing Class.                                                            |
| T      | Embed element                 | If set to TRUE the element being created is embedded in the target element; otherwise should be left blank.                                                                                              |
| U      | Precedes Separator LEAF       | If set to TRUE results in a menu separator being added to the Quick Linker menu; otherwise should be left blank.                                                                                          |
| V      | Precedes Separator GROUP      | If set to TRUE results in a menu separator being added to the Quick Linker sub-menu; otherwise should be left blank.                                                                                     |
| W      | Dummy Column                  | Depending on which spreadsheet application you use, this column might require a value in every cell to force CSV export to work correctly with trailing blank values. |
16.1.3.2 Quick Linker Example

This example uses a Class element with the stereotype «quick». The example scenario is this: when you drag a connector away from one of these elements, you want to create a Dependency either to or from a component element. When you drag a connector onto an existing Port or component element, you want a Dependency either to or from the component or, in the case of a component, you want to be able to create an embedded Port element.

This results in 8 records in the Quick Linker definition file.

1. Dependency to new Component
2. Dependency from new Component
3. Dependency to existing Component
4. Dependency from existing Component
5. Dependency to existing Port
6. Dependency from existing Port
7. Dependency to existing Component, create new Port
8. Dependency from existing Component, create new Port

In the spreadsheet, this is implemented by the following values:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>W/Source Element Type</td>
<td>Source ST Filter</td>
<td>Target Element Type</td>
<td>Target ST Filter</td>
<td>Diagram Filter</td>
<td>New Element Type</td>
<td>New Element ST</td>
<td>New Link Type</td>
<td>New Link ST</td>
<td>New Link Direction</td>
<td>New Link Caption</td>
</tr>
<tr>
<td>2</td>
<td>Class</td>
<td>quick</td>
<td>Component</td>
<td>Dependency to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Class</td>
<td>quick</td>
<td>Component</td>
<td>Dependency from</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Class</td>
<td>quick</td>
<td>Component</td>
<td>Dependency to</td>
<td>Dependency to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Class</td>
<td>quick</td>
<td>Component</td>
<td>Dependency from</td>
<td>Dependency from</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Class</td>
<td>quick</td>
<td>Port</td>
<td>Dependency to</td>
<td>Dependency to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Class</td>
<td>quick</td>
<td>Port</td>
<td>Dependency from</td>
<td>Dependency from</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Class</td>
<td>quick</td>
<td>Component</td>
<td>Port</td>
<td>Dependency to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Class</td>
<td>quick</td>
<td>Component</td>
<td>Port</td>
<td>Dependency from</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the spreadsheet, this is implemented by the following values:

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>U</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New Link &amp; Element Caption</td>
<td>Create Link</td>
<td>Create Element</td>
<td>Disallow Self connector</td>
<td>No Inherit from Metatype</td>
<td>Menu Group</td>
<td>Complexity Level</td>
<td>Target Must Be Parent</td>
<td>Embed element</td>
<td>Preceeds Separator LEAF</td>
<td>Preceeds Separator GROUP</td>
</tr>
<tr>
<td>2</td>
<td>Dependency to</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Component</td>
<td>Dependency to</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Component</td>
</tr>
<tr>
<td>3</td>
<td>Dependency from</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Component</td>
<td>Dependency to</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Component</td>
</tr>
<tr>
<td>4</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dependency to</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Port</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>9</td>
<td>Dependency from</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Port</td>
<td>0</td>
<td>TRUE</td>
<td>0</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

This saves to the following CSV:

```
Class,quick,...,Component,Dependency,to,Dependency to,TRUE,TRUE,TRUE,TRUE,Component,0,...,
Class,quick,...,Component,Dependency,from,Dependency from,TRUE,TRUE,TRUE,TRUE,Component,0,...,TRUE,
Class,quick,Component,...,Dependency,from,Dependency from,TRUE,TRUE,TRUE,TRUE,Component,0,...,TRUE,
Class,quick,Port,...,Dependency,to,Dependency to,TRUE,TRUE,TRUE,0,...,TRUE,
Class,quick,Port,...,Dependency,from,Dependency from,TRUE,TRUE,TRUE,0,...,TRUE,
Class,quick,Component,...,Port,Dependency,to,Dependency to,TRUE,TRUE,TRUE,Port,0,...,TRUE,
Class,quick,Component,...,Port,Dependency,from,Dependency from,TRUE,TRUE,TRUE,Port,0,...,TRUE,
```

You can create the following profile and cut and paste the CSV data into the document artifact to test the effect.
16.1.3.3  Hide Default Quick Linker Settings

If you have a Quick Linker definition with the *Exclusive to stereotype* flag (column P) set to **TRUE**, then the default Quick Linker definitions between the given source and target are overridden. However, you might want to override the defaults without actually having a Quick Linker definition. For example, if you don’t define any Quick Links for a «*quick*» Class to another «*quick*» Class, Enterprise Architect displays the default Quick Links for a Class to another Class. To override this behaviour, create a Quick Linker definition that has the source element type, source stereotype filter, target element type and target stereotype filter fields (columns A, B, C and D) all set, with the *Exclusive to stereotype* flag (column P) set to **TRUE**, and with the new link type field (column H) set to «*none*».

For example, add this line to the example in Quick Linker Example:

```
Class,quick,Interface,,,,,<none>,,,,,,,,TRUE,,0,,,,,
```

This overrides the default Class-to-Interface Quick Links when a Quick Link is dragged from a «*quick*» Class to an Interface element.

**Note:**

This technique does not affect the automatic appearance of Dependency, Trace, Information Flow and Help items on the Quick Linker menu.

16.1.3.4  Quick Linker Object Names

List of Element Types

The following element names can be used in Quick Linker definitions:

- **Action**
- **ActionPin**
- **Activity**
- **ActivityParameter**
- **Actor**
- **Artifact**
- **Boundary**
- **CentralBufferNode**
- **Change**
- **ChoiceState**
- **Class**
- **ExitPoint**
- **ExitState**
- **ExpansionNode**
- **Feature**
- **GUIElement**
- **HistoryState**
- **InformationItem**
- **InitialActivity**
- **InitialState**
- **InteractionOccurrence**
- **Interface**
- **Issue**
- **Lifeline**
- **Port**
- **ProvidedInterface**
- **Receive**
- **RequiredInterface**
- **Requirement**
- **Screen**
- **Send**
- **Sequence**
- **Signal**
- **State**
- **StateLifeline**
<table>
<thead>
<tr>
<th>Collaboration</th>
<th>JunctionState</th>
<th>StateMachine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>MergeNode</td>
<td>Synchronization_H</td>
</tr>
<tr>
<td>Decision</td>
<td>MessageEndpoint</td>
<td>Synchronization_V</td>
</tr>
<tr>
<td>DeepHistoryState</td>
<td>n-ary Association</td>
<td>SynchState</td>
</tr>
<tr>
<td>Device</td>
<td>Node,</td>
<td>UMLDiagram</td>
</tr>
<tr>
<td>DiagramGate</td>
<td>Object</td>
<td>UseCase</td>
</tr>
<tr>
<td>EntryPoint</td>
<td>ObjectNode</td>
<td>ValueLifeline</td>
</tr>
<tr>
<td>EntryState</td>
<td>Package</td>
<td></td>
</tr>
<tr>
<td>ExecutionEnvironment</td>
<td>Part</td>
<td></td>
</tr>
</tbody>
</table>

**List of Connector Types**

The following connector names can be used in Quick Linker definitions:

<table>
<thead>
<tr>
<th>Aggregation</th>
<th>Deployment</th>
<th>PackageMerge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>Extension</td>
<td>Realization</td>
</tr>
<tr>
<td>CommunicationPath</td>
<td>Generalization</td>
<td>Redefinition</td>
</tr>
<tr>
<td>Composition</td>
<td>InterfaceLink</td>
<td>Sequence</td>
</tr>
<tr>
<td>ConnectorLink</td>
<td>Manifest</td>
<td>StateFlow</td>
</tr>
<tr>
<td>ControlFlow</td>
<td>Nesting</td>
<td>UCExtends</td>
</tr>
<tr>
<td>DelegateLink</td>
<td>ObjectFlow</td>
<td>UCIIncludes</td>
</tr>
<tr>
<td>Dependency</td>
<td>PackageImport</td>
<td>UseCase</td>
</tr>
</tbody>
</table>
16.2 MDG Technologies in SDK

The Model Driven Generation (MDG) Technologies enable Enterprise Architect users to access and use resources pertaining to a specific technology in Enterprise Architect. There are various options for an administrator or individual user to bring MDG Technologies into use with Enterprise Architect, as described in the Enterprise Architect User Guide. You should read the MDG Technology topics to understand how MDG Technologies are accessed and used within Enterprise Architect, especially the Manage MDG Technologies topic.

A further option is that Technology Developers can develop new MDG Technologies and deploy them to the project team as appropriate; this is described in the following topics:

- Create MDG Technologies
- Working With MTS Files
- Customize Toolbox Profiles
- Create Diagram Profiles
- Create Tasks Pane Profiles
- Add Icons and Logos in a Technology
- Define Validation Configuration
- Incorporate Model Templates
- Deploy an MDG Technology
16.2.1 Create MDG Technologies

Using the MDG Technology Wizard, you can create MDG Technology files that can include UML Profiles, Code Modules, Patterns, Images and Tagged Value Types. To create an MDG Technology file, follow the steps below:

1. Select the **Tools | Generate MDG Technology File** menu option. The MDG Technology Creation Wizard screen displays.

Create new MDG Technology file

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 Resources Window.

This wizard will guide you through the process of creating an MDG Technology file

2. Click on the **Next** button to proceed. The MDG Technology Wizard prompts you to:
   - Create an MDG Technology File by creating a new MDG Technology Selection (MTS) file
   - Create an MDG Technology File using an existing MTS file
   - Not use any MTS file.
An MTS file stores the selected options that you define during the creation of an MDG Technology File. If you use an MTS file, you can later modify it to add or remove specific items in the MDG Technology File. This is the recommended process.

3. Select the appropriate MTS file option. Click on the Next button.

If you selected an MTS file, the MDG Technology Wizard prompts you to save the changes in the existing MTS file or into a new MTS file. This enables you to create a modification based on the existing MTS file, while preserving the original file.

4. Select the appropriate option and click on the Next button. The MDG Technology Wizard - Create screen displays.
5. Complete the fields on this screen as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>Type the name of the MDG Technology.</td>
</tr>
<tr>
<td>Filename</td>
<td>Type or select the path and filename of the MDG Technology File (the file extension for this file is .xml).</td>
</tr>
<tr>
<td>ID</td>
<td>Type a reference for the MDG Technology File, up to 12 characters long.</td>
</tr>
<tr>
<td>Version</td>
<td>Type the version number of the MDG Technology File.</td>
</tr>
<tr>
<td>Notes</td>
<td>Type a short explanation of the functionality of the MDG Technology.</td>
</tr>
</tbody>
</table>

Select Items to be included in this Technology:

- Profiles  
- Code Modules
- Patterns
- MDA Transforms
- Tagged Value Types
- Images

Note:
If you select the Profiles checkbox, you must include a Stereotype Profile. However, if you do not select the Profiles checkbox, you can add other profiles later without including a Stereotype Profile.

6. The items selected in the Select Items to be included in this Technology panel run specific dialogs to enable selection of the specific items to be included in the MDG Technology. The methods used for selection of specific items are found in the following topics:
- Add a Profile
- Add a Pattern
- Add Tagged Values
- Add Code Modules
- Add MDA Transformations
- Add Images

7. Click on the Next button. The MDG Technology Wizard - Finish screen displays, providing information on the items included in the MDG Technology File.
8. If you have used an MTS file and want to update it, select the Save to MTS checkbox.
9. If you are satisfied with the selection of items, click on the Finish button. You can now edit the MTS file, if required, to add further items such as:
   - Diagram Profiles
   - Task Panes
   - Model Search definitions
   - Enterprise Architect UML Toolbox pages
   - Model Views
   - Icons and logos for the technology
   - Model Validation configurations
   - Model Templates.

   To make the MDG Technology File accessible to an Enterprise Architect model, you must add the technology file path to the MDG Technologies - Advanced dialog. See the Access Remote MDG Technologies topic.

16.2.1.1 Add a Profile

When creating an MDG Technology file, you can include UML 2.1-compliant profiles. To use the Profiles section of the MDG Technology Wizard, follow the steps below:

1. Follow the steps in the Create MDG Technologies topic up to and including Step 5.
2. In the Select Items to be included in this Technology panel, select the Profiles checkbox. The MDG Technology Wizard - Profile files selection dialog displays.
3. In the **Directory** field, navigate to the directory containing the required Profile or Profiles.

4. To select each required Profile individually, highlight the Profile in the **Available Files** list and click on the -- > button.

   Alternatively, to select all available Profiles, click on the -- >> button.

   **Note:**
   Make sure you include your stereotypes profile.

5. Click on the **Next** button to proceed.

**16.2.1.2 Add a Pattern**

When creating an MDG Technology file, you can include Patterns. To use the Patterns section of the MDG Technology Wizard, follow the steps below:

1. Follow the steps in the [Create MDG Technologies](#) topic up to and including Step 5.

2. In the **Select Items to be included in this Technology** panel, select the **Pattern** checkbox. The MDG Technology Wizard - Pattern files selection dialog displays.
3. In the **Directory** field, navigate to the directory containing the required Pattern or Patterns.

4. To select each required Pattern individually, highlight the Pattern in the **Available Files** list and click on the --> button. Alternatively, to select all available Patterns, click on the -->> button.

5. Click on the **Next** button to proceed.

### 16.2.1.3 Add Tagged Values

When creating an MDG Technology file, you can include Tagged Value Types. To use the Tagged Value Types section of the MDG Technology Types Wizard, follow the steps below:

1. Follow the steps in the [Create MDG Technologies](#) topic up to and including **Step 5**.

2. In the **Select Items to be included in this Technology** panel, select the **Tagged Value Types** checkbox. The **MDG Technology Wizard - Tagged Value Types** dialog displays.
3. To select each required Tagged Value Type individually, highlight the Tagged Value Type in the Available Tagged Values list and click on the -- button.

Alternatively, click on the -->> button to select all available Tagged Value Types.

4. Click on the Next button to proceed.

16.2.1.4 Add Code Modules

When creating an MDG Technology file, you can include code modules. To use the Code Modules section of the MDG Technology Wizard, follow the steps below:

1. Follow the steps in the Create MDG Technologies topic up to and including Step 5.

2. In the Select Items to be included in this Technology panel, select the Code Modules checkbox. The MDG Technology Wizard - Code Modules dialog displays.
3. Select the checkbox against each of the appropriate Code Modules (Product, Data Types, and Code Templates).

**Note:**

The code modules listed are those defined in your current project. These could be the Enterprise Architect default languages, or those you have defined yourself using code templates and the Code Template Editor. Before you can set up a code template for the new language in the editor, you must define at least one data type for the language. Once the MDG Technology file is created, it can be loaded into your current model and into other models.

4. To select the Code Options, click on the [...] button against each required code option. This enables you to select an XML document that provides additional settings for the language that are not covered by the data types or code templates.

The root node of the XML document should be CodeOptions. The child nodes should be called CodeOption and should contain a name attribute. The supported code options are as follows:

<table>
<thead>
<tr>
<th>Code Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultExtension</td>
<td>The default extension when generating code.</td>
</tr>
<tr>
<td>ImplementationExtension</td>
<td>The extension used by Enterprise Architect to generate an implementation file.</td>
</tr>
<tr>
<td>ImplementationPath</td>
<td>The relative path from the source file to generate the implementation file.</td>
</tr>
<tr>
<td>PackagePathSeparator</td>
<td>The delimiter used to separate package names when using the packagePath macro from the code templates.</td>
</tr>
<tr>
<td>DefaultSourceDirectory</td>
<td>The default path where Enterprise Architect generates new files to.</td>
</tr>
<tr>
<td>Editor</td>
<td>The external editor used for editing source of this language.</td>
</tr>
</tbody>
</table>

An example of a valid code options file is shown below.

```
<CodeOptions>
  <CodeOption name="DefaultExtension">.ext</CodeOption>
  <CodeOption name="Editor">C:\Windows\notepad.exe</CodeOption>
</CodeOptions>
```
5. Click on the Next button to proceed. You can edit the code option values for source code engineering and for each required language using the appropriate Language Options page of the Options dialog.

### 16.2.1.5 Add MDA Transforms

When creating an MDG Technology file, you can include the MDA Transformations that have been modified in the model. To use the Transform Modules section of the MDG Technology Wizard, follow the steps below:

1. Follow the steps in the Create MDG Technologies topic up to and including Step 5.
2. In the Select Items to be included in this Technology panel, select the MDA Transforms checkbox. The MDG Technology Wizard - Transform Modules dialog displays.

![MDG Technology Wizard - Transform Modules](image)

3. Click the checkbox against the template name of each required template that is present in the current model.
4. Click on the Next button to proceed.

### 16.2.1.6 Add Images

When creating an MDG Technology file, you can include the images that have been imported into the model. To use the Image Selection section of the MDG Technology Wizard, follow the steps below:

1. Follow the steps in the Create MDG Technologies topic up to and including Step 5.
2. In the Select Items to be included in this Technology section, select the Images checkbox. The MDG Technology Wizard - Image Selection dialog displays.
3. For each required model image available in the current model, select the checkbox next to the image name.

4. Click on the **Next** button to proceed.
16.2.2 Working with MTS Files

An MDG Technology Selection (.MTS) file stores the selected options that you define when creating an MDG Technology File using the MDG Technology Wizard. If you use a .MTS file, you can edit it to change the features selected when you generated the file, and to add or remove the advanced features described in this topic.

Create a .MTS File

To create a .MTS file, select the Tools | Generate MDG Technology File menu option to launch the MDG Technology Wizard, and work through the screens as described in Create MDG Technologies. On the second page, select the Create a new MTS file option.

Advanced Options For Your .MTS File

Once you have created the .MTS file, you can add:

- Your own Enterprise Architect UML Toolbox profiles
- Diagram Profiles
- Tasks Pane profiles
- Model Search definitions

Note: If you use a custom SQL search, the SQL must include `ea_guid AS CLASSGUID` and the object type.

- Model Views

Notes:

- Technology views do not store Favorite packages, only Views.
- If your exported views run searches that you have defined you must also include those searches in your MDG Technology.

- Icons and logos for the technology
- Model Validation configurations
- Model Templates

Open the .MTS file in a text editor. To make it easier for you, you can copy the following lines and paste them into the file before the last line of the file (i.e. just before the `</MDG.Selections>` lines:

```
<ModelSearches file=""/>
<UIToolboxes directory="" files=""/>
<UITaskpanes directory="" files=""/>
<ModelViews file=""/>
```

(The code for the icons, logos, model validation configurations and model templates is provided in the corresponding sections, accessed via the links in the list above.)

You can, if necessary, have more than one line for each inclusion; for example, more than one ModelSearch or, if your profiles are in different locations, more than one DiagramProfile. For each inserted line:

- In the directory attributes, enter the full path of the folder in which your Toolbox Profiles, Diagram Profiles or Task Pane Profiles are kept
- In the file attributes, enter the filename of the Model Search XML file or Model View XML file
- In the files attributes, enter a comma-separated ordered list of profile XML filenames.

Save the .MTS file.

Update the MDG Technology

Again select the Tools | Generate MDG Technology File menu option, but this time on the second page select Open an Existing MTS file and specify the file path of the .MTS file you have updated. Click on <Next> until the wizard is finished. Your MDG Technology file is updated.
16.2.3 Customize Toolbox Profiles

The following is a road map of how to create a set of custom toolboxes for Enterprise Architect.

1. Create a set of Toolbox Profiles that contain the definitions that Enterprise Architect requires to create the toolboxes.
2. Create a .MTS file containing instructions on how to build your MDG Technology. Use this .MTS file to build your MDG Technology.
3. Deploy the MDG Technology file.
4. Add some finishing touches:
   - Create hidden sub-menus
   - Override Enterprise Architect’s default toolboxes
   - Change the default icons for toolbox items.

16.2.3.1 Create Toolbox Profiles

You can create multiple toolbox profiles within an MDG Technology. Each toolbox profile contains definitions that determine what appears in the Enterprise Architect UML Toolbox when a specific toolbox page is open, either by selecting from the More tools... option in the Enterprise Architect UML Toolbox window, or by opening or creating a diagram of the type that is linked to the toolbox profile.

To create a toolbox profile, follow the steps below:

1. Create a diagram in a profile package. Give it a name by which you can refer to it later, such as MyClassDiagram. In the Notes field for the diagram give it an alias and a description in the following format:
   Alias=MyClass;Notes=Structural elements for class diagrams;
2. On the diagram, create a Class, name it ToolboxPage and give it the «metaclass» stereotype.
3. Create a «stereotype» element for each of the toolbox pages to create within your toolbox, such as MyClassElements and MyClassRelationships. Set their Alias to the text to display in the title bar of each toolbox page, such as My Class Elements and My Class Relationships respectively. Use the Notes field to define the tool-tip for each toolbox page; that is, Elements for Class Diagrams and Relationships for Class Diagrams. Use the «extends» connector to set the stereotype elements to extend ToolboxPage. See also: Toolbox Page Attributes.
4. In the «stereotype» elements, create an attribute for each toolbox item. The name of the attribute should be the name of the element to be dropped, including namespace, e.g. UML::Package, UML::Class and UML::Interface. The toolbox items display in the same order as the attributes in the Class, so make use of the attribute ordering buttons to define the order of your toolbox.

To name an attribute for an item from your own technology, precede it with your profile name as the namespace, and then follow it in brackets with the element type that you are extending (so that Enterprise Architect knows what element to create). For example, a SysML block element would appear as SysML::Block(UML::Class). See also: complete list of elements that can be extended.

You might prefer not to use names such as UML::Package or UML::Class in your toolbox, so give the attributes an Initial Value of, for example, Package and Class.
5. To save the toolbox profile, right-click the diagram and select the Save as Profile context menu option.

Incorporate Toolbox in MDG Technology

To incorporate your Toolbox profile in an MDG Technology, modify the MTS file that you use to create your MDG Technology and then rebuild the MDG Technology. For further information, see Working With MTS Files (the <UIToolboxes directory="" files="" line).

16.2.3.1.1 Toolbox Page Attributes

The following attributes can be added to a stereotype Class that extends the ToolboxPage metaclass:

- imagesOnly: if you give a toolbox page an attribute named imagesOnly with Initial Value set to true, the toolbox page displays without the text labels next to the icons
- isCommon: if you give a toolbox page an attribute named isCommon with Initial Value set to true, the toolbox page is common to all defined toolboxes while your technology is active
- isCollapsed: if you give a toolbox page an attribute named isCollapsed with Initial Value set to true, the toolbox page is initially minimized.
16.2.3.2 Create Hidden Sub-Menus

To create a sub-menu, create an additional « stereotype » element in the same toolbox profile and give it an attribute named isHidden with Initial Value of true. Define the toolbox item attributes as before. In the parent « stereotype » element, create an attribute with the identical name to the sub-menu element. The sub-menu element can have an alias.

This technique is very useful for 'disambiguating' stereotypes that can be applied to multiple metaclasses. In the example below, the « MyStereo » stereotype can be applied to either a Class or an Interface. On dragging and dropping one from the toolbox, a hidden menu displays giving the choice of Class or Interface, then the appropriate element is dropped:

```
16.2.3.3 Override Default Toolboxes

Enterprise Architect has many default Toolbox Profiles, one for each of its inbuilt diagram types. These define the Toolbox pages that are displayed, by default, every time a diagram of a specific type is opened or brought into view.

To replace one of Enterprise Architect's default toolboxes with one of your own (for example, if you have your own version of the UML::Class toolbox that you want to be opened every time a Class diagram is opened - as long as your technology is active) then include a RedefinedToolbox clause in the Notes field for the diagram properties of your Toolbox Profile diagram. For example, the profile diagram's Notes field could resemble the following:

RedefinedToolbox=UML::Class;Alias=Class;Notes=Structural elements for Class diagrams;

This states that the profile is the new Enterprise Architect default for all UML Class diagrams. For a list of inbuilt diagram types, see the names of the toolboxes.

16.2.3.4 Assign Icons To Toolbox Items

To assign an icon to a toolbox item, create a new « stereotype » element in the same toolbox profile as the toolbox item. Have the stereotype element extend a « metaclass » element named ToolboxItemImage. The « stereotype » element must have the same name as the attribute that it is assigning an image to (e.g. MyTech::MyStereo(UML::Class) in the diagram below) and must have an attribute named Icon with Initial Value set to the full path and file name of the image to be used. The image must be a 16x16 .BMP file.
16.2.3.5 Enterprise Architect Toolboxes

The following is a list of the Enterprise Architect UML Toolboxes that can be overridden:

- UML::Activity
- UML::Class
- UML::Communication
- UML::Component
- UML::Composite
- UML::Deployment
- UML::Interaction
- UML::Metamodel
- UML::Object
- UML::Profile
- UML::State
- UML::Timing
- UML::UseCase

16.2.3.6 Elements Used in Toolboxes

The following elements (all preceded with the namespace UML::) can be extended or redefined in Enterprise Architect Toolbox pages. The text in red indicates the label name displayed in the default Enterprise Architect Toolbox pages, where this differs in any way from the UML:: statement text.

- Action
- Activity
- ActivityFinal (Final)
- ActivityInitial (Initial)
- ActivityPartition (Partition)
- ActivityRegion (Region)
- Actor
- Artifact
- AssociationElement (Association)
- Boundary (for Use Cases)
- CentralBufferNode (Central Buffer Node)

- Issue
- Junction
- Lifeline
- MergeNode (Merge)
- MessageEndPoint (Endpoint or Message Endpoint)
- MessageLabel (Message Label)
- MetaClass
- Node
- Object
- ObjectBoundary (Boundary)
- ObjectControl (Control)
<table>
<thead>
<tr>
<th>Connector</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>ObjectEntity (Entity)</td>
</tr>
<tr>
<td>Choice</td>
<td>Package</td>
</tr>
<tr>
<td>Class</td>
<td>Part</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Port</td>
</tr>
<tr>
<td>Comment (Note)</td>
<td>Primitive</td>
</tr>
<tr>
<td>Component</td>
<td>Process</td>
</tr>
<tr>
<td>Constraint</td>
<td>Profile</td>
</tr>
<tr>
<td>Datastore</td>
<td>ProvidedInterface (Expose Interface)</td>
</tr>
<tr>
<td>Decision</td>
<td>ReceiveEvent (Receive)</td>
</tr>
<tr>
<td>DeploymentSpecification (Deployment Specification)</td>
<td>Requirement</td>
</tr>
<tr>
<td>Device</td>
<td>RobustBoundary (Boundary)</td>
</tr>
<tr>
<td>DocumentArtifact (Document Artifact or Document)</td>
<td>RobustControl (Control)</td>
</tr>
<tr>
<td>Entity (Information)</td>
<td>RobustEntity (Entity)</td>
</tr>
<tr>
<td>EntityObject (Entity)</td>
<td>Screen</td>
</tr>
<tr>
<td>EntryState (Entry)</td>
<td>SendEvent (Send)</td>
</tr>
<tr>
<td>Enumeration</td>
<td>SequenceBoundary (Boundary)</td>
</tr>
<tr>
<td>ExceptionHandler (Exception)</td>
<td>SequenceControl (Control)</td>
</tr>
<tr>
<td>ExecutionEnvironment (Execution Environment)</td>
<td>SequenceEntity (Entity)</td>
</tr>
<tr>
<td>ExitState (Exit)</td>
<td>Signal</td>
</tr>
<tr>
<td>Feature</td>
<td>State</td>
</tr>
<tr>
<td>FinalState (Final)</td>
<td>StateMachine (State Machine)</td>
</tr>
<tr>
<td>FlowFinalNode (Flow Final)</td>
<td>StateTimeLine (State Lifeline)</td>
</tr>
<tr>
<td>ForkJoinH (Fork/Join - Horizontal)</td>
<td>Stereotype</td>
</tr>
<tr>
<td>ForkJoinV (Fork/Join - Vertical)</td>
<td>StructuredActivity (Structured Activity)</td>
</tr>
<tr>
<td>Gate (Diagram Gate)</td>
<td>SynchState (Synch)</td>
</tr>
<tr>
<td>GUIElement (UI Control)</td>
<td>Table</td>
</tr>
<tr>
<td>HistoryState (History)</td>
<td>Terminate</td>
</tr>
<tr>
<td>InformationItem (Information Item)</td>
<td>TestCase (Test Case)</td>
</tr>
<tr>
<td>InitialState (Initial)</td>
<td>UseCase (Use Case)</td>
</tr>
<tr>
<td>InteractionFragment (Fragment)</td>
<td>UMLBoundary (Boundary)</td>
</tr>
<tr>
<td>InteractionState (State/Continuation)</td>
<td>ValueTimeLine (Value Lifeline)</td>
</tr>
<tr>
<td>Interface</td>
<td></td>
</tr>
</tbody>
</table>

### 16.2.3.7 Connectors Used In Toolboxes

The following connectors (all preceded with the namespace `UML::`) can be extended or redefined in Enterprise Architect toolboxes. The text in red indicates the label name displayed in the default Enterprise Architect Toolbox pages, where this differs in any way from the `UML::` statement text.

<table>
<thead>
<tr>
<th>Connector</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation (Aggregate)</td>
<td>NoteLink (Note Link)</td>
</tr>
<tr>
<td>Assembly</td>
<td>ObjectFlow (Object Flow)</td>
</tr>
<tr>
<td>Association (Associate)</td>
<td>Occurrence</td>
</tr>
<tr>
<td>AssociationClass (Association Class)</td>
<td>PackageImport (Package Import)</td>
</tr>
<tr>
<td>CallFromRecursion (Call)</td>
<td>PackageMerge (Package Merge)</td>
</tr>
<tr>
<td>CommunicationPath (Communication Path)</td>
<td>Precedes</td>
</tr>
<tr>
<td>Composition (Compose)</td>
<td>ProfileApplication (Application)</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Connector</td>
<td>Realization (Realize or Implements)</td>
</tr>
<tr>
<td>ControlFlow (Control Flow)</td>
<td>Recursion</td>
</tr>
<tr>
<td>Delegate</td>
<td>Redefinition</td>
</tr>
<tr>
<td>Dependency</td>
<td>Representation</td>
</tr>
<tr>
<td>Deployment</td>
<td>Represents</td>
</tr>
<tr>
<td>Extension</td>
<td>RoleBinding (Role Binding)</td>
</tr>
<tr>
<td>Generalization (Generalize or Inheritance)</td>
<td>SelfMessage (Self-Message)</td>
</tr>
<tr>
<td>InformationFlow (Information Flow)</td>
<td>TagValAssociation (Tagged Value)</td>
</tr>
<tr>
<td>InterruptFlow (Interrupt Flow)</td>
<td>TraceLink (Trace)</td>
</tr>
<tr>
<td>Invokes</td>
<td>Transition</td>
</tr>
<tr>
<td>Manifest</td>
<td>UCEntend (Extend)</td>
</tr>
<tr>
<td>Message</td>
<td>UCIInclude (Include)</td>
</tr>
<tr>
<td>Nesting</td>
<td>UseCaseLink (Use)</td>
</tr>
</tbody>
</table>
### 16.2.4 Create Diagram Profiles

#### Custom Diagram Types

You can create extended diagram types in Enterprise Architect and include them in MDG Technologies. To do this, perform the following steps.

1. Create a profile with the same name as the MDG Technology in which it is to be included; e.g. SysML.
2. Create a «stereotype» Class element which is named as the custom diagram, e.g. BlockDefinition.
3. Create a Class element and name it as one of the Built-In Diagram Types prefixed with Diagram_, for example Diagram_Logical for Class diagrams or Diagram_Use Case for Use Case diagrams.
4. Give the Diagram_x class the «metaclass» stereotype and draw an «extends» connector from the stereotype to the metaclass.
5. In the Notes field, give the stereotype Class a brief description of what the diagram is used for. This description displays in the bottom right-hand corner of the New Diagram dialog.
6. Give the Diagram_x class the following attributes as required:
   - `alias: string = Type` (where Type appears before the word ‘Diagram’ on the diagram title bar)
   - `diagramID: string = abc` (where abc is the diagram type that appears in the diagram frame label)
   - `toolbox: string = ToolboxName` (where ToolboxName is the name of the toolbox profile for the toolbox that opens automatically each time a diagram is opened)
   - `frameString: string = FrameFormatString` (where FrameFormatString is a string containing substitution macros for defining the frame title, with or without additional delimiters such as [ ]; macros that can be used are:
     - `#DGMSTEREO#`
     - `#DGMDISTO#`
     - `#DGMID#`
     - `#DGMTYPE#`
     - `#DGMALIAS#`
     - `#DGMOWNERNAME#`
     - `#DGMOWNERNAMEFULL#`
     - `#DGMNAME#`
     - `#DGMNAMEFULL#`
   - `styleex: string = TConnectorNotation=Option;` (where Option is one of UML 2.1, IDEF1X, or Information Engineering)
   - `styleex: string = ShowAsList=1;` (to make the diagram open directly into the Element List)

The following example shows the DFD diagram profile which defines a DFD diagram as an extension of the Enterprise Architect Analysis diagram.

7. Save the diagram as a profile in the usual manner.
8. If you are using the MDG Technology Wizard with an MTS file to generate your technology, edit the MTS file to get the DiagramProfile generated into your technology file. Inside the root `<MDG.Selections>` node insert the following.

   ```xml
   <DiagramProfile directory="MyPath" files="MyFile1.xml,MyFile2.xml"/>
   ``

   Where MyPath is the directory your DiagramProfile is saved in, and MyFile1.xml and MyFile2.xml are the diagram profiles to include in this technology.
16.2.4.1 Built-In Diagram Types

The following is a full list of built-in diagram types provided by Enterprise Architect.

- Activity
- Analysis
- Collaboration
- Component
- CompositeStructure
- Custom
- Deployment
- InteractionOverview
- Logical
- Object
- Package
- Sequence
- Statechart
- Timing
- Use Case

Note the use of Logical for Class diagrams and also notice the space in the middle of Use Case. These names are used in Defining Child Diagram Types or prefixed by Diagram_ in creating Diagram Profiles.
16.2.5 Create Tasks Pane Profiles

Defining Tasks Pane profiles is a four-part process:

1. Define Toolboxes
   Create any number of stereotype elements that each define a Tasks Pane toolbox page.

2. Define Contexts
   Create any number of stereotype elements that each define a named Context. A context might be when a specific diagram type is open, or when a specific element type is selected.

3. Allocate Contexts to Toolboxes
   Define the many-to-many relationships between the Tasks Pane toolboxes and the available contexts.

4. Create the Profile
   And incorporate it into your Technology.

16.2.5.1 Define Tasks Pane Toolboxes

A Tasks Pane toolbox is defined by a «stereotype» Class that extends a «metaclass» ToolboxPage element. These elements must be owned by a «profile» package. Each «stereotype» Class represents the contents of the Tasks Pane for a given context, and each attribute of the «stereotype» Class defines a command button in the Tasks Pane. The following diagram shows an example of a Tasks Pane toolbox.

The title bar of the Tasks Pane toolbox is defined by the Alias of the «stereotype» Class, in this case Debugging Profile. This example uses the following standard attribute types:

- EASystem::MenuCmd. These entries name an Enterprise Architect main menu command inside round brackets. See the complete list of inbuilt commands. Type the text to appear in the Tasks Pane into the Initial Value field.

- EASystem::ShowHelp. These entries name a page from the Enterprise Architect User Guide inside round brackets. To find out the names of pages in the Enterprise Architect User Guide, right-click the page and select the Properties menu option. Type the text to appear in the Tasks Pane into the Initial Value field.

- EASystem::ShowMovie. These entries give the URL of a movie inside round brackets. Type the text to appear in the Tasks Pane into the Initial Value field.

- SEPARATOR. This entry indicates that a separator should be placed in the Tasks Pane toolbox. If it is necessary to place multiple separators in a single toolbox, note that Enterprise Architect does not allow identically named attributes for a Class: simply change the case of one or more letters to get around the problem.

Other useful attributes include:

- isCommon: A boolean attribute with Initial Value set to True defines a Tasks Pane toolbox as context-free and common, appearing for all contexts.

- You can also run Add-In functions from the Tasks Pane.
Next Step
The next step is to create a set of Tasks Pane Contexts.

16.2.5.1.1 Built-In Tasks Pane Commands

The following Enterprise Architect commands can all be used in user-defined Tasks Pane profiles. Tasks Pane pages have attributes named in the form EASystem::MenuCmd(<CommandName>) where <CommandName> is the name chosen from the following list:

- AddDiagram
- AddElement
- AddPackage
- AutoRecordThread
- AddModelFromPattern
- Build
- BuildScripts
- ConfigureCSV
- ConfigureValidation
- CreateBaseLine
- CreateSequenceDiagram
- DebugPause
- DebugRun
- DebugStop
- Deploy
- DiagramsOnlyReport
- ElementUsage
- ExportXML
- FileNew
- FileOpen
- GenerateDDL
- GenerateWSDL
- GenerateXMLSchema
- ImplementationDetails
- ImportBinary
- ImportExportCSV
- ImportSchema
- ImportSourceDirectory
- ImportWSDL
- ImportXML
- ImportXMLSchema
- Run
- RunHTMLReport
- RunRTFReport
- SetClassifier
- ShowHideExecution
- StartDebugRecording
- StepInto
- StepOut
- StepOver
- StopDebugRecording
- Test
- TestingReport
- ToggleLevelNumbering
- TransformPackage
- TransformSelectedElements
- ValidateModel
- ViewAuditing
- ViewDebug
- ViewElementList
- ViewForum
- ViewHierarchy
- ViewMaintenance
- ViewOutput
- ViewProjectManagement
- ViewRelationships
- ViewRequirementTypes
- ViewRules
- ViewSearch
- ViewSourceCode
- ViewTaggedValues
- ViewTesting
- ViewTestingDetails
- ViewWebBrowser

16.2.5.1.2 Run Add-In Functions

To run Add-In functions from the Tasks Pane, you create an attribute in the Tasks Pane stereotype Class with the following format:

"Assembly::FunctionName()"

where Assembly is the name of the Add-In and FunctionName is the name of a public function in the Add-In. Give the attribute an initial value of the text that is to appear in the Tasks Pane. The function receives two parameters and returns a success status, as in the following VB.Net example:

```vbnet
Public Function ShowMyDiagram(ByVal Repository As EA.Repository, ByVal args As Object) As String
    Dim ret As String
    ret = Repository.SQLQuery("select ea_guid from t_diagram where diagram_type='Custom' and StyleEx like ";"MDGDgm=MyDiagrams::MyCustomDiagram;\%")
    If ret Is Nothing Then
        ShowMyDiagram = False
        Exit Function
    End If
```
16.2.5.2 Define Tasks Pane Contexts

Named Contexts

To create a context-sensitive set of Tasks Panes, you must define a set of named contexts that can be used in the definition. A named context is a «stereotype» Class which extends a «metaclass» named ToolboxContext. These elements must be owned by the same «profile» package that owns the Task Pane toolbox definitions. The context Class has one of the following attributes:

- contextDiagramType; this should have an Initial Value set to a valid diagram type
- contextItemType; this should have an Initial Value set to a valid element type
- contextKey.

Example Context Profile

Next Step

The next step is to allocate Tasks Pane contexts to Tasks Pane toolboxes.
16.2.5.3 Allocate Tasks Pane Contexts

Once you have defined your Tasks Pane toolboxes and Tasks Pane contexts, you can allocate your toolboxes to as many contexts as apply, and any number of toolboxes can be allocated to a single context. This is done by creating a «taggedValue» connector from the toolbox element to the context element. The association end at the context end must be named. The following diagram shows how this might appear.

Next Step

The next step is to Save your Tasks Pane Profile.

16.2.5.4 Save a Tasks Pane Profile

The best organization structure for the model in which you are creating your Tasks Pane Profile is:

- A single «profile» package
- Three diagrams within the «profile» package named Toolboxes, Contexts and Context Allocations
- Each toolbox page «stereotype» element is owned by the «profile» package and appears on the Toolboxes and Context Allocations diagrams
- Each context «stereotype» element is owned by the «profile» package and appears on the Contexts and Context Allocations diagrams
- Each «metaclass» element is owned by the «profile» package and appears on the Toolboxes or Contexts diagram.

From this structure, creating a Tasks Pane Profile is as simple as right-clicking the «profile» package in the Project Browser and selecting the Save Package as UML Profile context menu option.

Incorporate Tasks Pane in Technology

To incorporate your Tasks Pane profile in an MDG Technology, modify the MTS file that you use to create your MDG Technology and then rebuild your MDG Technology. For further information, see Working With MTS Files (the «UITaskpanes directory="" files=""/> line.)
### 16.2.6 Add Icons and Logos for Technology

It is possible to define an icon and a logo for a technology. The icon is a 16x16 bitmap image that is shown in the list of technologies on the left of the MDG Technologies dialog. The logo is a 64x64 bitmap image that is shown in the display pane on the top-right corner of the MDG Technologies dialog.

To specify an icon and a logo, add icon and logo attributes in the `<Technology>` node of the MTS file. Then use the MDG Technology Wizard to rebuild the technology file, and finally re-deploy the technology file and re-start Enterprise Architect as described in [Working with MTS Files](#).

Below is an example of a `<Technology>` node showing the use of the icon and logo attributes:

```xml
<Technology id="AUTOSAR"
    name="AUTOSAR"
    version="0.4"
    notes="Support for modeling using the AUTOSAR standard."
    filename="Y:\Dev\Builds\AddIns\AUTOSAR\AUTOSAR.xml"
    alias="MDG Technology for AUTOSAR"
    icon="Y:\Dev\Builds\AddIns\AUTOSAR\Icons\icon.bmp"
    logo="Y:\Dev\Builds\AddIns\AUTOSAR\Icons\logo.bmp"/>
```
16.2.7 Define Validation Configuration

The Model Validation Configuration dialog can be opened using the Project | Model Validation | Configure... menu option. Using this dialog, you can choose which sets of validation rules are and are not executed when a user performs a validation. Rather than perform this configuration manually and potentially have to change the settings every time Enterprise Architect is started and a different technology is set active, you can define the configuration settings within the MTS file.

To specify a set of rules as a white-list (i.e. anything added to this list is turned ON), open your MTS file in a text editor and copy and paste the following `<ModelValidation>` block at the top level inside the `<MDG.Selections>` block:

```xml
<ModelValidation>
    <RuleSet name="BPMNRules"/>
    <RuleSet name="MVR7F0001"/> <!-- notice you can turn on/off system rules as well! -->
</ModelValidation>
```

To specify a set of rules as a black-list (i.e. anything added to this list is turned OFF), open your MTS file in a text editor and copy and paste the following `<ModelValidation>` block at the top level inside the `<MDG.Selections>` block:

```xml
<ModelValidation isBlackList="true">
    <RuleSet name="BPMNRules"/>
    <RuleSet name="MVR7F0001"/>
</ModelValidation>
```

In the examples above, "BPMNRules" is the rule-set ID defined in the Project.DefineRuleCategory call - see Project Interface for details. "MVR7F0001" is one of Enterprise Architect's built-in rule-sets. These validation options are applied when you activate the appropriate technology. The global (default) technology has all rules turned on.
16.2.8 Incorporate Model Templates

Enterprise Architect has a number of Model Templates that can be added into a model, either on creation of the model, or at any time by right-clicking a package in the Project Browser and selecting the Add | Add Model using Wizard... context menu option. You can create your own templates and include them in your MDG Technology. The first step is to create a template package and save it to the MTS file.

Open your MTS file in a text editor and copy and paste the following <ModelTemplates> block at the top level inside the <MDG.Selections> block:

```xml
<ModelTemplates>
    <Model name="Template Name"          description="This is the description."          location="MyTemplatePackage.xml"          default="yes"          icon = "34"          filter= "Filter Name"/>
</ModelTemplates>
```

You can include as many <ModelTemplates> blocks as you have model templates. The attributes have the following meanings:

- **Model name**: The name of the model template as shown in the Select model(s) dialog, which displays when you create a new model or when you execute the Add Model using Wizard menu option.
- **description**: The text that is displayed in the Select model(s) dialog when the name is selected.
- **location**: Contains the path of the XML file that contains the XMI export of the model template package, relative to the location of the MDG Technology file. If the XMI file is in the same folder as the technology file then this just contains the file name.
- **default**: Contains either yes indicating that the model template is checked by default, or no indicating that the model template is un-checked by default.
- **icon**: Contains an index to Enterprise Architect's base icons list. To show the appropriate view icon, use one of the following values: 29 = Use Case, 30 = Dynamic; 31 = Class; 32 = Component; 33 = Deployment; 34 = Simple.
- **filter**: If you have a large number of model templates, you can group them on the Select model(s) dialog by giving all the model templates in the same group the same filter name. The filter name given appears in the Select from: list box in the Select model(s) dialog.
16.2.9 Deploy An MDG Technology

An MDG Technology can be deployed in one of two ways: as a file or from an Add-In.

Deploy From a File

To deploy your technology as a file, you have a number of choices:

- Copy it to a folder named MDGTechnologies, which you must create under your Enterprise Architect installation directory (by default this is C:\Program Files\Sparx Systems\EA). When you restart Enterprise Architect, your MDG Technology is deployed.
- Copy it to any folder in your file system, including network drives: use the Enterprise Architect Settings | MDG Technologies... menu option, press the Advanced button and add the folder to the Technologies path. This deployment method enables you to quickly and easily deploy a technology to all Enterprise Architect users on a LAN.
- Upload it to an internet or intranet location: use the Enterprise Architect Settings | MDG Technologies... menu option, press the Advanced button and add the URL to the Technologies path. This deployment method enables you to quickly and easily deploy a technology to an even wider group of Enterprise Architect users.

Deploy From an Add-in

To deploy your technology from an Add-In, you must write an EA_OnInitializeTechnologies function. The following example is written in VB.Net:

```vbnet
Public Function EA_OnInitializeTechnologies(ByVal Repository As EA.Repository) As Object
    EA_OnInitializeTechnologies = My.Resources.MyTechnology
End Function
```
16.3 Shape Scripts

Introduction
Enterprise Architect Shape Scripts enable 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 is drawn for every element of that stereotype. The following topics describe how to create and apply Shape Scripts:

- Getting Started with Shape Scripts
- Write Scripts
- Example Scripts
- Shape Editor
- Add Shape Scripts to UML Profiles
16.3.1 Getting Started With Shape Scripts

Shape Scripts are associated with stereotypes and are defined via the Stereotypes tab of the UML Types dialog. To access this dialog, select the Settings | UML menu option. Each stereotype defined can have a Shape Script.

You can create a Shape Script for an existing stereotype by selecting the stereotype from the list. Alternatively, you can create new stereotypes by clicking on the New button and giving the stereotype a name. Select a base class and click on the Save button. Once the stereotype is saved, it displays in the list.

To override the appearance, select the Shape Script radio button and then click on the Assign button. The Shape Script Editor displays.
Type the example shape scripts in the Edit window. You can click on the Refresh button in order to view the shape in the preview window.

**Note:**

If you define a composite Shape Script (such as the connector at the end of the Example Scripts topic), click on the Next Shape button to page through the components of the shape.

Once you have finished writing your Shape Script, click on the OK button. To save the Shape Script you must click on the Save button on the Stereotypes tab.

Once you have created your Shape Script for a particular stereotype, you can assign that stereotype to an element or connector. The appearance reflects the Shape Script you created. To do this, drag and drop the appropriate element or connector to your diagram.

**Note:**

Shape Scripts do not function in Sequence diagrams.

Right-click on the element or connector and click on the Properties button. Click on the Stereotype drop-down arrow, select the stereotype you created and click on the OK button. The object's shape now reflects the shape script you created.
16.3.2 Write Scripts

This topic is a detailed reference for writing shape scripts. For an introduction to writing shape scripts, see the Getting Started and Example Scripts topics.

See the following reference topics for more detailed information on shape scripting:

- Syntax Grammar
- Shape attributes
- Drawing methods
- Color queries
- Conditional branching
- Query methods
- Display item properties
- Sub-shapes
- Reserved Names
- Miscellaneous

16.3.2.1 Syntax Grammar

Grammar symbols:
- `*` = zero or more
- `+` = one or more
- `|` = or
- `;` = terminator

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShapeScript</td>
<td>::= &lt;Shape&gt;*;</td>
</tr>
<tr>
<td>Shape</td>
<td>::= &lt;ShapeDeclaratio&gt; &lt;ShapeBody&gt;;</td>
</tr>
<tr>
<td>ShapeDeclaration</td>
<td>::= &lt;ShapeType&gt; &lt;ShapeName&gt;;</td>
</tr>
<tr>
<td>ShapeType</td>
<td>::= &quot;shape&quot;</td>
</tr>
<tr>
<td>ShapeName</td>
<td>::= &lt;ReservedShapeName&gt;</td>
</tr>
<tr>
<td>ReservedShapeName</td>
<td>::= See Reserved Names for full reserved shape listing</td>
</tr>
<tr>
<td>ShapeBody</td>
<td>::= &quot;{&quot; &lt;InitialisationAttributeAssignment&gt; * &lt;DrawingStatement&gt; * &lt;SubShape&gt; &quot;}&quot;;</td>
</tr>
<tr>
<td>InitialisationAttributeAssignment</td>
<td>::= &lt;Attribute&gt; &quot;=&quot; &lt;Value&gt; &quot;;&quot;;</td>
</tr>
<tr>
<td>Attribute</td>
<td>::= See Shape Attributes for full listing of attribute names</td>
</tr>
<tr>
<td>DrawingStatement</td>
<td>::= &lt;IfElseSection&gt;</td>
</tr>
<tr>
<td>IfElseSection</td>
<td>::= &quot;if&quot; &quot;{&quot; &lt;QueryExpression&gt; &quot;}&quot; &lt;TrueSection&gt; [&lt;ElseSection&gt;];</td>
</tr>
<tr>
<td>QueryExpression</td>
<td>::= &lt;QueryName&gt; &quot;{&quot; &lt;ParameterList&gt; &quot;}&quot;;</td>
</tr>
<tr>
<td>QueryName</td>
<td>::= See Query Methods for a full listing of Query names</td>
</tr>
<tr>
<td>TrueSection</td>
<td>::= &quot;{&quot; &lt;DrawingStatement&gt; &quot;}&quot;</td>
</tr>
<tr>
<td>ElseSection</td>
<td>::= &quot;else&quot; &quot;{&quot; &lt;DrawingStatement&gt; &quot;}&quot;</td>
</tr>
<tr>
<td>Method</td>
<td>::= &lt;MethodName&gt; &quot;{&quot; &lt;ParameterList&gt; &quot;}&quot; &quot;;&quot;;</td>
</tr>
<tr>
<td>MethodName</td>
<td>::= See Drawing Methods for a full listing of method names</td>
</tr>
</tbody>
</table>

Enterprise Architect User Guide
16.3.2.2 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");
}
```

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottomAnchorOffset</td>
<td>(int,int)</td>
<td>When creating a shape script for an embedded element (e.g. Port), use this attribute to offset the shape from the bottom edge of its parent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example: <code>bottomAnchorOffset=(0,-10)</code>: move embedded element up 10 pixels from the bottom edge.</td>
</tr>
<tr>
<td>editableField</td>
<td>string</td>
<td>Defines a shape as an editable region of the element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field impacts element shapes only, line glyphs are not supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid Values: alias, name, note, stereotype</td>
</tr>
<tr>
<td>endpointy, endpointx</td>
<td>integer</td>
<td>Only used for the reserved target and source shapes for connectors; this point determines where the main connector line connects to the end shapes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0 and 0</td>
</tr>
<tr>
<td>h_align</td>
<td>string</td>
<td>Affects horizontal placement of printed text and subshapes depending on the <code>layoutType</code> attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values: left, center, or right</td>
</tr>
<tr>
<td>layoutType</td>
<td>string</td>
<td>Determines how subshapes are sized and positioned. See Subshape Layout for further details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values: leftright, topleftdown, border</td>
</tr>
<tr>
<td>leftAnchorOffset</td>
<td>(int,int)</td>
<td>When creating a shape script for an embedded element (e.g. Port), use this attribute to offset the shape from the left edge of its parent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example: <code>leftAnchorOffset=(10,0)</code>: move embedded element right 10 pixels from the left edge.</td>
</tr>
<tr>
<td>noshadow</td>
<td>string</td>
<td>Assign to true to suppress the shapes shadow from being rendered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values: true or false (default= false)</td>
</tr>
<tr>
<td>orientation</td>
<td>string</td>
<td>Applies to decoration shapes only. Determines where the decoration is positioned within the containing element glyph.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values: NW, N, NE, E, SE, S, SW, W</td>
</tr>
<tr>
<td>preferredHeight</td>
<td></td>
<td>Used by border <code>layoutType</code> - north and south</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used in drawing the source and target shapes for connectors to determine how wide the line is.</td>
</tr>
<tr>
<td>preferredWidth</td>
<td></td>
<td>Used by border <code>layoutType</code> - east and west.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used by leftright <code>layoutType</code>, shapes where <code>scalable</code> is false to determine how much space they occupy for layout purposes.</td>
</tr>
<tr>
<td>rightAnchorOffset</td>
<td>(int,int)</td>
<td>When creating a shape script for an embedded element (e.g. Port), use this attribute to offset the shape from the right edge of its parent.</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotatable</td>
<td>string</td>
<td>Set to false to prevent rotation of the shape. This attribute is only applicable to the source and target shapes for lines glyphs.</td>
</tr>
<tr>
<td>scalable</td>
<td>string</td>
<td>Set to false to stop the shape from being relatively sized to the associated diagram glyph.</td>
</tr>
<tr>
<td>topAnchorOffset</td>
<td>(int,int)</td>
<td>When creating a shape script for an embedded element (e.g. Port), use this attribute to offset the shape from the top edge of its parent.</td>
</tr>
<tr>
<td>v_align</td>
<td>string</td>
<td>Affects vertical placement of printed text and subshapes depending on the layoutType attribute.</td>
</tr>
</tbody>
</table>

### 16.3.2.3 Drawing Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addsubshape(</td>
<td>Adds a sub-shape with the name shapename that must be defined within the current shape definition.</td>
</tr>
<tr>
<td>arc( int left,</td>
<td>Draws an elliptical anticlockwise 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 to the point (startingpointx, startingpointy). The end of the arc is similarly defined by the intersection of the ellipse and the line from the center of the ellipse to the point (endingpointx, endingpointy).</td>
</tr>
<tr>
<td>arcto( int left,</td>
<td>As for the arc method, except that a line is drawn from the current position to the starting point of the arc, and then the current position is updated to the end point of the arc.</td>
</tr>
</tbody>
</table>
### Method Name

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int bottom, int startingpointx, int startingpointy, int endingpointx, int endingpointy</td>
<td>Draws a bezier curve and updates the pen position.</td>
</tr>
<tr>
<td>int controlpoint1x, int controlpoint1y, int controlpoint2x, int controlpoint2y, int endpointx, int endpointy</td>
<td>Draws a bezier curve and updates the pen position.</td>
</tr>
<tr>
<td>int width, int height</td>
<td>Sets the default size of the element. This can appear in IF and ELSE clauses with different values in each, and causes the element to be resized automatically each time the values change. For example: <code>if(HasTag(&quot;horizontal&quot;,&quot;true&quot;)) { defSize(100,20); rectangle(0,0,100,100); } else { defSize(20,100); rectangle(0,0,100,100); } </code> The above example sets the shape to the specified default size each time the Tagged Value horizontal is changed. When this is set, [Alt]+[Z] also resizes the shape to the defined dimensions. <strong>Note:</strong> The minimum value for both int width and int height is 10.</td>
</tr>
<tr>
<td>Causes Enterprise Architect to render the shape using its usual, non-Shapescript notation. Subsequent drawing commands are super-imposed over the native notation. This method is only enabled for element shape scripts; line shape scripts are not supported.</td>
<td></td>
</tr>
<tr>
<td>int left, int top, int right, int bottom</td>
<td>Draws an ellipse with extents defined by left, top, right and bottom.</td>
</tr>
<tr>
<td></td>
<td>Ends the sequence of drawing commands that define a path.</td>
</tr>
<tr>
<td>Fills the previously defined path with the current fill color, then draws its outline with the current pen.</td>
<td></td>
</tr>
<tr>
<td>Fills the previously defined path with the current fill color.</td>
<td></td>
</tr>
<tr>
<td>Hides the label specified by labelname.</td>
<td></td>
</tr>
<tr>
<td>Draws the image that has the name imageld in the Image Manager.</td>
<td></td>
</tr>
<tr>
<td>Draws a line from the current cursor position to the point specified by x and y</td>
<td></td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>int x, int y)</td>
<td>and then updates the pen cursor to that position.</td>
</tr>
<tr>
<td>moveto( int x, int y)</td>
<td>Moves the pen cursor to the point specified by x and y.</td>
</tr>
<tr>
<td>polygon( int centerx, int centery, int numberofsides, int radius, float rotation)</td>
<td>Draws a regular polygon with center at the point (centerx, centery), and numberofsides number of sides.</td>
</tr>
<tr>
<td>print( string text)</td>
<td>Prints the specified text string.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot change the font size, type or color of this text.</td>
</tr>
<tr>
<td>printifdefined( string propertyname, string truepart[, string falsepart])</td>
<td>Prints the truepart if the given property exists and has a non-empty value, otherwise prints the optional falsepart.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot change the font size, type or color of this text.</td>
</tr>
<tr>
<td>println( string text)</td>
<td>Appends a line of text to the shape and a line break.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot change the font size, type or color of this text.</td>
</tr>
<tr>
<td>printwrapped( string text)</td>
<td>Prints the specified text string, wrapped over multiple lines if the text is wider than its containing shape.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot change the font size, type or color of this text.</td>
</tr>
<tr>
<td>rectangle( int left, int top, int right, int bottom)</td>
<td>Draws a rectangle with extents at left, top, right, bottom. Values are percentages.</td>
</tr>
<tr>
<td>roundrect( int left, int top, int right, int bottom, int abs_cornerwidth, int abs_cornerheight)</td>
<td>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.</td>
</tr>
<tr>
<td>setdefaultcolors()</td>
<td>Returns the brush and pen color to the default settings, or to the user-defined colors if available. See Color Queries.</td>
</tr>
<tr>
<td>setfillcolor( int red, int green, int blue)</td>
<td>Sets the fill color.</td>
</tr>
<tr>
<td>setlinestyle( string linestyle)</td>
<td>Changes the stroke pattern for commands that use the pen. Parameters:</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>string linestyle</strong>: the following styles are valid:</td>
<td></td>
</tr>
<tr>
<td>• solid</td>
<td></td>
</tr>
<tr>
<td>• dash (short dash pattern)</td>
<td></td>
</tr>
<tr>
<td>• dot</td>
<td></td>
</tr>
<tr>
<td>• dashdot (long dash pattern)</td>
<td></td>
</tr>
<tr>
<td>• dashdotdot</td>
<td></td>
</tr>
</tbody>
</table>
| **setorigin** (string relativeTo, int xOffset, int yOffset) | Positions floating text labels relative to the main shape. relativeTo is one of N, NE, E, SE, S, SW, W, NW, CENTER.
| xOffset and yOffset are in pixels, not percentage values, and can be negative. | |
| **setpen** (int red, int green, int blue[, int penwidth]) | Sets the pen to the defined color and optionally sets the pen width.  
**Note:** This method is only for line-drawing commands. It does not affect any text commands. |
| **setpencolor** (int red, int green, int blue) | Sets the pen color.  
**Note:** This method is only for line-drawing commands. It does not affect any text commands. |
| **setpenwidth** (int penwidth) | Sets the width of the pen. Pen width should be between 1 and 5.  
**Note:** This method is only for line-drawing commands. It does not affect any text commands. |
| **showlabel** (string labelname) | Reveals the hidden label specified by labelname. |
| **startcloudpath** (puffWidth, puffHeight, noise) | Similar to **StartPath**, except that it draws the path with cloud-like curved segments (puffs). Parameters:  
• float puffWidth (default = 30), the horizontal distance between puffs  
• float puffHeight (default = 15), the vertical distance between puffs  
• float noise (default = 1.0), the randomization of the puffs’ positions. |
| **startpath()** | Starts the sequence of drawing commands that define a path. |
| **strokepath()** | Draws the outline of the previously defined path with the current pen. |

### 16.3.2.4 Color Queries

Color queries can only be used to retrieve arguments for the **SetPenColor** and **SetFillColor** commands. These queries can be used in place of the arguments.

```javascript
getUserFillColor()
ggetUserBorderColor()  
getUserFontColor()  
getUserPenSize()  

shape main  
{  
setfillcolor(getuserbordercolor());  
setpencolor(getuserfillcolor());  
rectangle(0,0,100,100);  
}
```
16.3.2.5 Conditional Branching
Shape Scripts provide condition branching with the if else statement, and query methods that evaluate to either True or False. See:
- Syntax Grammar for IF statement syntax.
- Query Methods for methods that can be used as the conditional expression for IF statements.
- Example Scripts for an example.

16.3.2.6 Query Methods
Two query methods are available for seeing if the associated element has certain tags or properties; these methods can be used as the conditional expression for an if else statement.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean HasTag(</td>
<td></td>
</tr>
<tr>
<td>string tagname,</td>
<td></td>
</tr>
</tbody>
</table>
|   [string tagvalue])        | 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])   | 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 Display Item Properties for a list of valid values for propertyname.      |

16.3.2.7 Display Item Properties
The commands print, println, and printwrapped all take a string parameter representing the text to be printed. Element and connector properties can be added to the text using the substitution macro #propertyname#.

For example: println("name: #NAME#");

In addition to the properties listed below, Tagged Values can also be displayed by prefixing the Tagged Value name with TAG:

For example: print("#TAG:condition#");

Properties Visible to Shape Scripts

Properties for Element Shape Scripts
- alias
- author
- cardinality
- classifier
- classifier.alias
- classifier.metatype
- classifier.name
- classifier.stereotype
- complexity
- concurrency
- datecreated
- datemodified
- diagram.name
- diagram.stereotype
- diagram.type
- haslinkeddocument
- isabstract
- isactive
Properties for Connector Shape Scripts

- alias
- diagram.name
- diagram.stereotype
- diagram.type
- direction
- isroot
- isleaf
- name
- notes
- source.aggregation
- source.alias
- source.changeable
- source.constraints
- source.element.name
- source.element.stereotype
- source.metatype
- source.multiplicity
- source.multiplicityisordered
- source.qualifiers
- source.stereotype
- source.targetscope
- stereotype
- subtype
- target.aggregation
16.3.2.8 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 initialization 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 position subshapes side by side, with the first added on the left, and subsequent subshapes 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 is to occupy: N, E, S, W or CENTER. Each region can only be occupied by one subshape.

A subshape that is 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 their preferredheight attribute set. In this case, the values for these attributes are treated as static lengths and do not scale glyphs.

For example:

```plaintext
shape main
{
    layouttype="topdown";
    setfillcolor(0,0,255);
    rectangle(0,0,100,100);
    addsubshape("sub",50,100,20,0);
    addsubshape("sub",50,100,30,-100);
    addsubshape("sub",50,100,40,-200);
    addsubshape("sub",50,100,50,-300);
    shape sub
    {
        setfillcolor(0,255,0);
        ellipse(0,0,100,100);
    }
}
```

The above script provides the following shape:
16.3.2.9 Reserved Names

Elements
Elements (such as Class, State or Event) have two reserved names for parts of the shape.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>The main shape is the whole shape.</td>
</tr>
<tr>
<td>label</td>
<td>Define a shape for label to give that shape a detached label.</td>
</tr>
</tbody>
</table>

Connectors
Connectors (such as Association, Dependency or Generalization) have three reserved names for parts of the shape.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>The main shape is the whole shape.</td>
</tr>
<tr>
<td>source</td>
<td>The source shape is an extra shape at the source end of the connector.</td>
</tr>
<tr>
<td>target</td>
<td>The target shape is an extra shape at the target end of the connector.</td>
</tr>
</tbody>
</table>

16.3.2.10 Miscellaneous

Return Command
Execution of the script can 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. For example:
// C Style Single Line comment
/* Multi Line
   comment supported */

String Manipulation
Not Supported.

Arithmetical Operations
Not Supported.

Variables Declaration
Not Supported.
Change ShapeScript Fonts
Not possible.

Can I apply a Shapescript without using Stereotypes?
No.
### 16.3.3 Example Scripts

Below is a selection of example Shape Scripts.

<table>
<thead>
<tr>
<th>Code</th>
<th>Result</th>
</tr>
</thead>
</table>
|//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)
} | ![BASIC SHAPES](image) |

//SINGLE CONDITIONAL SHAPE
shape main{
  if (HasTag("Trigger","Link")){
    // Only draw if the object has a Tagged Value
    // Trigger=Link
    setfillcolor(0,0,0);startpath();
    moveto(23,40);lineto(23,60);lineto(50,60);lineto(50,76);lineto(76,50);lineto(50,23);lineto(50,40);endpath();
    fillandstrokepath();return;
  }
} | ![SINGLE CONDITIONAL SHAPE](image) |

//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"))
} | ![MULTI CONDITIONAL SHAPE](image) |
Code

```plaintext
{ rectangle(22,22,78,78);
  moveto(22,22);
  lineto(50,50);
  lineto(78,22);
  return;
}
```

//SUB SHAPES

```plaintext
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);
  }
}
```

//Editable Field Shape

```plaintext
shape main {
  rectangle(0,0,100,100);
  addsubshape("namecompartment", 100, 20);
  addsubshape("stereotypeme compartment", 100, 40);
  shape namecompartment {
    h_align = "center";
    editablefield = "name";
    rectangle(0,0,100,100);
    println("name: #name#");
  }
  shape stereotypeme compartment {
    h_align = "center";
    editablefield = "stereotype";
    rectangle(0,0,100,100);
    println("stereotype: #stereotype#");
  }
}
```
### Code

```plaintext
// 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);
}

// Cloud Path Example Shape
shape main{
    StartCloudPath();
    Rectangle(0,0,100,100);
    EndPath();
    FillAndStrokePath();
}

// Connector Example
shape main{
    // draw a dashed line
    noshadow=true;
    setlinestyle("DASH");
    moveto(0,0);
    lineto(16,6);
    lineto(16,-6);
    endpath();
    fillandstrokepath();
}

shape source{
    // draw a circle at the source end
    rotatable = true;
    startpath();
    ellipse(0,6,12,-6);
    endpath();
    fillandstrokepath();
}

shape target{
    // draw an arrowhead at the target end
    rotatable = true;
    startpath();
    moveto(0,0);
    lineto(16,6);
    lineto(16,-6);
    endpath();
    fillandstrokepath();
}

// Double Line
shape main{
    noshadow=true;
    moveto(0,-10);
    lineto(100,-10);
    moveto(0,10);
    lineto(100,10);
}
```

### Result

<table>
<thead>
<tr>
<th>Code</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>// Return Statement Shape</td>
<td><img src="image1.png" alt="Return Statement Shape" /></td>
</tr>
</tbody>
</table>
| 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); |
| // Cloud Path Example Shape | ![Cloud Path Example Shape](image2.png) |
| shape main {
    StartCloudPath();
    Rectangle(0,0,100,100);
    EndPath();
    FillAndStrokePath(); |
| // Connector Example | ![Connector Example](image3.png) |
| shape main {
    // draw a dashed line
    noshadow=true;
    setlinestyle("DASH");
    moveto(0,0);
    lineto(16,6);
    lineto(16,-6);
    endpath();
    fillandstrokepath();
} |
| shape source {
    // draw a circle at the source end
    rotatable = true;
    startpath();
    ellipse(0,6,12,-6);
    endpath();
    fillandstrokepath();
} |
| shape target {
    // draw an arrowhead at the target end
    rotatable = true;
    startpath();
    moveto(0,0);
    lineto(16,6);
    lineto(16,-6);
    endpath();
    fillandstrokepath();
} |
| // Double Line | ![Double Line](image4.png) |
| shape main {
    noshadow=true;
    moveto(0,-10);
    lineto(100,-10);
    moveto(0,10);
    lineto(100,10); |
16.3.4 **Shape Editor**

The **Shape Editor** enables you to create **Shape Scripts**. To access the **Shape Editor**, follow the steps below:

1. Select the **Settings | UML** menu option; the **UML Types** dialog displays, defaulted to the **Stereotypes** tab.
2. Type a name in the **Stereotype** field, or click on an the required stereotype in the list.
3. From the **Override Appearance** panel, select the **Shape Script** radio button.
4. Click on the **Assign** button. The **Shape Editor** dialog displays.

<table>
<thead>
<tr>
<th>Option</th>
<th>Use to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>Select the ShapeScript version.</td>
</tr>
<tr>
<td>Import</td>
<td>Import a shape script from a text file.</td>
</tr>
<tr>
<td>Export</td>
<td>Export a shape script to a text file.</td>
</tr>
<tr>
<td>OK</td>
<td>Exit from the <strong>Shape Editor</strong>, don't forget to save your script from the <strong>Stereotypes</strong> tab. See <strong>Getting Started</strong>.</td>
</tr>
<tr>
<td>Next Shape</td>
<td>Rotate though the multiple shape definitions.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Parse your script and display the result in the <strong>Preview</strong> window.</td>
</tr>
</tbody>
</table>
16.4 Tagged Value Types

A range of predefined Tagged Values have been created in Enterprise Architect to enable you to rapidly create masked Tagged Values. This enables you to create structured tags that adhere to a specific format. For example, for model features that use the predefined tag Boolean you can use the Tagged Values window to assign a value of True or False and no other value, as described in the Enterprise Architect User Guide.

In addition, you can define a custom masked tag type enabling you to define an almost unlimited number of structured tag types.

**Note:**

You can transport Tagged Value Type definitions between models, using the Export Reference Data and Import Reference Data options on the Tools menu. Tagged Value Types are exported as Property Types.
16.4.1 Create Structured Tags

To create a structured tag, follow the steps below:

1. Select the Settings | UML menu option. The UML Types dialog displays. Select the Tagged Value Types tab.

2. Click on the New button.

3. In the Tag Name field type an appropriate tag name.

4. In the Description field type the purpose of the tag, if required.

5. In the Detail field type a value for the type of Tagged Value, and any extra information as required. In the example above the predefined values have been set for a Spin type, with the Upper and Lower Bound for the field.

6. Click on the Save button.

The tag type displays in the Defined Tag Types list.
### 16.4.2 Predefined Tagged Value Types

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

**Note:**
In the table below, Tagged Value Type and Format entries are case-sensitive.

<table>
<thead>
<tr>
<th>Tagged Value Type</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Type=Integer;</td>
<td>Enables entry of an Integer value.</td>
</tr>
<tr>
<td>Float, Decimal, Double</td>
<td>Type=Float; Type=Decimal; Type=Double;</td>
<td>Enable entry of a Float, Decimal or Double value. These types all map to the same type of data.</td>
</tr>
<tr>
<td>String</td>
<td>Type=String;</td>
<td>Enables entry of a String value.</td>
</tr>
<tr>
<td>Enum</td>
<td>Type=Enum; Values=Val1,Val2,Val3; Default=Val2;</td>
<td>Enables definition of a comma-separated list, where Val1, Val2 and Val3 represent values in the list and Default represents the default value of the list.</td>
</tr>
<tr>
<td>Const</td>
<td>Type=Const; Default=Val;</td>
<td>Enables creation of a read-only constant value.</td>
</tr>
<tr>
<td>Color</td>
<td>Type=Color;</td>
<td>Enables input of a color value from a color chooser menu.</td>
</tr>
<tr>
<td>Custom</td>
<td>Type= Custom;</td>
<td>Enables you to create your own template for predefined types; more information is provided in the Custom Tagged Values Type topic.</td>
</tr>
<tr>
<td>DateTime</td>
<td>Type=DateTime;</td>
<td>Enables input of the date and time for the Tagged Value from a calendar menu.</td>
</tr>
<tr>
<td>Boolean</td>
<td>Type=Boolean;</td>
<td>Enables input of a true or false value.</td>
</tr>
<tr>
<td>Memo</td>
<td>Type=Memo;</td>
<td>Enables input of large and complex tag values.</td>
</tr>
<tr>
<td>Spin</td>
<td>Type=Spin; LowerBound=x; UpperBound=x;</td>
<td>Enables creation of a spin control with the value of LowerBound being the lowest value and UpperBound being the highest value.</td>
</tr>
<tr>
<td>File</td>
<td>Type=File;</td>
<td>Enables input of a filename from a file browser dialog. The named file can be launched in its default application.</td>
</tr>
<tr>
<td>Classifier</td>
<td>Type=Classifier; Values=Type1,Type2; Stereotypes=Stereotype 1;</td>
<td>Deprecated. Use RefGUID and RefGUIDList. Returns the name of a user-selected element from the model, where Type1 and Type2 specify one or more allowed element types and Stereotype1 represents an allowed stereotype.</td>
</tr>
<tr>
<td>RefGUID</td>
<td>Type=RefGUID; Values=Type1,Type2; Stereotypes=Stereotype 1;</td>
<td>Enables the Tagged Value to reference an element from the model by specifying the element’s GUID, where Type1 and Type2 specify one or more allowed diagram objects (such as Class, Component, Attribute or Operation, and Stereotype1 represents an allowed stereotype. Set the classifier, attribute or operation for a Tagged Value of this type by clicking on the [...] button against the Tagged Value in the Tagged Value window.</td>
</tr>
<tr>
<td>RefGUIDList</td>
<td>Type=RefGUIDList; Values=Type1,Type2; Stereotypes=Stereotype 1;</td>
<td>Enables the Tagged Value to reference a list of elements from the model by specifying each element's GUID, where Type1 and Type2 specify one or more allowed diagram objects (such as Class or Component) and Stereotype1 represents an allowed stereotype.</td>
</tr>
</tbody>
</table>
Set the classifier for a Tagged Value of this type by clicking on the [...] button against the Tagged Value in the Tagged Value window.

Tag Filters

You can restrict where a predefined Tagged Value is available. The following table details filters that can be used to restrict where a Tagged Value can be applied.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
</table>
| AppliesTo      | AppliesTo=Type1,Type2; | 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; | Restricts the stereotypes that this tag belongs to, where S1 and S2 are the allowed stereotypes. |
### 16.4.3 Predefined Reference Data

This table details the predefined Reference Data Tagged Value types that are used to return the values held in a relevant table in Enterprise Architect, along with the syntax required for their use:

<table>
<thead>
<tr>
<th>Tagged ValueType</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Type=Enum; List=Authors;</td>
<td>Returns a drop-down list of the Authors that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Type=Enum; List=Cardinality;</td>
<td>Returns a drop-down list of the Cardinality types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>Clients</td>
<td>Type=Enum; List=Clients;</td>
<td>Returns a drop-down list of the Clients that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>ComplexityTypes</td>
<td>Type=Enum; List=ComplexityTypes;</td>
<td>Returns a drop-down list of the Complexity types that have been defined for the Enterprise Architect model. Still complexity types can be exported and imported as project reference data, but they cannot be updated and so are effectively standard across all projects.</td>
</tr>
<tr>
<td>ConstraintTypes</td>
<td>Type=Enum; List=ConstraintTypes;</td>
<td>Returns a drop-down list of the Constraint types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>EffortTypes</td>
<td>Type=Enum; List=EffortTypes;</td>
<td>Returns a drop-down list of the Effort types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>MaintenanceTypes</td>
<td>Type=Enum; List=MaintenanceTypes;</td>
<td>Returns a drop-down list of the Maintenance types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>ObjectTypes</td>
<td>Type=Enum; List=ObjectTypes;</td>
<td>Returns a drop-down list of the Object types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>Phases</td>
<td>Type=Enum; List=Phases;</td>
<td>Returns a drop-down list of the Phases that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>ProblemTypes</td>
<td>Type=Enum; List=ProblemTypes;</td>
<td>Returns a drop-down list of the Problem types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>RoleTypes</td>
<td>Type=Enum; List=RoleTypes;</td>
<td>Returns a drop-down list of the Role types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>RequirementTypes</td>
<td>Type=Enum; List=RequirementTypes;</td>
<td>Returns a drop-down list of the Requirement types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>Resources</td>
<td>Type=Enum; List=Resources;</td>
<td>Returns a drop-down list of the Resources that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>RiskTypes</td>
<td>Type=Enum; List=RiskTypes;</td>
<td>Returns a drop-down list of the Risk types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>RTFTemplates</td>
<td>Type=Enum; List=RTFTemplates;</td>
<td>Returns a drop-down list of the RTF Templates that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>ScenarioTypes</td>
<td>Type=Enum; List=ScenarioTypes;</td>
<td>Returns a drop-down list of the Scenario types that have been defined for the Enterprise Architect model.</td>
</tr>
<tr>
<td>TestTypes</td>
<td>Type=Enum; List=TestTypes;</td>
<td>Returns a drop-down list of the Test types that have been defined for the Enterprise Architect model.</td>
</tr>
</tbody>
</table>

See the [Create Predefined Reference Data Tagged Value Type](#) topic.

**Note:**

You can transport the predefined reference data between models, using the [Export Reference Data](#) and [Import Reference Data](#) options on the [Tools](#) menu.
16.4.4 Create Predefined Reference Data

To create a predefined reference data Tagged Value type, follow the steps below:

1. Select the Settings | UML menu option. The UML Types dialog displays; select the Tagged Value Types tab.

2. In the Tag field type an appropriate tag name.

3. In the Description field type a description of the purpose of the tag if required.

4. In the Detail field type a value for the Tagged Value type, and any extra information as required by the predefined type. In the example above, the predefined value's return values have been set to return the values for all of the Authors in the Enterprise Architect model.

This enables you to assign any of the previously defined Authors to a model feature (Model features that can have Tagged Values applied to them are detailed in the Model Elements and Features with Tagged Values topic in the Enterprise Architect User Guide).
Note:
If the values in the reference data are changed after the Tagged Value Type is created, Enterprise Architect must be reloaded in order to reflect the changes with the Tagged Value Type.
16.4.5 Create Custom Tagged Value Type

Creating a custom Tagged Value gives you great flexibility for creating your own masked Tagged Values. To create a masked Tagged Value follow the steps below:

1. Select the Settings | UML menu option. The UML Types dialog displays; select the Tagged Value Types tab.
2. In the Tag field type an appropriate name for the tag.
3. In the Description field type a description of the purpose of the tag.
4. In the Detail field type Type=Custom;

By defining the type as Custom, you can set up the appropriate mask using the following characters to define the format of the mask:

<table>
<thead>
<tr>
<th>Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Enables the Tagged Value to display digits only.</td>
</tr>
<tr>
<td>d</td>
<td>Enables the Tagged Value to display digits or spaces.</td>
</tr>
<tr>
<td>+</td>
<td>Enables the use of +, - or spaces.</td>
</tr>
<tr>
<td>C</td>
<td>Enables the use of alpha characters only.</td>
</tr>
<tr>
<td>c</td>
<td>Enables the Tagged Value to be an alpha character or a space.</td>
</tr>
<tr>
<td>A</td>
<td>Enables the use of alphanumeric characters.</td>
</tr>
<tr>
<td>a</td>
<td>Enables the Tagged Value to use alphanumeric values or a space.</td>
</tr>
</tbody>
</table>

In the diagram below the Mask configuration option shows syntax that first defines seven blank spaces, which are 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 enableable information can 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 enableable characters as digits or spaces. The hyphen is 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 that 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 is to be occupied by data input by you and defined in the Mask option.
### Tag Name: MemberZip

**Description:** Zip code

**Type:** Custom

**Mask:** dddddd-dddd

**Template-Date:** Zip: _______;

---

### Defined Tag Types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Database field</td>
</tr>
<tr>
<td>Membership</td>
<td>Membership</td>
</tr>
<tr>
<td>Role</td>
<td>Person role</td>
</tr>
<tr>
<td>Software</td>
<td>Software component</td>
</tr>
</tbody>
</table>
The Code Template Framework (CTF) is used during forward engineering of UML models. It is introduced in the Enterprise Architect User Guide. This section of the SDK discusses how you customize the way in which Enterprise Architect generates source code, using the Code Template Editor.

Enterprise Architect'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. The Base Templates provided in Enterprise Architect are described in the Enterprise Architect User Guide.
16.5.1 Code Template Syntax

Code Templates are written as plain text, using Enterprise Architect's code template editor (see The Code Template Editor). The template syntax centers on three basic constructs:

- **Literal Text**
- **Macros**
- **Variables**

Templates can contain any or all of these constructs.

16.5.1.1 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:

```plaintext
%PI=" "%%CONVERT_SCOPE(classScope)%%classStereotype=="static" ? "static": ""%%classStereotype=="final" ? "final": ""%%classStereotype=="static final" ? "static final": ""%%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 must be generated from within the templates, they can be safely reproduced using the following direct substitution macros:

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%dl%</td>
<td>Produces a literal $ character.</td>
</tr>
<tr>
<td>%pc%</td>
<td>Produces a literal % character.</td>
</tr>
<tr>
<td>%qt%</td>
<td>Produces a literal &quot; character.</td>
</tr>
</tbody>
</table>

16.5.1.2 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 item from the Class Declaration template:

```plaintext
... 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:

```plaintext
... class Foo ...
```

16.5.1.2.1 Template Substitution Macros

The template substitution macros correspond to the 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> can 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
- AttributeNotes
- Attribute
- Class
- ClassImpl
- ClassBase
- ClassBody
- ClassBodyImpl
- ClassDeclaration
- ClassDeclarationImpl
- ClassInherits
- ClassInterface
- ClassNotes
- ClassParameter
- ClassImpl
- ImportSection
- ImportSectionImpl
- InnerClass
- InnerClassImpl
- LinkedAttribute
- LinkedAttributeNotes
- LinkedAttributeDeclaration
- LinkedClassBase
- LinkedClassInterface
- Namespace
- NamespaceBody
- NamespaceDeclaration
- NamespaceImpl
- Operation
- OperationBody
- OperationBodyImpl
- OperationDeclaration
- OperationDeclarationImpl
- OperationImpl
- OperationNotes
- Parameter

### 16.5.1.2.2 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.

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attAlias</td>
<td>Attributes dialog: Alias.</td>
</tr>
<tr>
<td>attallowDuplicates</td>
<td>Attributes Detail dialog: Allow Duplicates checkbox.</td>
</tr>
<tr>
<td>attClassifierGUID</td>
<td>The unique GUID for the classifier of the current attribute.</td>
</tr>
<tr>
<td>attCollection</td>
<td>Attributes Detail dialog: Attribute is a Collection checkbox.</td>
</tr>
<tr>
<td>attConst</td>
<td>Attributes dialog: Const checkbox.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>attContainerType</td>
<td>Attributes Detail dialog: Container Type.</td>
</tr>
<tr>
<td>attContainerType</td>
<td>Attributes dialog: Containment.</td>
</tr>
<tr>
<td>attContainment</td>
<td>Attributes dialog: Derived checkbox.</td>
</tr>
<tr>
<td>attDerived</td>
<td>Attributes dialog: Is Literal checkbox.</td>
</tr>
<tr>
<td>attGUID</td>
<td>The unique GUID for the current attribute.</td>
</tr>
<tr>
<td>attInitial</td>
<td>Attributes dialog: Initial.</td>
</tr>
<tr>
<td>attIsEnumLiteral</td>
<td>Attributes dialog: Is Literal checkbox.</td>
</tr>
<tr>
<td>attLength</td>
<td>Column dialog: Length.</td>
</tr>
<tr>
<td>attLowerBound</td>
<td>Attributes Detail dialog: Lower Bound.</td>
</tr>
<tr>
<td>attName</td>
<td>Attributes dialog: Name.</td>
</tr>
<tr>
<td>attNotes</td>
<td>Attributes dialog: Notes.</td>
</tr>
<tr>
<td>attProperty</td>
<td>Attributes dialog: Property checkbox.</td>
</tr>
<tr>
<td>attOrderedMultiplicity</td>
<td>Attributes Detail dialog: Ordered Multiplicity checkbox.</td>
</tr>
<tr>
<td>attQualType</td>
<td>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.</td>
</tr>
<tr>
<td>attScope</td>
<td>Attributes dialog: Scope.</td>
</tr>
<tr>
<td>attStatic</td>
<td>Attributes dialog: Static checkbox.</td>
</tr>
<tr>
<td>attStereotype</td>
<td>Attributes dialog: Stereotype.</td>
</tr>
<tr>
<td>attType</td>
<td>Attributes dialog: Type.</td>
</tr>
<tr>
<td>attUpperBound</td>
<td>Attributes Detail dialog: Upper Bound.</td>
</tr>
<tr>
<td>attVolatile</td>
<td>Attributes Detail dialog: Transient checkbox.</td>
</tr>
<tr>
<td>classAbstract</td>
<td>Class dialog: Abstract checkbox.</td>
</tr>
<tr>
<td>classAlias</td>
<td>Class dialog: Alias.</td>
</tr>
<tr>
<td>classArguments</td>
<td>Class Detail dialog: C++ Templates: Arguments.</td>
</tr>
<tr>
<td>classAuthor</td>
<td>Class dialog: Author.</td>
</tr>
<tr>
<td>classBaseName</td>
<td>Type Hierarchy dialog: Class Name (for use where no connector exists between child and base Classes).</td>
</tr>
<tr>
<td>classBaseScope</td>
<td>The scope of the inheritance as reverse engineered. (For use where no connector exists between child and base Classes.)</td>
</tr>
<tr>
<td>classBaseVirtual</td>
<td>The virtual property of the inheritance as reverse engineered. (For use where no connector exists between child and base Classes.)</td>
</tr>
<tr>
<td>classComplexity</td>
<td>Class dialog: Complexity.</td>
</tr>
<tr>
<td>classCreated</td>
<td>The date and time the Class was created.</td>
</tr>
<tr>
<td>classGUID</td>
<td>The unique GUID for the current Class.</td>
</tr>
<tr>
<td>classHasParent</td>
<td>True, if the Class in scope has one or more base Classes.</td>
</tr>
<tr>
<td>classImports</td>
<td>Code Gen dialog: Imports.</td>
</tr>
<tr>
<td>classIsActive</td>
<td>Class Advanced dialog: Is Active checkbox.</td>
</tr>
<tr>
<td>classIsInstantiated</td>
<td>True, if the Class is an instantiated template Class.</td>
</tr>
<tr>
<td>classIsLeaf</td>
<td>Class Advanced dialog: Is Leaf checkbox.</td>
</tr>
<tr>
<td>classIsRoot</td>
<td>Class Advanced dialog: Is Root checkbox.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>classKeywords</td>
<td>Class dialog: Keywords.</td>
</tr>
<tr>
<td>classLanguage</td>
<td>Class dialog: Language.</td>
</tr>
<tr>
<td>classMacros</td>
<td>A space separated list of macros defined for the Class.</td>
</tr>
<tr>
<td>classModified</td>
<td>The date and time the Class was last modified.</td>
</tr>
<tr>
<td>classMultiplicity</td>
<td>Class Advanced dialog:Multiplicity.</td>
</tr>
<tr>
<td>className</td>
<td>Class dialog: Name.</td>
</tr>
<tr>
<td>classNotes</td>
<td>Class dialog: Note.</td>
</tr>
<tr>
<td>classParamDefault</td>
<td>Class Detail dialog.</td>
</tr>
<tr>
<td>classParamName</td>
<td>Class Detail dialog.</td>
</tr>
<tr>
<td>classParamType</td>
<td>Class Detail dialog.</td>
</tr>
<tr>
<td>classPersistence</td>
<td>Class dialog: Persistence.</td>
</tr>
<tr>
<td>classPhase</td>
<td>Class dialog: Phase.</td>
</tr>
<tr>
<td>classQualifiedName</td>
<td>The Class name prefixed by its outer Classes. Class names are separated by double colons (::).</td>
</tr>
<tr>
<td>classScope</td>
<td>Class dialog: Scope.</td>
</tr>
<tr>
<td>classStereotype</td>
<td>Class dialog: Stereotype.</td>
</tr>
<tr>
<td>classStatus</td>
<td>Class dialog: Status.</td>
</tr>
<tr>
<td>classVersion</td>
<td>Class dialog: Version.</td>
</tr>
<tr>
<td>connectorAlias</td>
<td>Connector Properties dialog: Alias.</td>
</tr>
<tr>
<td>connectorDestAccess</td>
<td>Connector target role properties: Access.</td>
</tr>
<tr>
<td>connectorDestAggregation</td>
<td>Connector target role properties: Aggregation.</td>
</tr>
<tr>
<td>connectorDestAlias</td>
<td>Connector target role properties: Alias.</td>
</tr>
<tr>
<td>connectorDestAllowDuplicates</td>
<td>Connector target role properties: Allow Duplicates.</td>
</tr>
<tr>
<td>connectorDestChangeable</td>
<td>Connector target role properties: Changeable.</td>
</tr>
<tr>
<td>connectorDestConstraint</td>
<td>Connector target role properties: Constraint.</td>
</tr>
<tr>
<td>connectorDestContainment</td>
<td>Connector target role properties: Containment.</td>
</tr>
<tr>
<td>connectorDestDerived</td>
<td>Connector target role properties: Derived.</td>
</tr>
<tr>
<td>connectorDestDerivedUnion</td>
<td>Connector target role properties: DerivedUnion.</td>
</tr>
<tr>
<td>connectorDestElem*</td>
<td>A set of macros that access a property of the element at the target end of a connector. The * (asterisk) is a wildcard that corresponds to any class substitution macro in this list; for example: connectorDestElemAlias (classAlias), connectorDestElemAuthor (classAuthor).</td>
</tr>
<tr>
<td>connectorDestElemType</td>
<td>The element type of the connector destination element. (Separate from the connectorDestElem* macros because there is no classType substitution macro.).</td>
</tr>
<tr>
<td>connectorDestMemberType</td>
<td>Connector target role properties: Member Type.</td>
</tr>
<tr>
<td>connectorDestMultiplicity</td>
<td>Connector target role properties: Multiplicity.</td>
</tr>
<tr>
<td>connectorDestNavigability</td>
<td>Connector target role properties: Navigability.</td>
</tr>
<tr>
<td>connectorDestNotes</td>
<td>Connector target role properties: Notes.</td>
</tr>
<tr>
<td>connectorDestOrdered</td>
<td>Connector target role properties: Multiplicity is Ordered.</td>
</tr>
<tr>
<td>connectorDestOwned</td>
<td>Connector target role properties: Owned.</td>
</tr>
<tr>
<td><strong>Macro Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>connectorDestQualifier</td>
<td>Connector target role properties: <strong>Qualifier</strong>.</td>
</tr>
<tr>
<td>connectorDestRole</td>
<td>Connector target role properties: <strong>Role</strong>.</td>
</tr>
<tr>
<td>connectorDestScope</td>
<td>Connector target role properties: <strong>Target Scope</strong>.</td>
</tr>
<tr>
<td>connectorDestStereotype</td>
<td>Connector target role properties: <strong>Stereotype</strong>.</td>
</tr>
<tr>
<td>connectorDirection</td>
<td>Connector properties: <strong>Direction</strong>.</td>
</tr>
<tr>
<td>connectorEffect</td>
<td>Transition Constraints dialog: <strong>Effect</strong>.</td>
</tr>
<tr>
<td>connectorGuard</td>
<td><strong>Object Flow</strong> and Transition Constraints dialog: <strong>Guard</strong>.</td>
</tr>
<tr>
<td>connectorGUID</td>
<td>The unique GUID for the current connector.</td>
</tr>
<tr>
<td>connectorName</td>
<td>Connector properties: <strong>Name</strong>.</td>
</tr>
<tr>
<td>connectorNotes</td>
<td>Connector properties: <strong>Notes</strong>.</td>
</tr>
<tr>
<td>connectorSourceAccess</td>
<td>Connector source role properties: <strong>Access</strong>.</td>
</tr>
<tr>
<td>connectorSourceAggregation</td>
<td>Connector source role properties: <strong>Aggregation</strong>.</td>
</tr>
<tr>
<td>connectorSourceAlias</td>
<td>Connector source role properties: <strong>Alias</strong>.</td>
</tr>
<tr>
<td>connectorSourceAllowDuplicates</td>
<td>Connector source role properties: <strong>Allow Duplicates</strong>.</td>
</tr>
<tr>
<td>connectorSourceChangeable</td>
<td>Connector source role properties: <strong>Changeable</strong>.</td>
</tr>
<tr>
<td>connectorSourceConstraint</td>
<td>Connector source role properties: <strong>Constraint</strong>.</td>
</tr>
<tr>
<td>connectorSourceContainment</td>
<td>Connector source role properties: <strong>Containment</strong>.</td>
</tr>
<tr>
<td>connectorSourceDerived</td>
<td>Connector source role properties: <strong>Derived</strong>.</td>
</tr>
<tr>
<td>connectorSourceDerivedUnion</td>
<td>Connector source role properties: <strong>Derived Union</strong>.</td>
</tr>
<tr>
<td>connectorSourceElem*</td>
<td>A set of macros that access a property of the element at the source end of a connector. The * (asterisk) is a wildcard that corresponds to any class substitution macro in this list; for example: connectorSourceElemAlias (classAlias), connectorSourceElemAuthor (classAuthor).</td>
</tr>
<tr>
<td>connectorSourceElemType</td>
<td>The element type of the connector source element. (Separate from the connectorSourceElem* macros because there is no classType substitution macro.)</td>
</tr>
<tr>
<td>connectorSourceMemberType</td>
<td>Connector source role properties: <strong>Member Type</strong>.</td>
</tr>
<tr>
<td>connectorSourceMultiplicity</td>
<td>Connector source role properties: <strong>Multiplicity</strong>.</td>
</tr>
<tr>
<td>connectorSourceNavigability</td>
<td>Connector source role properties: <strong>Navigability</strong>.</td>
</tr>
<tr>
<td>connectorSourceNotes</td>
<td>Connector source role properties: <strong>Notes</strong>.</td>
</tr>
<tr>
<td>connectorSourceOrdered</td>
<td>Connector source role properties: <strong>Multiplicity is Ordered</strong>.</td>
</tr>
<tr>
<td>connectorSourceOwned</td>
<td>Connector source role properties: <strong>Owned</strong>.</td>
</tr>
<tr>
<td>connectorSourceQualifier</td>
<td>Connector source role properties: <strong>Qualifier</strong>.</td>
</tr>
<tr>
<td>connectorSourceRole</td>
<td>Connector source role properties: <strong>Role</strong>.</td>
</tr>
<tr>
<td>connectorSourceScope</td>
<td>Connector source role properties: <strong>Target Scope</strong>.</td>
</tr>
<tr>
<td>connectorSourceStereotype</td>
<td>Connector source role properties: <strong>Stereotype</strong>.</td>
</tr>
<tr>
<td>connectorStereotype</td>
<td>Connector properties: <strong>Stereotype</strong>.</td>
</tr>
<tr>
<td>connectorTrigger</td>
<td>Transition Constraints dialog: <strong>Trigger</strong>.</td>
</tr>
<tr>
<td>connectorType</td>
<td>The connector type. e.g. Association or Generalization.</td>
</tr>
<tr>
<td>connectorWeight</td>
<td><strong>Object Flow Constraints</strong> dialog: <strong>Weight</strong>.</td>
</tr>
<tr>
<td>eaDateTime</td>
<td>The current time with format: DD-MMM-YYYY HH:MM:SS AM/PM.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>eaGUID</td>
<td>A unique GUID for this generation.</td>
</tr>
<tr>
<td>elemType</td>
<td>The element type: Interface or Class.</td>
</tr>
<tr>
<td>fileExtension</td>
<td>The file type extension of the file being generated.</td>
</tr>
<tr>
<td>fileName</td>
<td>The name of the file being generated.</td>
</tr>
<tr>
<td>fileNameImpl</td>
<td>The filename of the implementation file for this generation, if applicable.</td>
</tr>
<tr>
<td>fileHeaders</td>
<td>Code Gen dialog: Headers.</td>
</tr>
<tr>
<td>fileImports</td>
<td>Code Gen dialog: Imports. For supported languages this also includes</td>
</tr>
<tr>
<td></td>
<td>dependencies derived from associations.</td>
</tr>
<tr>
<td>filePath</td>
<td>The full path of the file being generated.</td>
</tr>
<tr>
<td>filePathImpl</td>
<td>The full path of the implementation file for this generation, if applicable.</td>
</tr>
<tr>
<td>genOptCDefaultAttributeType</td>
<td>C Specifications dialog: Default Attribute Type.</td>
</tr>
<tr>
<td>genOptCGenMethodNotesInBody</td>
<td>C Specifications dialog: Method Notes In Implementation.</td>
</tr>
<tr>
<td>genOptCGenMethodNotesInHeader</td>
<td>C Specifications dialog: Method Notes In Header.</td>
</tr>
<tr>
<td>genOptCSynchNotes</td>
<td>C Specifications dialog: Synchronize Notes in Generation.</td>
</tr>
<tr>
<td>genOptCNamespaceDelimiter</td>
<td>C Specifications dialog: Namespace Delimiter.</td>
</tr>
<tr>
<td>genOptCOperationRefParam</td>
<td>C Specifications dialog: Reference as Operation Parameter.</td>
</tr>
<tr>
<td>genOptCConstructorName</td>
<td>C Specifications dialog: Default Constructor Name.</td>
</tr>
<tr>
<td>genOptCDestructorName</td>
<td>C Specifications dialog: Default Destructor Name.</td>
</tr>
<tr>
<td>genOptCPPCommentStyle</td>
<td>C++ Specifications dialog: Comment Style.</td>
</tr>
<tr>
<td>genOptCPPDefaultAttributeType</td>
<td>C++ Specifications dialog: Default Attribute Type.</td>
</tr>
<tr>
<td>genOptCPPDefaultReferenceType</td>
<td>C++ Specifications dialog: Default Reference Type.</td>
</tr>
<tr>
<td>genOptCPPGenMethodNotesInHeader</td>
<td>C++ Specifications dialog: Method Notes In Header checkbox.</td>
</tr>
<tr>
<td>genOptCPPGenMethodNotesInBody</td>
<td>C++ Specifications dialog: Method Notes In Body checkbox.</td>
</tr>
<tr>
<td>genOptCPPGetPrefix</td>
<td>C++ Specifications dialog: Get Prefix.</td>
</tr>
<tr>
<td>genOptCPPSetPrefix</td>
<td>C++ Specifications dialog: Set Prefix.</td>
</tr>
<tr>
<td>genOptCPPSynchronizeCPPFile</td>
<td>C++ Specifications dialog: Synchronize Notes.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>genOptCSDefaultAttributeType</td>
<td>C# Specifications dialog: Default Attribute Type.</td>
</tr>
<tr>
<td>genOptCSSourceExtension</td>
<td>C# Specifications dialog: Default file extension.</td>
</tr>
<tr>
<td>genOptCSGenDispose</td>
<td>C# Specifications dialog: Generate Dispose.</td>
</tr>
<tr>
<td>genOptCSGenFinalizer</td>
<td>C# Specifications dialog: Generate Finalizer.</td>
</tr>
<tr>
<td>genOptCSGenNamespace</td>
<td>C# Specifications dialog: Generate Namespace.</td>
</tr>
<tr>
<td>genOptDefaultAssocAttName</td>
<td>Attribute Specifications dialog: Default name for associated attrib.</td>
</tr>
<tr>
<td>genOptDefaultConstructorScope</td>
<td>Object Lifetimes dialog: Default Constructor Visibility.</td>
</tr>
<tr>
<td>genOptDefaultCopyConstructorScope</td>
<td>Object Lifetimes dialog: Default Copy Constructor Visibility.</td>
</tr>
<tr>
<td>genOptDefaultDatabase</td>
<td>Code Editors dialog: Default Database.</td>
</tr>
<tr>
<td>genOptDefaultDestructorScope</td>
<td>Object Lifetimes dialog: Default Destructor Constructor Visibility.</td>
</tr>
<tr>
<td>genOptGenConstructor</td>
<td>Object Lifetimes dialog: Generate Constructor checkbox.</td>
</tr>
<tr>
<td>genOptGenConstructorInline</td>
<td>Object Lifetimes dialog: Constructor Inline checkbox.</td>
</tr>
<tr>
<td>genOptGenCopyConstructor</td>
<td>Object Lifetimes dialog: Generate Copy Constructor checkbox.</td>
</tr>
<tr>
<td>genOptGenCopyConstructorInline</td>
<td>Object Lifetimes dialog: Copy Constructor Inline checkbox.</td>
</tr>
<tr>
<td>genOptGenDestructor</td>
<td>Object Lifetimes dialog: Generate Destructor checkbox.</td>
</tr>
<tr>
<td>genOptGenDestructorInline</td>
<td>Object Lifetimes dialog: Destructor Inline checkbox.</td>
</tr>
<tr>
<td>genOptGenDestructorVirtual</td>
<td>Object Lifetimes dialog: Virtual Destructor checkbox.</td>
</tr>
<tr>
<td>genOptGenPrefixBoolProperties</td>
<td>Source Code Engineering dialog: Use is prefix for boolean property Get().</td>
</tr>
<tr>
<td>genOptGenRoleNames</td>
<td>Source Code Engineering dialog: Autogenerate role names when creating code.</td>
</tr>
<tr>
<td>genOptGenUnspecAssocDir</td>
<td>Source Code Engineering dialog: Do not generate members where Association direction is unspecified checkbox.</td>
</tr>
<tr>
<td>genOptJavaDefaultAttributeType</td>
<td>Java Specifications dialog: Default attribute type.</td>
</tr>
<tr>
<td>genOptPHPGetPrefix</td>
<td>PHP Specifications dialog: Get Prefix.</td>
</tr>
<tr>
<td>genOptPHPSetPrefix</td>
<td>PHP Specifications dialog: Set Prefix.</td>
</tr>
<tr>
<td>genOptPHPSourceExtension</td>
<td>PHP Specifications dialog: Default file extension.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>genOptVBMultiUse</td>
<td>VB Specifications dialog: Multiuse checkbox.</td>
</tr>
<tr>
<td>genOptVBPersistable</td>
<td>VB Specifications dialog: Persitible checkbox.</td>
</tr>
<tr>
<td>genOptVBDataBindingBehavior</td>
<td>VB Specifications dialog: Data binding behavior checkbox.</td>
</tr>
<tr>
<td>genOptVBDataSourceBehavior</td>
<td>VB Specifications dialog: Data source behavior checkbox.</td>
</tr>
<tr>
<td>genOptVBGlobal</td>
<td>VB Specifications dialog: Global namespace checkbox.</td>
</tr>
<tr>
<td>genOptVBCreatable</td>
<td>VB Specifications dialog: Creatable checkbox.</td>
</tr>
<tr>
<td>genOptVBEExposed</td>
<td>VB Specifications dialog: Exposed checkbox.</td>
</tr>
<tr>
<td>genOptVBVersion</td>
<td>VB Specifications dialog: Default Version</td>
</tr>
<tr>
<td>genOptWrapComment</td>
<td>Source Code Engineering dialog: Wrap length for comment lines.</td>
</tr>
<tr>
<td>importClassName</td>
<td>The name of the Class being imported.</td>
</tr>
<tr>
<td>importFileName</td>
<td>The filename of the Class being imported.</td>
</tr>
<tr>
<td>importFilePath</td>
<td>The full path of the Class being imported.</td>
</tr>
<tr>
<td>importFromAggregation</td>
<td>T if the Class has an Aggregation connector to a Class in this file, F otherwise.</td>
</tr>
<tr>
<td>importFromAssociation</td>
<td>T if the Class has an Association connector to a Class in this file, F otherwise.</td>
</tr>
<tr>
<td>importFromAtt</td>
<td>T if an attribute of a Class in the current file is of the type of this Class, F otherwise.</td>
</tr>
<tr>
<td>importFromDependency</td>
<td>T if the Class has a Dependency connector to a Class in this file, F otherwise.</td>
</tr>
<tr>
<td>importFromGeneralization</td>
<td>T if the Class has a Generalization connector to a Class in this file, F otherwise.</td>
</tr>
<tr>
<td>importFromMeth</td>
<td>T if a method return type of a Class in the current file is the type of this Class, F otherwise.</td>
</tr>
<tr>
<td>importFromParam</td>
<td>T if an method parameter of a Class in the current file is of the type of this Class, F otherwise.</td>
</tr>
<tr>
<td>importFromRealization</td>
<td>T if the Class has a Realization connector to a Class in this file, F otherwise.</td>
</tr>
<tr>
<td>importInFile</td>
<td>T if the Class is in the current file, F otherwise.</td>
</tr>
<tr>
<td>importPackagePath</td>
<td>The package path with a <code>.</code> separator of the Class being imported.</td>
</tr>
<tr>
<td>ImportRelativeFilePath</td>
<td>The relative file path of the Class being imported from the file path of the file being generated.</td>
</tr>
<tr>
<td>linkAttAccess</td>
<td>Association Properties Target Role dialog: Access.</td>
</tr>
<tr>
<td>linkAttCollectionClass</td>
<td>The collection appropriate for the linked attribute in scope.</td>
</tr>
<tr>
<td>linkAttContainment</td>
<td>Association Properties Target Role dialog: Containment.</td>
</tr>
<tr>
<td>linkAttName</td>
<td>Association Properties dialog: Target.</td>
</tr>
<tr>
<td>linkAttNotes</td>
<td>Association Properties Target Role dialog: Role Notes.</td>
</tr>
<tr>
<td>linkAttQualifiedName</td>
<td>The Association target qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited).</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>linkAttRole</td>
<td>Association Properties Target Role dialog: Role.</td>
</tr>
<tr>
<td>linkAttStereotype</td>
<td>Association Properties Target Role dialog: Stereotype.</td>
</tr>
<tr>
<td>linkAttTargetScope</td>
<td>Association Properties Target Role dialog: Target Scope.</td>
</tr>
<tr>
<td>linkCard</td>
<td>Link Properties Target Role dialog: Multiplicity.</td>
</tr>
<tr>
<td>linkGUID</td>
<td>The unique GUID for the current connector.</td>
</tr>
<tr>
<td>linkParentName</td>
<td>Generalization Properties dialog: Target.</td>
</tr>
<tr>
<td>linkParentQualName</td>
<td>The Generalization target qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited).</td>
</tr>
<tr>
<td>linkStereotype</td>
<td>The stereotype of the current connector.</td>
</tr>
<tr>
<td>linkVirtualInheritance</td>
<td>Generalization Properties dialog: Virtual Inheritance.</td>
</tr>
<tr>
<td>opAbstract</td>
<td>Operation dialog: Virtual checkbox.</td>
</tr>
<tr>
<td>opAlias</td>
<td>Operation dialog: Alias.</td>
</tr>
<tr>
<td>opBehavior</td>
<td>Operation Behavior dialog: Behavior.</td>
</tr>
<tr>
<td>opConcurrency</td>
<td>Operation dialog: Concurrency.</td>
</tr>
<tr>
<td>opConst</td>
<td>Operation dialog: Const checkbox.</td>
</tr>
<tr>
<td>opGUID</td>
<td>The unique GUID for the current operation.</td>
</tr>
<tr>
<td>opImplMacros</td>
<td>A space separated list of macros defined in the implementation of this operation.</td>
</tr>
<tr>
<td>opIsQuery</td>
<td>Operation dialog: IsQuery checkbox.</td>
</tr>
<tr>
<td>opMacros</td>
<td>A space separated list of macros defined in the declaration for this operation.</td>
</tr>
<tr>
<td>opName</td>
<td>Operation dialog: Name.</td>
</tr>
<tr>
<td>opNotes</td>
<td>Operation dialog: Notes.</td>
</tr>
<tr>
<td>opPure</td>
<td>Operation dialog: Pure checkbox.</td>
</tr>
<tr>
<td>opReturnArray</td>
<td>Operation dialog: Return Array checkbox.</td>
</tr>
<tr>
<td>opReturnClassifierGUID</td>
<td>The unique GUID for the classifier of the current operation.</td>
</tr>
<tr>
<td>opReturnQualType</td>
<td>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.</td>
</tr>
<tr>
<td>opReturnType</td>
<td>Operation dialog: Return Type.</td>
</tr>
<tr>
<td>opScope</td>
<td>Operation dialog: Scope.</td>
</tr>
<tr>
<td>opStatic</td>
<td>Operation dialog: Static checkbox.</td>
</tr>
<tr>
<td>Operation dialog: Stereotype</td>
<td>Operation dialog: Stereotype.</td>
</tr>
<tr>
<td>opSynchronized</td>
<td>Operation dialog: Synchronized checkbox.</td>
</tr>
<tr>
<td>packageAbstract</td>
<td>Package dialog: Abstract.</td>
</tr>
<tr>
<td>packageAlias</td>
<td>Package dialog: Alias.</td>
</tr>
<tr>
<td>packageAuthor</td>
<td>Package dialog: Author.</td>
</tr>
<tr>
<td>packageComplexity</td>
<td>Package dialog: Complexity.</td>
</tr>
<tr>
<td>packageGUID</td>
<td>The unique GUID for the current package.</td>
</tr>
<tr>
<td>packageKeywords</td>
<td>Package dialog: Keywords.</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>packageLanguage</td>
<td>Package dialog: Language.</td>
</tr>
<tr>
<td>packagePath</td>
<td>The string representing the hierarchy of packages, for the Class in scope. Each package name is separated by a dot (.) .</td>
</tr>
<tr>
<td>packagePhase</td>
<td>Package dialog: Phase.</td>
</tr>
<tr>
<td>packageScope</td>
<td>Package dialog: Scope.</td>
</tr>
<tr>
<td>packageStatus</td>
<td>Package dialog: Status.</td>
</tr>
<tr>
<td>packageStereotype</td>
<td>Package dialog: Stereotype.</td>
</tr>
<tr>
<td>paramClassifierGUID</td>
<td>The unique GUID for the classifier of the current parameter.</td>
</tr>
<tr>
<td>paramDefault</td>
<td>Operation Parameters dialog: Default.</td>
</tr>
<tr>
<td>paramFixed</td>
<td>Operation Parameters dialog: Fixed checkbox.</td>
</tr>
<tr>
<td>paramGUID</td>
<td>The unique GUID for the current parameter.</td>
</tr>
<tr>
<td>paramsIsEnum</td>
<td>True, if the parameter uses the enum keyword (C++).</td>
</tr>
<tr>
<td>paramKind</td>
<td>Operation Parameters dialog: Kind.</td>
</tr>
<tr>
<td>paramName</td>
<td>Operation Parameters dialog: Name.</td>
</tr>
<tr>
<td>paramNotes</td>
<td>Operation Parameters dialog: Notes.</td>
</tr>
<tr>
<td>paramQualType</td>
<td>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.</td>
</tr>
<tr>
<td>paramType</td>
<td>Operation Parameters dialog: Type.</td>
</tr>
</tbody>
</table>

Field substitution macros can 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 enables 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> can be a combination of quoted strings and the keyword value; where the value is used, it is replaced with the macro’s value in the output.
Examples:
- %classAbstract=="T" ? "pure" : ""
- %opStereotype=="operator" ? "operator" : ""
- %paramDefault != "" ? " = " value : ""

The above three examples 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, the output from each of them would normally be:

= 10

Note:
In a conditional substitution macro, any white space following `<macroName>` is ignored. If white space is required in the output, it should be included within the quoted substitution strings.

16.5.1.2.3 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>"%

`<macroName>` can be one of:
- attTag
- classTag
- opTag
- packageTag
- paramTag
- connectorTag
- connectorSourceTag
- connectorDestTag
- linkTag
- linkAttTag

This corresponds to the tags for attributes, Classes, operations, packages, parameters, connectors with both ends and connectors 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> ]%

© 1998-2008 Sparx Systems Pty Ltd
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>` can 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.

### 16.5.1.2.4 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 results 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, which can be passed to the 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>)`.

**Note:**

Function macros are named according to the All-Caps style, as in: `%CONVERT_SCOPE(opScope)%`

#### CONVERT_SCOPE(<umlScope>)

For use with supported languages. 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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Package</th>
<th>Public</th>
<th>Private</th>
<th>Protected</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>public</td>
<td>public</td>
<td>private</td>
<td>protected</td>
</tr>
<tr>
<td>C#</td>
<td>internal</td>
<td>public</td>
<td>private</td>
<td>protected</td>
</tr>
<tr>
<td>Delphi</td>
<td>protected</td>
<td>public</td>
<td>private</td>
<td>protected</td>
</tr>
<tr>
<td>Java</td>
<td>public</td>
<td>public</td>
<td>private</td>
<td>protected</td>
</tr>
<tr>
<td>PHP</td>
<td>public</td>
<td>public</td>
<td>private</td>
<td>protected</td>
</tr>
<tr>
<td>VB</td>
<td>Protected</td>
<td>Public</td>
<td>Private</td>
<td>Protected</td>
</tr>
<tr>
<td>VB .Net</td>
<td>Friend</td>
<td>Public</td>
<td>Private</td>
<td>Protected</td>
</tr>
</tbody>
</table>

#### 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 Enterprise Architect Add-In function, which can 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 can be specified via parameters <prm_1> to <prm_n>. For example:

```delphi
$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 that contains any additional parameters from the `EXEC_ADD_IN` call. Return type should be `Variant`. For example:

```delphi
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**()

Returns 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>)

Substring of <src> starting at <start> and including <count> characters. Where <count> is omitted the rest of the string is included.

**PI**(<option>, <value>, ...)

Sets the PI for the current template to <value>. <option> controls when the new PI takes effect. Valid values are:

- `I`, Immediate: the new PI is generated before the next non-empty template line
- `N`, Next: the new PI is generated after the next non-empty template line.

Multiple pairs of options are allowed in one call. For more details, see the description of PI.

**REMOVE_DUPLICATES**(<source>, <separator>)

Where <source> is a <separator> separated list; this removes 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. Enables 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. For example:

```
$behavior = %WRAP_COMMENT(opBehavior, "40", "  ", "/")%
```

**Note:**
<wrap_length> must still be passed 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.

### 16.5.1.2.5 Control Macros

Control macros are used to control the processing and formatting of the templates. The basic types of control macro 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, which takes effect from the next non-empty line
- A PI function macro that enables setting PI to a variable and adds the ability to set the PI that is
The PI macro for formatting new lines in the output
The synchronization macros.

In general, control macros are named according to Camel casing.

List
The list macro 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> can be one of the following template names:

- Namespace
- Class
- ClassImpl
- Attribute
- InnerClass
- InnerClassImpl
- Operation
- OperationImpl
- Parameter
- ClassBase
- ClassInterface
- Custom Template (custom templates enable you to define your own templates; for more information see Custom Templates)

<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 the list items. 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 its items with a single new line and indent them two spaces respectively. If the Class in scope had any stereotyped attributes, they would be generated using the appropriately specialized 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 also generates attributes derived from associations by executing the appropriate LinkedAttribute template
- If the ClassBase template is used as an argument to the list macro, this also generates 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 also generates 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 Enterprise Architect 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 \(<\text{operator}>\) can be one of:

- `==`
- `!=`

and \(<\text{test}>\) can be one of:

- a string literal, enclosed within double quotation marks
- a direct substitution macro, without the enclosing percent signs
- a variable reference.

Branches can be nested, and multiple conditions can 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 elemType == "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 (\(\backslash n\), \(\backslash t\)) 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"%
%PI=" "%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%opReturnValue=="T" ? "[T]" : ""
%opStereotype=="operator" ? "operator" : ""
%opName%(%list="Parameter" @separator="", %)
```

Default PI is \(\backslash n\), so any attributes would be on their own line

Blank lines have no effect on the output

Set the PI, so keywords are separated by a space

Any keyword that does not apply, i.e. the macro produces an empty result, does not result in a space

Only one space is generated for this line

The final line in the template does not generate a
In the above example macros for the various keywords are to be arranged vertically for readability. In the output, however, each relevant keyword is to be separated by a single space. This is achieved by the line:

```
%PI=" "%
```

Notice how you do not 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>` can be a literal string enclosed by double quotes.

The following points apply to the `PI` macro:

- The value of the PI is not accessed explicitly
- Only template lines that 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 Enterprise Architect 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:

```
%%
```

where `<name>` can be one of the macros listed below and `<value>` is a literal string enclosed by double quotes.

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>synchNewClassNotesSpace</code></td>
<td>Space to append to a new Class note. Default value: <code>\n</code>.</td>
</tr>
<tr>
<td><code>synchNewAttributeNotesSpace</code></td>
<td>Space to append to a new attribute note. Default value: <code>\n</code>.</td>
</tr>
<tr>
<td><code>synchNewOperationNotesSpace</code></td>
<td>Space to append to a new operation note. Default value: <code>\n</code>.</td>
</tr>
<tr>
<td><code>synchNewOperationBodySpace</code></td>
<td>Space to append to a new operation body. Default value: <code>\t</code>.</td>
</tr>
<tr>
<td><code>synchNamespaceBodyIndent</code></td>
<td>Indent applied to Classes within non-global namespaces. Default value: <code>\t</code>.</td>
</tr>
</tbody>
</table>

### 16.5.1.3 Variables

Template variables provide a convenient way of storing and retrieving data within a template. This section explains how variables are defined and referenced.

#### Variable Definitions

Variable definitions take the basic form:

```
$<name> = <value>
```

where `<name>` can be any alpha-numeric sequence and `<value>` is derived from a macro or another variable.

A simple example definition would be:

```
$foo = %className%
```

Variables can be defined, using values from:

- Substitution, function or list macros
- String literals, enclosed within double quotation marks
- Variable references.

#### Definition Rules

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 have to be declared, prior to being defined.
- Variables must be defined using either the assignment operator (=), or the addition-assignment operator (+=).
- Multiple terms can 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 + ">
```

Variable References

Variable values can be retrieved by using a reference of the form:

```
$<name>
```

where <name> can be a previously defined variable.

Variable references can 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 1

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%
%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 2

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 3**

Substituting variable values into the output

\[ \textdollar\text{bases} = \%\text{classInherits}\% \]

Class \%\text{className}\%$\textdollar\text{bases}$

Store the result of the `ClassInherits` template in $\textdollar\text{bases}$. Now output the value of $\textdollar\text{bases}$ after the Class name.
16.5.2 The Code Template Editor in SDK

The Code Template Editor window is introduced in the Enterprise Architect User Guide. The following topics describe how you use it to create custom templates:

- Custom Templates
- Override Default Templates
- Add New Stereotyped Templates
- Create Templates For Custom Languages

16.5.2.1 Custom Templates

Custom templates enable you to generate an element in many different ways. Enterprise Architect enables 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 might list all of your parameters and their notes in your method notes.

Note:

User-defined Code Templates can be imported and exported as Reference Data (see the Import and Export Reference Data topic in the Enterprise Architect User Guide). The templates defined for each language appear in a list of tables with the language name having the suffix _Code_Templates.

To create a new custom template, follow the steps below:

1. Select the Settings | Code Generation Templates menu option, or press [Ctrl]+[Shift]+[P]. The Code Templates Editor tab opens.
2. In the Language field, click on the drop-down arrow and select the appropriate language.
3. Click on the Add New Custom Template button. The Create New Custom Template dialog displays.

<table>
<thead>
<tr>
<th>Template Type</th>
<th>OK</th>
<th>Cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. In the Template Type field, click on the drop-down arrow and select the appropriate element. The elements currently supported are:
   - Attribute
   - Class
   - Operation
   - Parameter
5. In the Template Name field, type an appropriate name, then click on the OK button.
6. On the Code Templates Editor tab, the new template displays in the Templates list with the value Yes in the Modified field. The template is called <Element Type>__<Template Name> - for example, Parameter__JavaDoc.
7. Select the appropriate template from the Templates list and edit the contents in the Template field to meet your requirements.
8. Click on 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.

### 16.5.2.2 Override Default Templates

Enterprise Architect has a set of built-in or default code generation templates. The Code Templates Editor enables you to modify these default templates, hence customizing the way in which Enterprise Architect generates code. You can choose to modify any or all of the base templates to achieve your required coding style.

Any templates that you have overridden are stored in the .EAP file. When generating code, Enterprise Architect first checks whether a template has been modified and if so, uses that template. Otherwise the appropriate default template is used.

**Procedure**

To override a default code generation template, follow the steps below.

1. Select the **Configuration | Code Generation Templates** menu option. The Code Templates Editor displays.
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 can select one of these from the **Stereotype Overrides** list.
5. In the **Code Templates Editor**, make the required modifications.
6. Click on the **Save** button. This stores the modified version of the template to the .EAP file. The template is marked as modified.

When generating code, Enterprise Architect now uses the overridden template, instead of the default template.
16.5.2.3 Add New Stereotyped Templates

Sometimes it is useful to define a specific code generation template for use with elements of a given stereotype. This enables different code to be generated for elements, depending on their stereotype. Enterprise Architect provides some default templates, which have been specialized for commonly used stereotypes in supported languages. For example, the Operation Body template for C# has been specialized for the property stereotype, so that it automatically generates its constituent get and set methods. Users can override the default stereotyped templates as described in the previous topic. Additionally users can 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 the Configuration | Code Generation Templates menu option 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. Click on the Add New Stereotyped Override button. The New Template Override dialog displays.
5. Select the required Feature and/or Class stereotype and click on the OK button.
6. The new stereotyped template override displays in Stereotype Overrides list, marked as modified.
7. Make the required modifications in the Code Templates Editor.
8. Click on the Save button. This stores the new stereotyped template in the .EAP file.

Enterprise Architect can now use the stereotyped template, when generating 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.

16.5.2.4 Create Custom Language Template

Enterprise Architect can forward generate code for languages that it does not specifically support, if the appropriate code generation templates are defined for that language. This topic 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:
   - Select the Settings | Code Datatypes menu option. The Programming Languages Datatypes dialog displays.
   - In the Product Name field type the name of the new language, and in the Datatype field type a datatype (one is enough to declare that the new language exists). See the Data Types topic in the Enterprise Architect User Guide for more details.
2. Select the Settings | Code Generation Templates menu option. The Code Templates Editor view displays.
3. In the Language field, click on the drop-down arrow and select the custom language.
4. From the Templates list, select one of the base templates.
5. Define the template using the Code Templates Editor.
6. Click on the Save button. 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 see the *Import Section*, *Namespace* and *Class* templates.
16.6 Enterprise Architect Add-In Model

Introduction
Add-Ins enable you to add functionality to Enterprise Architect. The Enterprise Architect Add-In model builds on the features provided by the Automation Interface to enable 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 can define Enterprise Architect menus and sub-menus
- Add-Ins receive notifications about various Enterprise Architect 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 Enterprise Architect environment
- Because a current version of Enterprise Architect is already running there is no requirement to start a second copy of Enterprise Architect via the automation interface
- Because the Add-In receives object handles associated with the currently running copy of Enterprise Architect, more information is available about the current user's activity; for example, which diagram objects are selected
- You are not required to do anything other than to install the Add-In to make it usable; that is, you do not have to configure Add-Ins to run on your systems.

Because Enterprise Architect is constantly evolving in response to customer requests, the Add-In interface is flexible:

- The Add-In interface does not have its own version, rather it is identified by the version of Enterprise Architect it first appeared in; for example, the current version of the Enterprise Architect Add-In interface is version 2.1.
- When creating your Add-In, you do not have to subscribe to a type-library.

Note:
From Enterprise Architect release 7.0 Add-Ins created before 2004 are no longer supported. If an Add-In subscribes to the Addn_Tmpl.tlb interface (2003 style), it will fail on load. In this event, contact the vendor or author of the Add-In and request an upgrade.

- Add-Ins do not have to implement methods that they never use.
- Add-Ins prompt users via context menus in the tree view and the diagram.
- Menu check and disable states can be controlled by the Add-In.

Add-Ins 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.

Create and Use Add-Ins
This topic covers the following information on Add-Ins:

- Add-In Tasks
- Add-In Events
- Broadcast Events
- Custom Views
- MDG Add-Ins
16.6.1 Add-In Tasks

This topic provides instructions on how to create, test, deploy and manage Add-Ins.

1. Create an Add-In
   - Define Menu Items
   - Respond to Menu Events
   - Handle Add-In Events
2. Deploy your Add-In
   - Potential Pitfalls
3. Manage Add-Ins
   - Register an Add-In (developed in-house or brought-in)
   - The Add-In Manager

16.6.1.1 Create Add-Ins

Before you start you must have an application development tool that is capable of creating ActiveX COM objects supporting the IDispatch interface, such as:

- Borland Delphi
- Microsoft Visual Basic

You should consider how to define menu items. To help with this, you could review some examples of Automation Interfaces (this web page provides examples of code used to create Add-Ins for Enterprise Architect).

Create an Add-In

An Enterprise Architect Add-In can be created in four steps:

1. Use a development tool to create an 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 have to define methods for events to respond to.
4. Add a registry key that identifies your Add-In to Enterprise Architect, as described in the Deploying Add-Ins topic.

16.6.1.1.1 Define Menu Items

Menu items are defined by responding to the GetMenuItems event.

The first time this event is called, MenuName is an empty string, representing the top-level menu. For a simple Add-In with just a single menu option you can return a string; for example:

```vbscript
Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant
    EA_GetMenuItems = "&Joe's Add-In"
End Function
```

To define sub-menus, prefix a parent menu with a dash. Parent and sub-items are defined as follows:

```vbscript
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.
        EA_GetMenuItems = Array("&Diagram", ",&Treeview")
    Case Else
        MsgBox "Invalid Menu", vbCritical
    End Select
End Function
```

In this example, "Diagram" and "Treeview" compose the "Joe's Add-In" sub-menu.
Similarly, you can define further sub-items:

```vba
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 "&Properties"
            EA_GetMenuItems = Array("&Diagram", "&TreeView")
        Case "-&Diagram"
            EA_GetMenuItems = "&Properties"
        Case Else
            MsgBox "Invalid Menu", vbCritical
    End Select
End Function
```

To enable or disable menu options by default, you can use this method to show particular items to the user:

```vba
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 If
End Select
End Sub
```

### 16.6.1.1.2 Deploy Add-Ins

To deploy Add-Ins to users' sites, follow the steps below:

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 using the `RegSvr32` command from the command prompt as shown in the example below.

![Command Prompt](image)

3. Place a new entry into the registry so that Enterprise Architect recognizes the presence of your Add-In by using the registry editor (`regedit`).
4. Add a new key value `EAAddins` under the location: `HKEY_CURRENT_USER\Software\Sparx Systems`

![Registry Keys](image)

5. Add a new key under this key with the project name.
6. Specify the default value by modifying the default value of the key.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>DWORD</td>
<td>(value not set)</td>
</tr>
</tbody>
</table>

7. Enter the value of the key by entering the \[project name\].\[class name\]; e.g. \EaRequirements.Requirements\ where \EaRequirements\ is the project name as shown in the example below.

Value name:

Value data:

\EaRequirements.Requirements\ | OK | Cancel |

16.6.1.3 Tricks and Traps

Visual Basic 5/6 Users Note

Visual Basic 5/6 users should note that the version number of the Enterprise Architect interface is stored in the VB project file in a form similar to the following:

Reference~\^\(\{64FB2BF4-9EFA-11D2-8307-C45860000000\}'\#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 Enterprise Architect to another, open the VB 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 Enterprise Architect Object model.

Add-In Fails to Load

From Enterprise Architect release 7.0, Add-Ins created before 2004 are no longer supported. If an Add-In subscribes to the Addn_Tmpl.tlb interface (2003 style), it will fail on load. In this event, contact the vendor or author of the Add-In and request an upgrade.
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:

- Enterprise Architect 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 can result in the second event querying objects that have no relationship with the current state of Enterprise Architect.
- When you close Enterprise Architect, all Add-Ins are asked to shut down. If there are any external automation clients Enterprise Architect must stay active, in which case all the Add-Ins are reloaded, losing all the data.
- Enterprise Architect acting as an automation client does 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.

Enterprise Architect Not Closing

.Net Specific Issues

Automation checks the use of objects and won't enable any of them to be destroyed until they are no longer being used.

As noted in the Automation Interface topic, if your automation controller was written using the .NET framework, Enterprise Architect does 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:

```csharp
GC.Collect();
GC.WaitForPendingFinalizers();
```

Additionally, because automation clients hook into Enterprise Architect, which creates Add-Ins which in turn hook back into Enterprise Architect, it is possible to get into a deadlock situation where Enterprise Architect and the Add-Ins won't let go of one another and keep each other active. An Add-In might retain hooks into Enterprise Architect because:

- It keeps a private reference to an Enterprise Architect object (see Holding State Information above), or
- 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 Enterprise Architect in the Disconnect() event. See the Add-In Methods topic for details.

It is possible that your Automation client controls a running instance of Enterprise Architect where the Add-Ins have not complied with the rule above. In this case you could call Repository.Exit() to terminate Enterprise Architect.

Miscellaneous

In developing Add-Ins using the .Net framework you must 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 might have to do that manually before Enterprise Architect recognizes the Add-In.

You can use your private Add-In key (as required for Add-In deployment) to store configuration information pertinent to your Add-In.

Concurrent calls

In Enterprise Architect releases up to release 7.0, there is a possibility that Enterprise Architect could call two Add-In methods concurrently if the Add-In calls:

- A message box
- A modal dialog
- VB DoEvents, .NET Application DoEvents or the equivalent in other languages.

In such cases, Enterprise Architect could initiate a second Add-In method before the first returns (re-entrancy).
In release 7.0. and subsequent releases, Enterprise Architect cannot make such concurrent calls.

If developing Add-Ins, ensure that the Add-In users are running Enterprise Architect release 7.0 or a later release to avoid any risk of concurrent method calls.
16.6.2 The Add-In Manager

You can use the Add-In Manager to view what Add-Ins are available and to disable those not to be used. Access the Add-In Manager dialog by selecting the Add-Ins | Manage Add-Ins menu option.

<table>
<thead>
<tr>
<th>Available Add-Ins</th>
<th>Status</th>
<th>Load on Startup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>Disabled</td>
<td></td>
</tr>
<tr>
<td>SearchAddin</td>
<td>Enabled</td>
<td>✓</td>
</tr>
</tbody>
</table>

To enable an Add-In for use, select the Load on Startup check box. To disable an Add-in, deselect the checkbox.

**Note:**
Enterprise Architect must be restarted for changes to take effect.
16.6.3 Add-In Search

Enterprise Architect enables Add-Ins to integrate with the Model Search (as described in the Enterprise Architect User Guide). Searches can be defined that execute a method within your Add-In and display your results in an integrated way.

The method that runs the search must be structured in the following way:

```
variant <method name> (Rep as Repository, SearchText as String, XMLResults as String)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rep</td>
<td>The currently open repository.</td>
</tr>
<tr>
<td>SearchText</td>
<td>An optional field that you can fill in through the Model Search.</td>
</tr>
<tr>
<td>XMLResults</td>
<td>At completion of the method, this should contain the results for the search. The results should be an XML String that conforms to the Search Data Format.</td>
</tr>
</tbody>
</table>

**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 provide special functionality.

**CLASSTYPE**

Returning a field of CLASSTYPE, containing the Object_Type value from the t_object table, displays the appropriate icon in the column you place the field.

**CLASSGUID**

Returning a field of CLASSGUID, containing an ea_guid value, enables the Model Search to track the object in the Project Browser and open the Properties window for the element by double-clicking in the Model Search.

16.6.3.1 XML Format (Search Data)

The XML below provides the format for the SearchData parameter of the RunModel method. See the Repository topic for more information.

```xml
<ReportViewData>
  <Row>
    <Field name=""/>
    <Field name=""/>
    <Field name=""/>
    <Field name=""/>
  </Row>
  <Row>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
  </Row>
  <Row>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
    <Field name="" value=""/>
  </Row>
</ReportViewData>
```

16.6.4 Add-In Events

All Enterprise Architect Add-Ins can choose to respond to the following general Add-In events:

- EA_Connect
- EA_Disconnect
- EA_GetMenuItems
- EA_MenuClick
- EA_GetMenuState
- EA_ShowHelp
- EA_OnOutputItemClicked
- EA_OnOutputItemDoubleClicked

16.6.4.1 EA_Connect

Details

EA_Connect events enable Add-Ins to identify their type and to respond to Enterprise Architect start up. This event occurs when Enterprise Architect first loads your Add-In. Enterprise Architect 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.

Also look at EA_Disconnect.

Syntax

Function EA_Connect(Repository As EA.Repository) As String

The EA_Connect function syntax has the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

String identifying a specialized type of Add-In:

<table>
<thead>
<tr>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;MDG&quot;</td>
<td>MDG Add-Ins receive MDG Events and extra menu options.</td>
</tr>
<tr>
<td>****</td>
<td>None-specialized Add-In.</td>
</tr>
</tbody>
</table>

16.6.4.2 EA_Disconnect

Details

The EA_Disconnect event enables the Add-In to respond to user requests to disconnect the model branch from an external project.

This function is called when the Enterprise Architect closes. If you have stored references to Enterprise Architect 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();

Also look at EA_Connect.
Syntax

Sub EA_Disconnect()

Return Value

None.

16.6.4.3 EA_GetMenuItems

Details

The \texttt{EA\_GetMenuItems} event enables the Add-In to provide the Enterprise Architect user interface with additional Add-In menu options 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 option.

This event is raised just before Enterprise Architect has to show particular menu options to the user, and its use is described in the \textit{Define Menu Items} topic. Also look at:

- \texttt{EA\_MenuClick}
- \texttt{EA\_GetMenuState}

Syntax

Function \texttt{EA\_GetMenuItems(Repository As EA\_Repository, MenuLocation As String, MenuName As String) As Variant} 

The \texttt{EA\_GetMenuItems} function syntax has the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MenuLocation</td>
<td>String</td>
<td></td>
<td>String representing the part of the user interface that brought up the menu. Can be \textit{TreeView}, \textit{MainMenu} or \textit{Diagram}.</td>
</tr>
<tr>
<td>MenuName</td>
<td>String</td>
<td></td>
<td>The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu it is an empty string.</td>
</tr>
<tr>
<td>Repository</td>
<td>\texttt{EA_Repository}</td>
<td>IN</td>
<td>An \texttt{EA_Repository} object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

One of the following types:

- A string indicating the label for a single menu option.
- An array of strings indicating a multiple menu options.
- 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.

16.6.4.4 EA_GetMenuState

Details

The \texttt{EA\_GetMenuState} event enables 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 option, but not enable it for use.

This event is raised just before Enterprise Architect has to show particular menu options to the user. Its use is described in the \textit{Define Menu Items} topic.

Also look at \texttt{EA\_GetMenuItems}. 

Enterprise Architect User Guide
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_GetMenuState function syntax has the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsChecked</td>
<td>Boolean</td>
<td></td>
<td>Boolean. Set to True to check this particular menu option.</td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Boolean</td>
<td></td>
<td>Boolean. Set to False to disable this particular menu option.</td>
</tr>
<tr>
<td>ItemName</td>
<td>String</td>
<td></td>
<td>The name of the option actually clicked, e.g. Create a New Invoice.</td>
</tr>
<tr>
<td>MenuLocation</td>
<td>String</td>
<td></td>
<td>String representing the part of the user interface that brought up the menu. Can be TreeView, MainMenu or Diagram.</td>
</tr>
<tr>
<td>MenuName</td>
<td>String</td>
<td></td>
<td>The name of the parent menu for which sub-items must be defined. In the case of the top-level menu it is an empty string.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

None.

16.6.4.5 EA_MenuClick

Details

EA_MenuClick events are received by an Add-In in response to user selection of a menu option.

The event is raised when the user clicks on a particular menu option. When a user clicks on one of your non-parent menu options, your Add-In receives 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 Enterprise Architect data and UI elements using Repository methods. Also look at EA_GetMenuItems.

Syntax

Sub EA_MenuClick(Repository As EA.Repository, MenuLocation As String, MenuName as String, ItemName as String)

The EA_MenuClick function syntax has the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemName</td>
<td>String</td>
<td></td>
<td>The name of the option actually clicked, e.g. Create a New Invoice.</td>
</tr>
<tr>
<td>MenuName</td>
<td>String</td>
<td></td>
<td>The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu it is an empty string.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>
### Parameter | Type | Direction | Description
--- | --- | --- | ---
 |  |  |  | data and user interface status information.

#### Return Value
None.

**16.6.4.6 EA_OnOutputItemClicked**

**Details**

`EA_OnOutputItemClicked` events inform Add-Ins that the user has clicked on a list entry in the system tab or one of the user defined output tabs.

Usually an Add-In responds 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 Enterprise Architect - 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.

Also look at `EA_OnOutputItemDoubleClicked`.

**Syntax**

```vbnet
EA_OnOutputItemClicked(Repository As EA.Repository, TabName As String, LineText As String, ID As Long)
```

The `EA_OnOutputItemClicked` function syntax has the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| ID | Long | IN | The ID value specified in the original call to `Repository.WriteOutput()`.
| LineText | String | IN | The text that had been supplied as the String parameter in the original call to `Repository.WriteOutput()`.
| Repository | `EA.Repository` | IN | An `EA.Repository` object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
| TabName | String | IN | The name of the tab that the click occurred in. Usually this would have been created through `Repository.AddTab()`.

#### Return Value
None.

**16.6.4.7 EA_OnOutputItemDoubleClicked**

**Details**

`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.

Usually an Add-In responds 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 Enterprise Architect - 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.

Also look at `EA_OnOutputItemClicked`.

**Syntax**

```vbnet
EA_OnOutputItemDoubleClicked(Repository As EA.Repository, TabName As String, LineText As String, ID As Long)
```
16.6.4.8 **EA_ShowHelp**

**Details**

The `EA_ShowHelp` event enables 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.

This event is raised when the user presses `[F1]` on a menu option that is not a parent menu.

Also look at `EA_GetMenuItems`.

**Syntax**

```vbs
Sub EA_ShowHelp(Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String)
```

The `EA_ShowHelp` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemName</td>
<td>String</td>
<td></td>
<td>The name of the option actually clicked, e.g. <code>Create a New Invoice</code>.</td>
</tr>
<tr>
<td>MenuLocation</td>
<td>String</td>
<td></td>
<td>String representing the part of the user interface that brought up the menu. Can be <code>Treeview</code>, <code>MainMenu</code> or <code>Diagram</code>.</td>
</tr>
<tr>
<td>MenuName</td>
<td>String</td>
<td></td>
<td>The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu it is an empty string.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

None.
16.6.5 Broadcast Events

Overview
The following general Broadcast events 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 Enterprise Architect detects that the Add-In has the required interface, the event is dispatched on to the Add-In.

- **File Open Event**
- **File Close Event**
- **Pre-Deletion Events**
- **Pre-New Events**
- **Post-New Events**
- **Technology Events**
- **Context Item Events**
- **Transformation Events**
- **Compartment Events**
- **Model Validation Broadcasts**
- **Retrieve Model Template Event**
- **Initialize Technology Event**

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.

16.6.5.1 EA_FileOpen

Details
The `EA_FileOpen` event enables the Add-In to respond to a File Open event. When Enterprise Architect opens a new model file, this event is raised and passed to all Add-Ins implementing this method.

The event occurs when the model being viewed by the Enterprise Architect user changes, for whatever reason (through user interaction or Add-In activity).

Also look at `EA_FileClose`.

Syntax

```vba
Sub EA_FileOpen(Repository As EA.Repository)
```

The `EA_FileOpen` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value
None.

16.6.5.2 EA_FileClose

Details
The `EA_FileClose` event enables the Add-In to respond to a File Close event. When Enterprise Architect closes an opened Model file, this event is raised and passed to all Add-Ins implementing this method.

This event occurs when the model currently opened within Enterprise Architect is about to be closed (when another model is about to be opened or when Enterprise Architect is about to shutdown).

Also look at `EA_FileOpen`.
Syntax

Sub EA_FileClose(Repository As EA.Repository)

The `EA_FileClose` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the Enterprise Architect model about to be closed. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

None.

16.6.5.3 Pre-Deletion Events

Enterprise Architect Add-Ins can respond to requests to delete elements, connectors, diagrams, packages and technologies using the following broadcast events:

- `EA_OnPreDeleteElement`
- `EA_OnPreDeleteConnector`
- `EA_OnPreDeleteDiagram`
- `EA_OnPreDeletePackage`

16.6.5.3.1 EA_OnPreDeleteElement

Details

`EA_OnPreDeleteElement` notifies Add-Ins that an element is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the element.

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 can disable 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>EA.EventProperties</td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> Objects for the element to be deleted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>ElementID</code>: A long value corresponding to <code>Element.ElementID</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

Return `True` to enable deletion of the element from the model. Return `False` to disable deletion of the element.

16.6.5.3.2 EA_OnPreDeleteConnector

Details

`EA_OnPreDeleteConnector` notifies Add-Ins that a connector is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the connector.

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 can disable 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA.EventProperties</code></td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> Objects for the connector to be deleted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <code>ConnectorID</code>: A long value corresponding to <code>Connector.ConnectorID</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return `True` to enable deletion of the connector from the model. Return `False` to disable deletion of the connector.

**16.6.5.3.3 EA_OnPreDeleteDiagram**

**Details**

`EA_OnPreDeleteDiagram` notifies Add-Ins that a diagram is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the diagram.

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 can disable 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA.EventProperties</code></td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> Objects for the connector to be deleted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <code>DiagramID</code>: A long value corresponding to <code>Diagram.DiagramID</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return `True` to enable deletion of the diagram from the model. Return `False` to disable deletion of the diagram.

**16.6.5.3.4 EA_OnPreDeletePackage**

**Details**

`EA_OnPreDeletePackage` notifies Add-Ins that a package is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the package.

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 can disable 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA_EventProperties</code></td>
<td>IN</td>
<td>Contains the following EventProperty Objects for the connector to be deleted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>· PackageID: A long value corresponding to <code>Package</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return `True` to enable deletion of the package from the model. Return `False` to disable deletion of the package.

### 16.6.5.4 Pre-New Events

Enterprise Architect Add-Ins can respond to requests to create new elements, connectors, objects, attributes, methods and packages using the following broadcast events:

- `EA_OnPreNewElement`
- `EA_OnPreNewConnector`
- `EA_OnPreNewDiagram`
- `EA_OnPreNewAttribute`
- `EA_OnPreNewMethod`
- `EA_OnPreNewPackage`

#### 16.6.5.4.1 EA_OnPreNewElement

**Details**

`EA_OnPreNewElement` notifies Add-Ins that a new element is about to be created on a diagram. It enables Add-Ins to permit or deny creation of the new element.

This event occurs when a user drags a new element from the Enterprise Architect UML Toolbox or Resources window onto a diagram. The notification is provided immediately before the element is created, so that the Add-In can disable addition of the element.

Also look at `EA_OnPostNewElement`.

**Syntax**

*Function EA_OnPreNewElement(Repository As EA.Repository, Info As EA.EventProperties) As Boolean*

The `EA_OnPreNewElement` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA_EventProperties</code></td>
<td>IN</td>
<td>Contains the following EventProperty Objects for the element to be created:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>· Type: A string value corresponding to <code>Element.Type</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>· Stereotype: A string value corresponding to <code>Element</code>.</td>
</tr>
</tbody>
</table>
## Parameter Types and Direction

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

### Return Value

Return **True** to enable addition of the new element to the model. Return **False** to disable addition of the new element.

### 16.6.5.4.2 EA_OnPreNewConnector

**Details**

EA_OnPreNewConnector notifies Add-Ins that a new connector is about to be created on a diagram. It enables Add-Ins to permit or deny creation of a new connector.

This event occurs when a user drags a new connector from the Enterprise Architect UML Toolbox or Resources window, onto a diagram. The notification is provided immediately before the connector is created, so that the Add-In can disable addition of the connector.

Also look at **EA_OnPostNewConnector**.

**Syntax**

*Function* EA_OnPreNewConnector(Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewConnector function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| Info      | EA.EventProperties | IN        | Contains the following EventProperty Objects for the connector to be created:
|           |                    |           | • 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
|           |                    |           | • DiagramID: A long value corresponding to Connector.DiagramID. |

| Repository | EA.Repository      | IN        | An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information. |

### Return Value

Return **True** to enable addition of the new connector to the model. Return **False** to disable addition of the new connector.

### 16.6.5.4.3 EA_OnPreNewDiagram

**Details**

EA_OnPreNewDiagram notifies Add-Ins that a new diagram object is about to be dropped on a diagram. It enables Add-Ins to permit or deny creation of the new object.
This event occurs when a user drags an object from the Enterprise Architect Project Browser or Resources window onto a diagram. The notification is provided immediately before the object is created, so that the Add-In can disable addition of the object.

Also look at [EA_OnPostNewDiagram](#).

### Syntax

**Function** `EA_OnPreNewDiagram(Repository As EA.Repository, Info As EA.EventProperties) As Boolean`

The `EA_OnPreNewDiagram` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| Info      | `EA.EventProperties` | IN        | Contains the following `EventProperty` Objects for the object to be created:
  - `Type`: A string value corresponding to `Object.Type`
  - `Stereotype`: A string value corresponding to `Object.Stereotype`
  - `ParentID`: A long value corresponding to `Object.ParentID`
  - `DiagramID`: A long value corresponding to the ID of the diagram to which the object is being added |
| Repository | `EA.Repository` | IN | An `EA.Repository` object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information |

### Return Value

Return `True` to enable addition of the object to the model. Return `False` to disable addition of the object.

### 16.6.5.4.4 EA_OnPreNewAttribute

**Details**

`EA_OnPreNewAttribute` notifies Add-Ins that a new attribute is about to be created on an element. It enables Add-Ins to permit or deny creation of the new attribute.

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 using the in-place editor on the diagram. The notification is provided immediately before the attribute is created, so that the Add-In can disable addition of the attribute.

Also look at [EA_OnPostNewAttribute](#).

### Syntax

**Function** `EA_OnPreNewAttribute(Repository As EA.Repository, Info As EA.EventProperties) As Boolean`

The `EA_OnPreNewAttribute` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| Info      | `EA.EventProperties` | IN        | Contains the following `EventProperty` Objects for the attribute to be created:
  - `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` |
| Repository | `EA.Repository` | IN | An `EA.Repository` object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information |
Return Value

Return **True** to enable addition of the new attribute to the model. Return **False** to disable addition of the new attribute.

### 16.6.5.4.5 EA_OnPreNewMethod

#### Details

**EA_OnPreNewMethod** notifies Add-Ins that a new method is about to be created on an element. It enables Add-Ins to permit or deny creation of the new method.

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 using the in-place editor on the diagram. The notification is provided immediately before the method is created, so that the Add-In can disable addition of the method.

Also look at **EA_OnPostNewMethod**.

#### Syntax

```plaintext
Function EA_OnPreNewMethod(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The **EA_OnPreNewMethod** function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| Info        | EA_EventProperties    | IN        | Contains the following EventProperty Objects for the method to be created:  
|             |                       |           | · 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. |
| Repository  | EA.Repository         | IN        | An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information. |

#### Return Value

Return **True** to enable addition of the new method to the model. Return **False** to disable addition of the new method.

### 16.6.5.4.6 EA_OnPreNewPackage

#### Details

**EA_OnPreNewPackage** notifies Add-Ins that a new package is about to be created in the model. It enables Add-Ins to permit or deny creation of the new package.

This event occurs when a user drags a new package from the Enterprise Architect UML Toolbox or Resources window onto a diagram, or by selecting the **New Package** icon from the Project Browser. The notification is provided immediately before the package is created, so that the Add-In can disable addition of the package.

Also look at **EA_OnPostNewPackage**.

#### Syntax

```plaintext
Function EA_OnPreNewPackage(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The **EA_OnPreNewPackage** function syntax contains the following elements:
### Return Value
Return `True` to enable addition of the new package to the model. Return `False` to disable addition of the new package.

#### 16.6.5.5 Post-New Events
Enterprise Architect Add-Ins can respond to the creation of new elements, connectors, objects, attributes, methods and packages using the following broadcast events:

- `EA_OnPostNewElement`
- `EA_OnPostNewConnector`
- `EA_OnPostNewDiagram`
- `EA_OnPostNewAttribute`
- `EA_OnPostNewMethod`
- `EA_OnPostNewPackage`

##### 16.6.5.5.1 EA_OnPostNewElement

**Details**

`EA_OnPostNewElement` notifies Add-Ins that a new element has been created on a diagram. It enables Add-Ins to modify the element upon creation.

This event occurs after a user has dragged a new element from the Enterprise Architect UML Toolbox or Resources window onto a diagram. The notification is provided immediately after the element is added to the model. Set `Repository.SuppressEADialogs` to `true` to suppress Enterprise Architect from showing its default dialogs.

Also look at `EA_OnPreNewElement`.

**Syntax**

```
Function EA_OnPostNewElement(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The `EA_OnPostNewElement` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>EA. EventProperties</td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> objects for the new element:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <code>ElementID</code>: A long value corresponding to <code>Element.ElementID</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>
Return Value
Return **True** if the element has been updated during this notification. Return **False** otherwise.

### 16.6.5.5.2 EA_OnPostNewConnector

**Details**

`EA_OnPostNewConnector` notifies Add-Ins that a new connector has been created on a diagram. It enables Add-Ins to modify the connector upon creation.

This event occurs after a user has dragged a new connector from the Enterprise Architect UML Toolbox or Resources window onto a diagram. The notification is provided immediately after the connector is added to the model. Set `Repository.SuppressEADialogs` to **true** to suppress Enterprise Architect from showing its default dialogs.

Also look at `EA_OnPreNewConnector`.

**Syntax**

```plaintext
Function EA_OnPostNewConnector(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The `EA_OnPostNewConnector` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>EA.EventProperties</td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> objects for the new connector:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>ConnectorID</strong>: A long value corresponding to <code>Connector.ConnectorID</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return **True** if the connector has been updated during this notification. Return **False** otherwise.

### 16.6.5.5.3 EA_OnPostNewDiagram

**Details**

`EA_OnPostNewDiagram` notifies Add-Ins that a new object has been created on a diagram. It enables Add-Ins to modify the object upon creation.

This event occurs after a user has dragged a new object from the Enterprise Architect Project Browser or Resources window onto a diagram. The notification is provided immediately after the object is added to the diagram. Set `Repository.SuppressEADialogs` to **true** to suppress Enterprise Architect from showing its default dialogs.

Also look at `EA_OnPreNewDiagram`.

**Syntax**

```plaintext
Function EA_OnPostNewDiagram(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The `EA_OnPostNewDiagram` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>EA.EventProperties</td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> objects for the new element:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>ObjectID</strong>: A long value corresponding to <code>Object.ObjectID</code>.</td>
</tr>
</tbody>
</table>
Parameter | Type | Direction | Description
--- | --- | --- | ---
Repository | EA.Repository | IN | An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

**Return Value**
Return **True** if the element has been updated during this notification. Return **False** otherwise.

### 16.6.5.5.4 EA_OnPostNewAttribute

**Details**
EA_OnPostNewAttribute notifies Add-Ins that a new attribute has been created on a diagram. It enables Add-Ins to modify the attribute upon creation.

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 using the in-place editor on the diagram. The notification is provided immediately after the attribute is created. Set Repository.SuppressEADialogs to **true** to suppress Enterprise Architect from showing its default dialogs.

Also look at **EA_OnPreNewAttribute**.

**Syntax**

Function EA_OnPostNewAttribute(Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The **EA_OnPostNewAttribute** function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| Info | EA.EventProperties | IN | Contains the following EventProperty objects for the new attribute:  
- AttributeID: A long value corresponding to Attribute.AttributeID. |
| Repository | EA.Repository | IN | An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information. |

**Return Value**
Return **True** if the attribute has been updated during this notification. Return **False** otherwise.

### 16.6.5.5.5 EA_OnPostNewMethod

**Details**
EA_OnPostNewMethod notifies Add-Ins that a new method has been created on a diagram. It enables Add-Ins to modify the method upon creation.

This event occurs when a user creates a new method on an element by either drag-dropping from the Project Browser, using the method's Properties dialog, or using the in-place editor on the diagram. The notification is provided immediately after the method is created. Set Repository.SuppressEADialogs to **true** to suppress Enterprise Architect from showing its default dialogs.

Also look at **EA_OnPreNewMethod**.

**Syntax**

Function EA_OnPostNewMethod(Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The **EA_OnPostNewMethod** function syntax contains the following elements:
### 16.6.5.5.6 EA_OnPostNewPackage

**Details**

`EA_OnPostNewPackage` notifies Add-Ins that a new package has been created on a diagram. It enables Add-Ins to modify the package upon creation.

This event occurs when a user drags a new package from the Enterprise Architect UML Toolbox or Resources window onto a diagram, or by selecting the **New Package** icon from the Project Browser. Set `Repository.SuppressEADialogs` to `true` to suppress Enterprise Architect from showing its default dialogs. Also look at `EA_OnPreNewPackage`.

**Syntax**

```vbnet
Function EA_OnPostNewPackage(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The `EA_OnPostNewPackage` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA_EventProperties</code></td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> objects for the new package:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <code>PackageID</code>: A long value corresponding to <code>Package.PackageID</code>.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return `True` if the package has been updated during this notification. Return `False` otherwise.

### 16.6.5.6 Technology Events

Enterprise Architect Add-Ins can respond to the following events associated with the use of MDG Technologies:

- `EA_OnPreDeleteTechnology`
- `EA_OnDeleteTechnology`
- `EA_OnImportTechnology`

#### 16.6.5.6.1 EA_OnPreDeleteTechnology

**Details**

`EA_OnPreDeleteTechnology` notifies Add-Ins that an MDG Technology resource is about to be deleted from the model. 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 can disable deletion of the MDG Technology.
Also look at `EA_OnImportTechnology` and `EA_OnDeleteTechnology`.

**Syntax**

```vbnet
Function EA_OnPreDeleteTechnology(Repository As EA.Repository, Info As EA.EventProperties) As Boolean
```

The `EA_OnPreDeleteTechnology` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA.EventProperties</code></td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> objects for the MDG Technology to be deleted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>· TechnologyID: A string value corresponding to the MDG Technology ID.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return `True` to enable deletion of the MDG Technology resource from the model. Return `False` to disable deletion of the MDG Technology resource.

**16.6.5.6.2 EA_OnDeleteTechnology**

**Details**

`EA_OnDeleteTechnology` notifies Add-Ins that an MDG Technology resource has been deleted from the model.

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 can catch this Add-In to disable certain functionality.

Also look at `EA_OnImportTechnology` and `EA_OnPreDeleteTechnology`.

**Syntax**

```vbnet
Sub EA_OnDeleteTechnology(Repository As EA.Repository, Info As EA.EventProperties)
```

The `EA_OnDeleteTechnology` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td><code>EA.EventProperties</code></td>
<td>IN</td>
<td>Contains the following <code>EventProperty</code> objects:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>· TechnologyID: A string value corresponding to the MDG Technology ID.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

**16.6.5.6.3 EA_OnImportTechnology**

**Details**

`EA_OnImportTechnology` notifies Add-Ins that you have imported an MDG Technology resource into the model.

This event occurs after you have imported an MDG Technology resource into the model. Add-Ins that require an MDG Technology resource to be loaded can catch this Add-In to enable certain functionality.
Also look at `EA_OnDeleteTechnology`.

### Syntax

**Sub EA_OnImportTechnology(Repository As EA.Repository, Info As EA.EventProperties)**

The `EA_OnImportTechnology` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>EA.EventProperties</td>
<td>IN</td>
<td>Contains the following EventProperty objects: <em>TechnologyID</em>: A string value corresponding to the MDG Technology ID.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

### Return Value

None.

#### 16.6.5.7 Context Item Events

Enterprise Architect Add-Ins can respond to the following events associated with changing context:

- `EA_OnContextItemChanged`
- `EA_OnContextItemDoubleClicked`
- `EA_OnNotifyContextItemModified`

#### 16.6.5.7.1 EA_OnContextItemChanged

### Details

`EA_OnContextItemChanged` notifies Add-Ins that a different item is now in context. This event occurs after a user has selected an item anywhere in the Enterprise Architect GUI. Add-Ins that require knowledge of the current item in context can subscribe to this broadcast function. If $ot = otRepository$, then this function behaves the same as `EA_FileOpen`. Also look at `EA_OnContextItemDoubleClicked` and `EA_OnNotifyContextItemModified`.

### Syntax

**Sub EA_OnContextItemChanged(Repository As EA.Repository, GUID As String, ot as EA.ObjectType)**

The `EA_OnContextItemChanged` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUID</td>
<td>String</td>
<td>IN</td>
<td>Contains the GUID of the new context item. This value corresponds to the following properties, depending on the value of the <code>ot</code> parameter: <em>ot (ObjectType)</em>: GUID value, <em>Element.ElementGUID</em></td>
</tr>
</tbody>
</table>
### Parameter Description Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ot</td>
<td><code>EA.ObjectType</code></td>
<td>IN</td>
<td>Specifies the type of the new context item.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

### Return Value

`None.`

#### 16.6.5.7.2 EA_OnContextItemDoubleClicked

**Details**

`EA_OnContextItemDoubleClicked` notifies Add-Ins that the user has double-clicked the item currently in context.

This event occurs when a user has double-clicked (or pressed `Enter`) on the item in context, either in a diagram or in the Project Browser. Add-Ins to handle events can subscribe to this broadcast function.

Also look at `EA_OnContextItemChanged` and `EA_OnNotifyContextItemModified`.

**Syntax**

```plaintext
Function EA_OnContextItemDoubleClicked(Repository As EA.Repository, GUID As String, ot as EA.ObjectType)
```

The `EA_OnContextItemDoubleClicked` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUID</td>
<td><code>String</code></td>
<td>IN</td>
<td>Contains the GUID of the new context item. This value corresponds to the following properties, depending on the value of the <code>ot</code> parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>ot (ObjectType)</code> - GUID value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>otElement</code> - <code>Element.ElementGUID</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>otPackage</code> - <code>Package.PackageGUID</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>otDiagram</code> - <code>Diagram.DiagramGUID</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>otAttribute</code> - <code>Attribute.AttributeGUID</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>otMethod</code> - <code>Method.MethodGUID</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>otConnector</code> - <code>Connector.ConnectorGUID</code></td>
</tr>
<tr>
<td><code>ot</code></td>
<td><code>EA.ObjectType</code></td>
<td>IN</td>
<td>Specifies the type of the new context item.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Return `True` to notify Enterprise Architect that the double-click event has been handled by an Add-In. Return `False` to enable Enterprise Architect to continue processing the event.

#### 16.6.5.7.3 EA_OnNotifyContextItemModified

**Details**

`EA_OnNotifyContextItemModified` notifies Add-Ins that the current context item has been modified.

This event occurs when a user has modified the context item. Add-Ins that require knowledge of when an item
has been modified can subscribe to this broadcast function. Also look at \texttt{EA\_OnContextItemChanged} and \texttt{EA\_OnContextItemDoubleClicked}.

### Syntax

\texttt{Sub EA\_OnNotifyContextItemModified(Repository As EA\_Repository, GUID As String, ot as EA\_ObjectType)}

The \texttt{EA\_OnNotifyContextItemModified} function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUID</td>
<td>String</td>
<td>IN</td>
<td>Contains the GUID of the new context item. This value corresponds to the following properties, depending on the value of the \texttt{ot} parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{ot (ObjectTyep)} - GUID value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{otElement} - Element.ElementGUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{otPackage} - Package.PackageGUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{otDiagram} - Diagram.DiagramGUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{otAttribute} - Attribute.AttributeGUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{otMethod} - Method.MethodGUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{otConnector} - Connector.ConnectorGUID</td>
</tr>
<tr>
<td>\texttt{ot}</td>
<td>EA_ObjectType</td>
<td>IN</td>
<td>Specifies the type of the new context item.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA_Repository</td>
<td>IN</td>
<td>An EA_Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

### Return Value
None.

#### 16.6.5.8 \texttt{EA\_OnPostTransform}

### Details

\texttt{EA\_OnPostTransform} notifies Add-Ins that an MDG transformation has taken place with the output in the specified target package.

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.

### Syntax

\texttt{Function EA\_OnPostTransform(Repository As EA\_Repository, Info As EA\_EventProperties) As Boolean}

The \texttt{EA\_OnPostTransform} function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>EA_EventProperties</td>
<td>IN</td>
<td>Contains the following EventProperty Objects for the transform performed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{Transform}: A string value corresponding to the name of the transform used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\texttt{PackageID}: A long value corresponding to Package. PackageID of the destination package.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA_Repository</td>
<td>IN</td>
<td>An EA_Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>
16.6.5.9 Compartment Events

Enterprise Architect Add-Ins can respond to the following events associated with user-generated element compartments:

- `EA_QueryAvailableCompartments`
- `EA_GetCompartmentData`

16.6.5.9.1 EA_QueryAvailableCompartments

**Details**

This event occurs when Enterprise Architect's diagrams are refreshed. It is a request for the Add-In to provide a list of user-defined compartments. The `EA_GetCompartmentData` event then queries each object for the data to display in each user-defined compartment.

**Syntax**

```vba
Function EA_QueryAvailableCompartments(Repository As EA.Repository) As Variant
    Dim sReturn As String
    sReturn = ""
    If m_FirstCompartmentVisible = True Then
        sReturn = sReturn + "first,"
    End If
    If m_SecondCompartmentVisible = True Then
        sReturn = sReturn + "second,"
    End If
    If m_ThirdCompartmentVisible = True Then
        sReturn = sReturn + "third,"
    End If
    If Len(sReturn) > 0 Then
        sReturn = Left(sReturn, Len(sReturn) - 1)
    End If
    EA_QueryAvailableCompartments = sReturn
End Function
```

**Return Value**

A `String` containing a comma-separated list of user-defined compartments.

16.6.5.9.2 EA_GetCompartmentData

**Details**

This event occurs when Enterprise Architect is instructed to redraw an element. It requests that the Add-In provide the data to populate the element's compartment.

**Syntax**

```vba
Function EA_GetCompartmentData(Repository As EA.Repository, sCompartment As String, sGUID As String) As Variant
    Dim sReturn As String
    If sCompartment = "first" Then
        sReturn = "First compartment data for object with GUID: " & sGUID
    ElseIf sCompartment = "second" Then
        sReturn = "Second compartment data for object with GUID: " & sGUID
    ElseIf sCompartment = "third" Then
        sReturn = "Third compartment data for object with GUID: " & sGUID
    End If
    EA_GetCompartmentData = sReturn
End Function
```
String, oType As EA.ObjectType) As Variant

The EA_QueryAvailableCompartments function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oType</td>
<td>ObjectType</td>
<td>IN</td>
<td>The type of the element for which data is being requested.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>sCompartment</td>
<td>String</td>
<td>IN</td>
<td>The name of the compartment for which data is being requested.</td>
</tr>
<tr>
<td>sGUID</td>
<td>String</td>
<td>IN</td>
<td>The GUID of the element for which data is being requested.</td>
</tr>
</tbody>
</table>

Return Value

Variant containing a formatted string. See the example below to understand the format.

Example

Function EA_GetCompartmentData(Repository As EA.Repository, sCompartment As String, sGUID As String, oType As EA.ObjectType) As Variant

    If Repository Is Nothing Then
        Exit Function
    End If

    Dim sCompartmentData As String
    Dim oXML As MSXML2.DOMDocument
    Dim Nodes As MSXML2.IXMLDOMNodeList
    Dim Node1 As MSXML2.IXMLDOMNode
    Dim Node As MSXML2.IXMLDOMNode
    Dim sData As String

    sCompartmentData = ""
    Set oXML = New MSXML2.DOMDocument
    sData = ""

    On Error GoTo ERR_GetCompartmentData

    oXML.loadXML (Repository.GetTreeXMLByGUID(sGUID))
    Set Node1 = oXML.selectSingleNode("/ModelItem")

    If Node1 Is Nothing Then
        Exit Function
    End If

    sCompartmentData = sCompartmentData + "Name=" + sCompartment + ";"
    sCompartmentData = sCompartmentData + "OwnerGUID=" + sGUID + ";"
    sCompartmentData = sCompartmentData + "Options=SkipIfOnDiagram&_eq_1&_sc_"

    Select Case sCompartment
        Case "ports"
            Set Nodes = Node1.selectNodes("ModelItem[@Metatype=""Part"]")
            For Each Node In Nodes
                sData = sData + "Data&_eq_" + Node.Attributes.getNamedItem("Name").nodeValue + ";_sc_"
                sData = sData + "GUID&_eq_" + Node.Attributes.getNamedItem("GUID").nodeValue + ";_sc_"
                Next
        Case "ports"
            Set Nodes = Node1.selectNodes("ModelItem[@Metatype=""Port"]")
            For Each Node In Nodes
                sData = sData + "Data&_eq_" + Node.Attributes.getNamedItem("Name").nodeValue + ";_sc_"
                sData = sData + "GUID&_eq_" + Node.Attributes.getNamedItem("GUID").nodeValue + ";_sc_"
                Next
        End Select

        ' If there's no data to display, then don't return any compartment data
        If sData <> "" Then
            sCompartmentData = sCompartmentData + "CompartmentData=" + sData + ";"
        End If
Else
    sCompartmentData = ""
End If

EA_GetCompartmentData = sCompartmentData
Exit Function

ERR_GetCompartmentData:
    EA_GetCompartmentData = ""
End Function

16.6.5.10 Model Validation Broadcasts

Perform Model Validation from an Add-In

Using Enterprise Architect broadcasts, it is possible to define a set of rules that are evaluated when the user instructs Enterprise Architect to perform model validation. An Add-In that performs model validation would involve the following broadcast events:

- **EA_OnInitializeUserRules** can be intercepted in order to define rule categories and rules.
- **EA_OnStartValidation** can be intercepted to perform any required processing prior to validation.
- The following functions intercept each request to validate an individual element, package, diagram, connector, attribute and method:
  - **EA_OnRunElementRule**
  - **EA_OnRunPackageRule**
  - **EA_OnRunDiagramRule**
  - **EA_OnRunConnectorRule**
  - **EA_OnRunAttributeRule**
  - **EA_OnRunMethodRule**
- **EA_OnEndValidation** can be intercepted to perform any required clean-up after validation has completed.

Consider the [Model Validation Example](#).

16.6.5.10.1 EA_OnInitializeUserRules

Details

**EA_OnInitializeUserRules** is called on Enterprise Architect start-up and requests that the Add-In provide Enterprise Architect with a rule category and list of rule IDs for model validation.

This function must be implemented by any Add-In that is to perform its own model validation. It must call **Project.DefineRuleCategory** once and **Project.DefineRule** for each rule; these functions are described in the **Project Interface** section.

Syntax

```vbscript
Sub EA_OnInitializeUserRules(Repository As EA.Repository)
```

The **EA_OnInitializeUserRules** function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

16.6.5.10.2 EA_OnStartValidation

Details

**EA_OnStartValidation** notifies Add-Ins that a user has invoked the model validation command from Enterprise Architect.
Syntax

Sub EA_OnStartValidation(Repository As EA.Repository, ParamArray Args() as Variant)

The EA_OnStartValidation function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Args</td>
<td>ParamArray of Variant</td>
<td>IN</td>
<td>Contains a list of Rule Categories that are active for the current invocation of model validation.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

16.6.5.10.3 EA_OnEndValidation

Details
EA_OnEndValidation notifies Add-Ins that model validation has completed. Use this event to arrange any clean-up operations arising from the validation.

Syntax

Sub EA_OnEndValidation(Repository As EA.Repository, ParamArray Args() as Variant)

The EA_OnEndValidation function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Args</td>
<td>ParamArray of Variant</td>
<td>IN</td>
<td>Contains a list of Rule Categories that were active for the invocation of model validation that has just completed.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

16.6.5.10.4 EA_OnRunElementRule

Details
This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each element in the selection being validated. If you don't want to perform the rule defined by RuleID on the given element, then simply return without performing any action. On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect. Also look at EA_OnInitializeUserRules.

Syntax

Sub EA_OnRunElementRule(Repository As EA.Repository, RuleID As String, Element As EA.Element)

The EA_OnRunElementRule function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>EA.Element</td>
<td>IN</td>
<td>The element to potentially perform validation on.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>RuleID</td>
<td>String</td>
<td>IN</td>
<td>The ID that was passed into the Project.DefineRule command.</td>
</tr>
</tbody>
</table>
16.6.5.10.5  EA_OnRunPackageRule

Details
This event is triggered once for each rule defined in `EA_OnInitializeUserRules` to be performed on each package in the selection being validated. If you don’t want to perform the rule defined by `RuleID` on the given package, then simply return without performing any action. On performing any validation, if a validation error is found, use the `Repository.ProjectInterface.PublishResult` method to notify Enterprise Architect.

Syntax

```
Sub EA_OnRunPackageRule(Repository As EA.Repository, RuleID As String, PackageID As Long)
```

The `EA_OnRunPackageRule` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageID</td>
<td>Long</td>
<td>IN</td>
<td>The ID of the package to potentially perform validation on. Use the <code>Repository.GetPackageByID</code> method to retrieve the package object.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>RuleID</td>
<td>String</td>
<td>IN</td>
<td>The ID that was passed into the <code>Project.DefineRule</code> method.</td>
</tr>
</tbody>
</table>

16.6.5.10.6  EA_OnRunDiagramRule

Details
This event is triggered once for each rule defined in `EA_OnInitializeUserRules` to be performed on each diagram in the selection being validated. If you don’t want to perform the rule defined by `RuleID` on the given diagram, then simply return without performing any action. On performing any validation, if a validation error is found, use the `Repository.ProjectInterface.PublishResult` method to notify Enterprise Architect.

Syntax

```
Sub EA_OnRunDiagramRule(Repository As EA.Repository, RuleID As String, DiagramID As Long)
```

The `EA_OnRunDiagramRule` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagramID</td>
<td>Long</td>
<td>IN</td>
<td>The ID of the diagram to potentially perform validation on. Use the <code>Repository.GetDiagramByID</code> method to retrieve the diagram object.</td>
</tr>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>RuleID</td>
<td>String</td>
<td>IN</td>
<td>The ID that was passed into the <code>Project.DefineRule</code> command.</td>
</tr>
</tbody>
</table>

16.6.5.10.7  EA_OnRunConnectorRule

Details
This event is triggered once for each rule defined in `EA_OnInitializeUserRules` to be performed on each connector in the selection being validated. If you don’t want to perform the rule defined by `RuleID` on the given connector, then simply return without performing any action. On performing any validation, if a validation error is found, use the `Repository.ProjectInterface.PublishResult` method to notify Enterprise Architect.

Syntax

```
Sub EA_OnRunConnectorRule(Repository As EA.Repository, RuleID As String, ConnectorID As Long)
```
The **EA_OnRunConnectorRule** function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorID</td>
<td>Long</td>
<td>IN</td>
<td>The ID of the connector to potentially perform validation on. Use the <strong>Repository.GetConnectorByID</strong> method to retrieve the connector object.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>RuleID</td>
<td>String</td>
<td>IN</td>
<td>The ID that was passed into the <strong>Project.DefineRule</strong> command.</td>
</tr>
</tbody>
</table>

### 16.6.5.10.8 EA_OnRunAttributeRule

**Details**

This event is triggered once for each rule defined in **EA_OnInitializeUserRules** to be performed on each attribute in the selection being validated. If you don't want to perform the rule defined by **RuleID** on the given attribute, then simply return without performing any action. On performing any validation, if a validation error is found, use the **Repository.ProjectInterface.PublishResult** method to notify Enterprise Architect.

**Syntax**

*Sub EA_OnRunAttributeRule(Repository As EA.Repository, RuleID As String, AttributeGUID As String, ObjectID As Long)*

The **EA_OnRunAttributeRule** function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeGUID</td>
<td>String</td>
<td>IN</td>
<td>The GUID of the attribute to potentially perform validation on. Use the <strong>Repository.GetAttributeByGuid</strong> method to retrieve the attribute object.</td>
</tr>
<tr>
<td>ObjectID</td>
<td>Long</td>
<td>IN</td>
<td>The ID of the object that owns the given attribute. Use the <strong>Repository.GetObjectByID</strong> method to retrieve the object.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>RuleID</td>
<td>String</td>
<td>IN</td>
<td>The ID that was passed into the <strong>Project.DefineRule</strong> command.</td>
</tr>
</tbody>
</table>

### 16.6.5.10.9 EA_OnRunMethodRule

**Details**

This event is triggered once for each rule defined in **EA_OnInitializeUserRules** to be performed on each method in the selection being validated. If you don't want to perform the rule defined by **RuleID** on the given method, then simply return without performing any action. On performing any validation, if a validation error is found, use the **Repository.ProjectInterface.PublishResult** method to notify Enterprise Architect.

**Syntax**

*Sub EA_OnRunMethodRule(Repository As EA.Repository, RuleID As String, MethodGUID As String, ObjectID As Long)*

The **EA_OnRunMethodRule** function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodGUID</td>
<td>String</td>
<td>IN</td>
<td>The GUID of the method to potentially perform validation on. Use the <strong>Repository.GetMethodByGuid</strong> method to retrieve the method object.</td>
</tr>
<tr>
<td>ObjectID</td>
<td>Long</td>
<td>IN</td>
<td>The ID of the object that owns the given method. Use the</td>
</tr>
</tbody>
</table>
### 16.6.5.10.10 EA_OnRunParameterRule

**Details**

This event is triggered once for each rule defined in `EA_OnInitializeUserRules` to be performed on each parameter in the selection being validated. If you don’t want to perform the rule defined by **RuleID** on the given parameter, then simply return without performing any action. On performing any validation, if a validation error is found, use the `Repository.ProjectInterface.PublishResult` method to notify Enterprise Architect.

**Syntax**

```csharp
Sub EA_OnRunParameterRule(Repository As EA.Repository, RuleID As String, ParameterGUID As String, MethodGUID As String, ObjectID As Long)
```

The `EA_OnRunParameterRule` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodGUID</td>
<td>String</td>
<td>IN</td>
<td>The GUID of the method that owns the given parameter. Use the <code>Repository.GetMethodByGuid</code> method to retrieve the method object.</td>
</tr>
<tr>
<td>ObjectID</td>
<td>Long</td>
<td>IN</td>
<td>The ID of the object that owns the given parameter. Use the <code>Repository.GetObjectByID</code> method to retrieve the object.</td>
</tr>
<tr>
<td>ParameterGUID</td>
<td>String</td>
<td>IN</td>
<td>The GUID of the parameter to potentially perform validation on. Use it to retrieve the parameter by iterating through the <code>Method.Parameters</code> collection.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>RuleID</td>
<td>String</td>
<td>IN</td>
<td>The ID that was passed into the <code>Project.DefineRule</code> command.</td>
</tr>
</tbody>
</table>

### 16.6.5.10.11 Model Validation Example

The following example code is written in C# and provides a skeleton model validation implementation that you might like to use as a starting point in writing your own model validation rules.

#### Main.cs

```csharp
using System;
namespace myAddin
{
    public class Main
    {
        public Rules theRules;
        public Main()
        {
            theRules = new Rules();
        }
        public string EA_Connect(EA.Repository Repository)
        {
            return "";
        }
    }
}
```
public void EA_Disconnect()
{
    GC.Collect();
    GC.WaitForPendingFinalizers();
}

private bool IsProjectOpen(EA.Repository Repository)
{
    try
    {
        EA.Collection c = Repository.Models;
        return true;
    }
    catch
    {
        return false;
    }
}

public object EA_GetMenuItems(EA.Repository Repository, string MenuLocation, string MenuName)
{
    switch (MenuName)
    {
    case "":
        return "+-&myAddin";
    case "+-&myAddin":
        string[] ar = { "&Test" ];
        return ar;
    }
    return "";
}

public void EA_GetMenuState(EA.Repository Repository, string MenuLocation, string MenuName, string ItemName, ref bool isEnabled, ref bool IsChecked)
{
    // if no open project, disable all menu options
    if (IsProjectOpen(Repository))
        isEnabled = true;
    else
        isEnabled = false;
}

public void EA_MenuClick(EA.Repository Repository, string MenuLocation, string MenuName, string ItemName)
{
    switch (ItemName)
    {
    case "&Test":
        DoTest(Repository);
        break;
    }
}

public void EA_OnInitializeUserRules(EA.Repository Repository)
{
    if (Repository != null)
    {
        theRules.ConfigureCategories(Repository);
        theRules.ConfigureRules(Repository);
    }
}

public void EA_OnRunElementRule(EA.Repository Repository, string RuleID, EA.Element element)
{
    theRules.RunElementRule(Repository, RuleID, element);
}

public void EA_OnRunDiagramRule(EA.Repository Repository, string RuleID, long IDiagramID)
{
    theRules.RunDiagramRule(Repository, RuleID, IDiagramID);
}

public void EA_OnRunConnectorRule(EA.Repository Repository, string RuleID, long IConnectorID)
{
    theRules.RunConnectorRule(Repository, RuleID, IConnectorID);
}
public void EA_OnRunAttributeRule(EA.Repository Repository, string RuleID, string AttGUID, long lObjectID)
{
    return;
}

public void EA_OnDeleteTechnology(EA.Repository Repository, EA.EventProperties Info)
{
    return;
}

public void EA_OnImportTechnology(EA.Repository Repository, EA.EventProperties Info)
{
    return;
}

private void DoTest(EA.Repository Rep)
{
    // TODO: insert test code here
}

Rules.cs

using System;
using System.Collections;

namespace myAddin
{
    public class Rules
    {
        private string m_sCategoryID;
        private System.Collections.ArrayList m_RuleIDs;
        private System.Collections.ArrayList m_RuleIDEx;

        private const string cRule01 = "Rule01";
        private const string cRule02 = "Rule02";
        private const string cRule03 = "Rule03";
        // TODO: expand this list as much as necessary

        public Rules()
        {
            m_RuleIDs = new System.Collections.ArrayList();
            m_RuleIDEx = new System.Collections.ArrayList();
        }

        private string LookupMap(string sKey)
        {
            return DoLookupMap(sKey, m_RuleIDs, m_RuleIDEx);
        }

        private string LookupMapEx(string sRule)
        {
            return DoLookupMap(sRule, m_RuleIDEx, m_RuleIDs);
        }

        private string DoLookupMap(string sKey, ArrayList arrValues, ArrayList arrKeys)
        {
            if (arrKeys.Contains(sKey))
                return arrValues[arrKeys.IndexOf(sKey)].ToString();
            else
                return "";
        }

        private void AddToMap(string sRuleID, string sKey)
        {
            m_RuleIDs.Add(sRuleID);
            m_RuleIDEx.Add(sKey);
        }

        private string GetRuleStr(string sRuleID)
        {
            switch (sRuleID)
            {
public void ConfigureCategories(EA.Repository Repository)
{
    EA.Project Project = Repository.GetProjectInterface();
    m_sCategoryID = Project.DefineRuleCategory("Enterprise Collaboration Architecture (ECA)
    Rules");
}

public void ConfigureRules(EA.Repository Repository)
{
    EA.Project Project = Repository.GetProjectInterface();
    AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrortype.mvError,
    GetRuleStr(cRule01)), cRule01);
    AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrortype.mvError,
    GetRuleStr(cRule02)), cRule02);
    AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrortype.mvError,
    GetRuleStr(cRule03)), cRule03);
    // TODO: expand this list
}

public void RunConnectorRule(EA.Repository Repository, String sRuleID, long lConnectoid)
{
    EA.Connector Connector = Repository.GetConnectorById((int)lConnectoid);
    if (Connector != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule02:
                // TODO: perform rule 2 check
                break;
            // TODO: add more cases
        }
    }
}

public void RunDiagramRule(EA.Repository Repository, String sRuleID, long lDiagramid)
{
    EA.Diagram Diagram = Repository.GetDiagramById((int)lDiagramid);
    if (Diagram != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule03:
                // TODO: perform rule 3 check
                break;
            // TODO: add more cases
        }
    }
}

public void RunElementRule(EA.Repository Repository, String sRuleID, EA.Element Element)
{
    if (Element != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule01:
            DoRule01(Repository, Element);
            break;
            // TODO: add more cases
        }
    }
}

private void DoRule01(EA.Repository Repository, EA.Element Element)
{
    if (Element.Stereotype != "myStereotype")
    {
        case cRule01:
            return "Error Message 01";
        case cRule02:
            return "Error Message 02";
        case cRule03:
            return "Error Message 03";
        // TODO: add extra cases as much as necessary
    }
    return "";
}
16.6.5.11  EA_OnRetrieveModelTemplate

Details

`EA_OnRetrieveModelTemplate` requests that an Add-in pass a model template to Enterprise Architect.

This event occurs when a user executes the **Add Model Using Wizard** command to add a model that has been defined by an MDG Technology. See the [Incorporate Model Templates in a Technology](#) topic for details of how to define such model templates.

Syntax

```plaintext
Function EA_OnRetrieveModelTemplate(Repository As EA.Repository, sLocation As String) As String
```

The `EA_OnRetrieveModelTemplate` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td><code>EA.Repository</code></td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>sLocation</td>
<td>String</td>
<td>IN</td>
<td>The name of the template requested. This should match the <code>location</code> attribute in the <code>&lt;ModelTemplates&gt;</code> section of an MDG Technology File. For more information, see the <a href="#">Incorporate Model Templates in a Technology</a> topic.</td>
</tr>
</tbody>
</table>

Return Value

Return a string containing the XMI export of the model that is being used as a template.

Example

```vbnet
Public Function EA_OnRetrieveModelTemplate(ByVal Rep As EA.Repository, ByVal sLocation As String) As String
    Dim sTemplate As String
    Select Case sLocation
        Case "Templates\Template1.xml"
            sTemplate = My.Resources.Template1
        Case "Templates\Template2.xml"
            sTemplate = My.Resources.Template2
        Case "Templates\Template3.xml"
            sTemplate = My.Resources.Template3
        Case Else
            MsgBox("Path for ", & sLocation & ", not found")
            sTemplate = ""
    End Select
    EA_OnRetrieveModelTemplate = sTemplate
End Function
```

16.6.5.12  EA_OnInitialize_Technologies

Details

`EA_OnInitializeTechnologies` requests that an add-in pass an MDG Technology to Enterprise Architect for loading.

This event occurs on Enterprise Architect startup. Return your technology XML to this function and Enterprise Architect loads and enables it.
Syntax

Function EA_OnInitializeTechnologies(Repository As EA.Repository) As Object

The EA_OnInitializeTechnologies function syntax contains the following element:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

Return the MDG Technology as a single XML string.

Example

Public Function EA_OnInitializeTechnologies(ByVal Repository As EA.Repository) As Object
    EA_OnInitializeTechnologies = My.Resources.MyTechnology
End Function
16.6.6 Custom Views

Enterprise Architect enables custom windows to be inserted as tabs in the Diagram View that appears at the center of the Enterprise Architect frame.

Creating a custom view enables you to easily and quickly tab between a custom interface and diagrams and other views normally provided by Enterprise Architect.

Uses for this facility include:
- Reports and graphs showing summary data of the model
- Alternative views of a diagram
- Alternative views of the model
- Views of external data related to model data
- Documentation tools.

16.6.6.1 Create a Custom View

A custom view must 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. See the documentation provided by the relevant vendor on how to create a custom control to produce an OCX file.

Once the custom control 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 would call it from an Add-In, and that Add-In is defined in the same OCX that provides the custom view.

Example C# code is shown below:

```
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 "&Create Control Demo";
        else
        {
            String[] ret = {"&Create", "&Show Button"};
            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();
    }
}
```
16.6.7 MDG Add-Ins

MDG Add-Ins are specialized types of Add-Ins that have additional features and extra requirements for Add-In authors who want to contribute to Enterprise Architect'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 an Enterprise Architect 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 Enterprise Architect model, which is done through the \texttt{MDG\_Connect} event.

MDG Add-Ins identify themselves as such during \texttt{EA\_Connect} by returning the string \texttt{MDG}.

Unlike ordinary Add-Ins, responding to MDG Add-In events is not optional, and methods must be published for each of the \texttt{MDG\_Events}.

Two examples of MDG Add-Ins are the commercially available \textit{MDG Link for Eclipse} and \textit{MDG Link for Visual Studio}, published by \textit{Sparx Systems}.

16.6.7.1 MDG Events

An MDG Add-In must respond to all MDG Events. These events usually identify processes such as Build, Run, Synchronize, PreMerge and PostMerge, amongst others.

An MDG Link Add-In is expected to implement some form of forward and reverse engineering capability within Enterprise Architect, 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.

- \texttt{MDG\_BuildProject}
- \texttt{MDG\_Connect}
- \texttt{MDG\_Disconnect}
- \texttt{MDG\_GetConnectedPackages}
- \texttt{MDG\_GetProperty}
- \texttt{MDG\_Merge}
- \texttt{MDG\_NewClass}
- \texttt{MDG\_PostGenerate}
- \texttt{MDG\_PostMerge}
- \texttt{MDG\_PreGenerate}
- \texttt{MDG\_PreMerge}
- \texttt{MDG\_PreReverse}
- \texttt{MDG\_RunExe}
- \texttt{MDG\_View}

16.6.7.1.1 MDG\_Build Project

Details

\texttt{MDG\_BuildProject} enables the Add-In to handle file changes caused by generation. This function is called in response to a user selecting the \textit{Add-Ins | Build Project} menu option.

Respond to this event by compiling the project source files into a running application.

Also look at \texttt{MDG\_RunExe}.

Syntax

\texttt{Sub MDG\_BuildProject(Repository As EA\_Repository, PackageGuid As String)}

The \texttt{MDG\_BuildProject} function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA_Repository</td>
<td>IN</td>
<td>An \textit{EA_Repository} object representing the currently open</td>
</tr>
</tbody>
</table>
### Parameter Type Direction Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

None.

### 16.6.7.1.2 MDGConnect

**Details**

MDG_Connect enables the Add-In to handle user driven request to connect a model branch to an external application. This function is called when the user attempts to connect a particular Enterprise Architect package to an as yet unspecified external project. This event enables 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. That is, users who share a common Enterprise Architect 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 Enterprise Architect 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 Enterprise Architect, should this be required.

Also look at MDG_Disconnect.

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The unique ID identifying the project provided by the Add-In when a connection to a project branch of an Enterprise Architect model was first established.</td>
</tr>
<tr>
<td>PackageID</td>
<td>Long</td>
<td>IN</td>
<td>The PackageID of the Enterprise Architect package the user has requested to have connected to an external project.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

**Return Value**

Returns a non-zero to indicate that a connection has been made; a zero indicates that the user has not nominated a project and connection should not proceed.

### 16.6.7.1.3 MDGDisconnect

**Details**

MDG_Disconnect enables the Add-In to respond to user requests to disconnect the model branch from an external project. This function is called when the user attempts to disconnect an associated external project. The Add-In is required to delete the details of the connection.

Also look at MDG_Connect.
Syntax

Function MDG_Disconnect(Repository As EA.Repository, PackageGuid As String) As Long

The `MDG_Disconnect` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

Returns a non-zero to indicate that a disconnection has occurred enabling Enterprise Architect to update the user interface. A zero indicates that the user has not disconnected from an external project.

16.6.7.1.4 MDGGetConnectedPackages

Details

`MDG_GetConnectedPackages` enables the Add-In to return a list of current connection between Enterprise Architect and an external application. This function is 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.

Also look at `MDG_Connect`.

Syntax

Function MDG_GetConnectedPackages(Repository As EA.Repository) As Variant

The `MDG_GetConnectedPackages` function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <code>EA.Repository</code> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

Returns an array of GUID strings representing individual Enterprise Architect packages.

16.6.7.1.5 MDGGetProperty

Details

`MDGGetProperty` provides miscellaneous Add-In details to Enterprise Architect. This function is called by Enterprise Architect to poll the Add-In for information relating to the `PropertyName`. This event should occur in as short a duration as possible as Enterprise Architect does not cache the information provided by the function.

Values corresponding to the following `PropertyNames` must 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 Enterprise Architect
- **HiddenMenus** - Return one or more values from the `MDGMenus` enumeration to hide menus that do not apply to your Add-In. For example:

  ```java
  if( PropertyName == "HiddenMenus" )
  return mgBuildProject + mgRun;
  ```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
<tr>
<td>PropertyName</td>
<td>String</td>
<td>IN</td>
<td>The name of the property that is used by Enterprise Architect. See Details for the possible values.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>

Return Value

See Details, above.

16.6.7.1.6 MDGMerge

Details

MDG_Merge enables the Add-In to jointly handle changes to both the model branch and the code project that the model branch is connected to. This event should 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 enable the Add-In to interact with the user to perform a merge between the model branch and the connected project.

Also look at MDG_Connect, MDG_PreMerge and MDG_PostMerge.

Syntax

Function MDG_Merge(Repository As EA.Repository, PackageGuid As String, SynchObjects As Variant, SynchType As String, ExportObjects As Variant, ExportFiles As Variant, ImportFiles As Variant, IgnoreLocked As String, Language As String) As Long

The MDG_Merge function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportFiles</td>
<td>Variant</td>
<td>OUT</td>
<td>A string array containing the list of files for each model object chosen for export by the Add-In. Each entry in this array must have a corresponding entry in the ExportObjects parameter at the same array index, so ExportFiles(2) must contain the filename of the object by ExportObjects(2).</td>
</tr>
<tr>
<td>ExportObjects</td>
<td>Variant</td>
<td>OUT</td>
<td>The string array containing the list of new model objects (in Object ID format) to be exported by Enterprise Architect to the code project.</td>
</tr>
<tr>
<td>IgnoreLocked</td>
<td>String</td>
<td>OUT</td>
<td>A value indicating whether to ignore any files locked by the code project (i.e &quot;TRUE&quot; or &quot;FALSE&quot;.</td>
</tr>
<tr>
<td>ImportFiles</td>
<td>Variant</td>
<td>OUT</td>
<td>A string array containing the list of code files made available to the code project to be newly imported to the model. Enterprise Architect imports each file listed in this array for import into the connected model branch.</td>
</tr>
<tr>
<td>Language</td>
<td>String</td>
<td>OUT</td>
<td>The string value containing the name of the code language supported by the code project connected to the model branch.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
</tbody>
</table>
## Parameter Description Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>SynchObjects</td>
<td>Variant</td>
<td>OUT</td>
<td>A string array containing a list of objects (Object ID format) to be jointly synchronized between the model branch and the project. See below for the format of the Object IDs.</td>
</tr>
<tr>
<td>SynchType</td>
<td>String</td>
<td>OUT</td>
<td>The value determining the user-selected type of synchronization to take place. See below for a list of valid values.</td>
</tr>
</tbody>
</table>

### Return Value

Return a non-zero if the merge operation completed successfully and a zero value when the operation has been unsuccessful.

### Merge

A merge consists 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 and such things 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 corresponds 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 should be composed in the following format:

```
(@namespace)*(#class)*($attribute|%operation|:property)*
```

### 16.6.7.1.7 MDGNewClass

#### Details

`MDG_NewClass` enables the Add-In to alter details of a Class before it is created.

This method is called when Enterprise Architect generates a new Class, and requires 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.

Also look at `MDG_PreGenerate`.

#### Syntax

```vbnet
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeID</td>
<td>String</td>
<td>IN</td>
<td>A string used to identify the code element before it is created, for</td>
</tr>
</tbody>
</table>
### Parameter | Type | Direction | Description
--- | --- | --- | ---
Language | String | IN | A string used to identify the programming language for the new Class. The language must be supported by Enterprise Architect.
PackageGuid | String | IN | The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.
Repository | EA.Repository | IN | An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

### Return Value
Returns a string containing the file path that should be assigned to the Class.

### 16.6.7.1.8 MDGPostGenerate

**Details**

MDG_PostGenerate enables the Add-In to handle file changes caused by generation.

This event is called after Enterprise Architect has prepared text to replace the existing contents of a file. Responding to this event enables the Add-In to write to the linked application's user interface rather than modify the file directly.

When the contents of a file are changed, Enterprise Architect passes FileContents as a non-empty string. New files created as a result of code generation are also sent through this mechanism, enabling Add-Ins to add new files to the linked project's file list.

When new files are created Enterprise Architect passes 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 Enterprise Architect that the file must be saved.

Also look at MDG_PreGenerate.

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileContents</td>
<td>String</td>
<td>IN</td>
<td>A string containing the proposed contents of the file.</td>
</tr>
<tr>
<td>FilePath</td>
<td>String</td>
<td>IN</td>
<td>The path of the file Enterprise Architect intends to overwrite.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
</tbody>
</table>
16.6.7.1.9 MDGPostMerge

Details

MDG_PostMerge is called after a merge process has been completed. This function is called by Enterprise Architect after the merge process has been completed.

Note:

File save checking should not be performed with this function, but should be handled by MDG_PreGenerate and MDG_PostGenerate and MDG_PreReverse.

Also look at MDG_PreMerge and MDG_Merge.

Syntax

Function MDG_PostMerge(Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PostMerge function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
</tbody>
</table>

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. Enterprise Architect assumes a non-zero return if this method is not implemented.

16.6.7.1.10 MDGPreGenerate

Details

MDG_PreGenerate enables the Add-In to deal with unsaved changes. This function is called immediately before Enterprise Architect attempts to generate files from the model. A possible use of this function would be to prompt the user to save unsaved source files.

Also look at MDG_PostGenerate.

Syntax

Function MDG_PreGenerate(Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PreGenerate function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
</tbody>
</table>

Return Value

Return a zero value to abort generation. Any other value enables the generation to continue.
16.6.7.1.11 MDGPreMerge

Details

MDG_PreMerge is called after a merge process has been initiated by the user and before Enterprise Architect performs the merge process.

This event is called after a user has performed their interactions with the merge screen and has confirmed the merge with the OK button, but before Enterprise Architect 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 with 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 MDG_PreGenerate and MDG_PostGenerate.

Syntax

Function MDG_PreMerge(Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PreMerge function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
</tbody>
</table>

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 proceed. If this method is not implemented then it is assumed that a merge process is used.

16.6.7.1.12 MDGPreReverse

Details

MDG_PreReverse enables the Add-In to save file changes before being imported into Enterprise Architect.

This function operates on a list of files that are about to be reverse-engineered into Enterprise Architect. If the user is working on unsaved versions of these files in an editor, you could either prompt the user or save automatically.

Syntax

Sub MDG_PreReverse(Repository As EA.Repository, PackageGuid As String, FilePaths As Variant)

The MDG_PreReverse function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>FilePaths</td>
<td>String array</td>
<td>IN</td>
<td>An array of filepaths pointed to the files that are to be reverse engineered.</td>
</tr>
</tbody>
</table>

**Return Value**
None.

### 16.6.7.1.13 MDGRunExe

**Details**

_MDG_RunExe_ enables the Add-In to run the target application. This function is called when the user selects the _Add-Ins | Run Exe_ menu option. Respond to this event by launching the compiled application.

Also look at _MDG_BuildProject_.

**Syntax**

**Sub MDG_RunExe(Repository As EA.Repository, PackageGuid As String)**

The _MDG_RunExe_ function syntax contains the following elements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <em>EA.Repository</em> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
</tbody>
</table>

**Return Value**
None.

### 16.6.7.1.14 MDGView

**Details**

_MDG_View_ enables the Add-In to display user specified code elements. This function is called by Enterprise Architect when the user asks to view a particular code element. This enables the Add-In to present that element in its own way, usually in a code editor.

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>EA.Repository</td>
<td>IN</td>
<td>An <em>EA.Repository</em> object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.</td>
</tr>
<tr>
<td>PackageGuid</td>
<td>String</td>
<td>IN</td>
<td>The GUID identifying the Enterprise Architect package sub-tree that is controlled by the Add-In.</td>
</tr>
</tbody>
</table>
| CodeID      | String        | IN        | Identifies the code element in the following format:
<type>ElementPart<type>ElementPart...
where each element is proceeded with a token identifying its type:
@ -namespace
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td># - Class</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$ - attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>% - operation</td>
</tr>
</tbody>
</table>

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:

@Name1#Class1%m_Name

**Return Value**

Return a non-zero value to indicate that the Add-In has processed the request. Returning a zero value results in Enterprise Architect employing the standard viewing process which is to launch the associated source file.
16.7 Enterprise Architect Object Model

Introduction
Automation provides a way for other applications to access the information in an Enterprise Architect model using Windows OLE Automation (ActiveX). Typically this involves scripting clients such as MS Word or Visual Basic.

The Automation Interface provides a way of accessing the internals of Enterprise Architect models. Examples of things you can do using the Automation Interface include:

- 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
- Perform ad hoc queries.

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, Microsoft C# and Java. There are also more detailed steps on how to set-up Visual Basic; the principles are 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, consult the extensive Reference Section.

Calling Executables from Enterprise Architect
Enterprise Architect can be set up to call an external application. You can pass parameters on the current position selected in the Project Browser to the application being called. For instructions, go to the Call from Enterprise Architect topic. A more sophisticated method is to create Add-Ins which are discussed in a separate topic.
16.7.1 Using the Automation Interface

This section provides instructions on how to connect to and use the Automation Interface, including:

- Connecting to the Interface
- Set References In Visual Basic
- Examples and Tips

16.7.1.1 Connect 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 sections describe how to connect using several such tools. The procedure might vary slightly with different versions of these products.

Microsoft Visual Basic 6.0

1. Create a new project.
2. Select the Project | References menu option.
3. Select Enterprise Architect Object Model 2.0 from the list. (If this does not appear, go to the command line and re-register Enterprise Architect using
   ```
   EA.exe /unregister
   then
   EA.exe /register).
   ```
4. See 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. Select the Project | Import Type Library menu option.
3. Select Enterprise Architect Object Model 2.0 from the list. (If this does not appear, go to the command line and re-register Enterprise Architect using
   ```
   EA.exe /unregister
   then
   EA.exe /register).
   ```
4. Click on the Create Unit button.
5. Include EA_TLB in Project1’s Uses clause.
6. See 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 the Visual Studio Project | Add Reference menu option.
2. Click on the Browse tab.
3. Navigate to the folder in which you installed Enterprise Architect (usually Program Files/Sparx Systems/EA) and select Interop.EA.dll.
4. See the general library documentation on the use of Classes. The following example creates and opens a repository object:

```csharp
private void button1_Click(object sender, System.EventArgs e){
    EA.Repository r = new EA.RepositoryClass();
    r.OpenFile("c:\eatest.eap");
}
```

Java

1. Copy the file SSJavaCOM.dll from the Java API subdirectory of your installed directory (usually Program Files/Sparx Systems/EA) into any location within the Windows PATH. For example, the windows\system32 directory.
2. Copy the eaapi.jar file from the Java API subdirectory of your installed directory (usually Program Files/Sparx Systems/EA) to a location in the Java CLASSPATH or where the Java class loader can find it at runtime.
3. All of the Classes described in the documentation are in the package org.sparx. See the general library documentation for their use. The following example creates and opens a repository object.

```java
public void OpenRepository(){
    org.sparx.Repository r = new org.sparx.Repository();
    r.OpenFile("c:\eatest.eap");
}
```

16.7.1.1 Set References In Visual Basic

This topic describes how to use the Enterprise Architect ActiveX interface with Visual Basic (VB). Use is ensured for Visual Basic version 6. This might vary slightly with versions other than version 6.

It is assumed that you have accessed VB 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. Follow the steps below:

1. Select the Tools | References menu option. The following dialog displays:

   ![Available References Dialog]

2. Select the Enterprise Architect Object Model 2.10 checkbox from the list.
3. Do the same for VB or VB Word: select the checkbox for the Microsoft Word 10.0 Object Library.
4. Click on the OK button.
Note:

If Enterprise Architect Object Model 2.10 does not appear in the list, go to the command line and manually re-register Enterprise Architect using the following:

- To unregister Enterprise Architect: ea.exe /unregister

Visual Basic 5/6 users should also note that the version number of the Enterprise Architect interface is stored in the VBP project file in a form similar to the following:

Reference="\[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 Enterprise Architect 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 Enterprise Architect Object model.

Reference to objects in Enterprise Architect and Word should now be available in the Object Browser. This can be accessed from the main menu by selecting View | Object Browser, or by pressing [F2].

The drop-down list on the top-left of the window should now include Enterprise Architect and Word. If MS-Project is installed this must also be set up.

16.7.1.2 Examples and Tips

Instructions for using the interface are provided through sample code. There are several sets of examples:

- VB 6 and C# examples are available in the Code Samples folder under your Enterprise Architect installation (default: C:\Program Files\Sparx Systems\EA\Code Samples).
- Enterprise Architect can be set up to call an external application.
- Several VB.NET code snippets are 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/resources/developers/autint_vb.html.
- Additional samples are available from the Sparx Systems website; see the Available Resources topic.
Additionally, you should note the following tips and tricks:

- An instance of the Enterprise Architect (EA.exe) process is executed when you initialize a new repository object. This process must remain running in order to perform automation tasks. If the main window is visible, you can safely minimize it, but it must remain running.

- The Enterprise Architect ActiveX Interface is a functional interface rather than a data interface. When you load data through the interface there is a noticeable delay as Enterprise Architect user interface elements (e.g. Windows, menus) are loaded and the specified database connection is established.

- Collections use a zero-based index; for example, Repository.Models(0) represents the first model in the repository.

- During the development of your client software your program might terminate unexpectedly and leave EA.exe running in such a state that it is unable to support further interface calls. If your program terminates abnormally, ensure that Enterprise Architect is not left running in the background (see the Windows Task Manager / Process tab).

- A handle to a currently running instance of Enterprise Architect can be obtained through the use of a GetObject() call. For more information, refer to the reference page for the App object. Accessing your Enterprise Architect model via the App object enables querying the current User Interface status, such as using GetContextItem() on the Repository object to detect the current selection by the user, allowing for rapid prototyping and testing.

**Enterprise Architect Not Closing**

If your automation controller was written using the .NET framework, Enterprise Architect does 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:

```csharp
GC.Collect();
GC.WaitForPendingFinalizers();
```

There are additional concerns when controlling a running instance of Enterprise Architect that loads Add-Ins - see the Tricks and Traps topic for details.

### 16.7.1.2.1 Call from Enterprise Architect

Enterprise Architect can be set up to call an external application. You can pass parameters on the current position selected in the Project Browser to the application being called.

To define an application that you can run from Enterprise Architect, select the Tools | Customize menu option. The Customize dialog displays. Select the Tools tab.

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:

- The Enterprise Architect file parameter $f$
- The current PackageID $p$

Hence the arguments should simply contain: $f, p$

The available parameters for passing information to external applications are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>Project Name</td>
<td>For example: C:\projects\EAexample.eap.</td>
</tr>
<tr>
<td>$F$</td>
<td>Calling Application (Enterprise Architect)</td>
<td>'Enterprise Architect'.</td>
</tr>
<tr>
<td>$p$</td>
<td>Current Package ID</td>
<td>For example: 144.</td>
</tr>
<tr>
<td>$P$</td>
<td>Package GUID</td>
<td>GUID for accessing this package.</td>
</tr>
<tr>
<td>$d$</td>
<td>Diagram ID</td>
<td>ID for accessing associated diagram.</td>
</tr>
<tr>
<td>$D$</td>
<td>Diagram GUID</td>
<td>GUID for accessing the associated diagram.</td>
</tr>
<tr>
<td>$e$</td>
<td>Comma separated list of element IDs</td>
<td>All elements selected in the current diagram.</td>
</tr>
<tr>
<td>$E$</td>
<td>Comma separated list of element GUIDs</td>
<td>All elements selected in the current diagram.</td>
</tr>
</tbody>
</table>

Once this has been set up, the application can be called from the main menu in Enterprise Architect using the Tools | YourApplication menu option.

### 16.7.1.2.2 Available Resources

Other available resources include:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB 6 Add-In for generating MS Word documentation.</td>
<td><a href="http://www.sparxsystems.com/resources/developers/autint_vb.html">www.sparxsystems.com/resources/developers/autint_vb.html</a></td>
</tr>
<tr>
<td>VB 6 Add-In to display a custom ActiveX graph control within the Enterprise Architect window as a new view.</td>
<td><a href="http://www.sparxsystems.com/resources/developers/autint_vb_custom_view.html">www.sparxsystems.com/resources/developers/autint_vb_custom_view.html</a></td>
</tr>
<tr>
<td>A basic Add-In framework written in C#. Useful as a starting point for authoring your own custom Enterprise Architect Add-In.</td>
<td><a href="http://www.sparxsystems.com/bin/CS_AddinFramework.zip">www.sparxsystems.com/bin/CS_AddinFramework.zip</a></td>
</tr>
<tr>
<td>An extension on the CS_AddinFramework example showing how to export Tagged Values to a .csv file.</td>
<td><a href="http://www.sparxsystems.com/bin/CS_AddinTaggedCSV.zip">www.sparxsystems.com/bin/CS_AddinTaggedCSV.zip</a></td>
</tr>
<tr>
<td>A basic Add-In skeleton written in Delphi.</td>
<td><a href="http://www.sparxsystems.com/bin/DelphiDemo.zip">www.sparxsystems.com/bin/DelphiDemo.zip</a></td>
</tr>
<tr>
<td>A simple example Add-In written in C#.</td>
<td><a href="http://www.sparxsystems.com/bin/CS_Sample.zip">www.sparxsystems.com/bin/CS_Sample.zip</a></td>
</tr>
</tbody>
</table>

For further information, see www.sparxsystems.com/resources/developers/autint.html.
16.7.2 Reference

This section provides detailed information on all the objects available in the object model provided by the Automation Interface, covering:

- Interface Overview
- App
- Enumerations
- Repository
- Element
- Element Features
- Connector
- Diagram Package
- Project Interface
- Code Samples

16.7.2.1 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 (e.g. Class, Use Case and Object)
- Element Features which are attributes and operations defined on an element
- Diagram Package the visible drawings contained in the model
- Connectors relationships between elements.

The following diagram illustrates the main interface elements and their associated contents. Each element in this document is creatable by Automation and can be accessed through the various collections that exist or, in some cases, directly.
The following diagram provides a high level overview of the Automation Interface for accessing, manipulating, modifying and creating Enterprise Architect 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 that provides access to the UML elements, diagrams and packages within the project. In general, the Role names applied at the Target end of associations indicate the name of the Collection that is used to access instances of that object.
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
- Logical diagram:: Automation Interface Package:: Automation Interface
- Logical diagram:: Automation Interface Package:: Automation Interface
Connectors

<table>
<thead>
<tr>
<th>Connector</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nesting source &gt; target</td>
<td>Connector Contained Element</td>
<td>Overview Containing Element</td>
</tr>
<tr>
<td>Nesting source &gt; target</td>
<td>Repository Contained Element</td>
<td>Overview Containing Element</td>
</tr>
<tr>
<td>Nesting source &gt; target</td>
<td>Diagram Contained Element</td>
<td>Overview Containing Element</td>
</tr>
<tr>
<td>Nesting source &gt; target</td>
<td>Element Contained Element</td>
<td>Overview Containing Element</td>
</tr>
<tr>
<td>Nesting source &gt; target</td>
<td>Project Interface Contained Element</td>
<td>Overview Containing Element</td>
</tr>
<tr>
<td>Nesting source &gt; target</td>
<td>ElementFeatures Contained Element</td>
<td>Overview Containing Element</td>
</tr>
</tbody>
</table>

16.7.2.2 App

The App object represents a running instance of Enterprise Architect. Its object provides access to the Automation Interface.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Project</td>
<td>Read Only. Provides a handle to the Project Interface.</td>
</tr>
<tr>
<td>Repository</td>
<td>Repository</td>
<td>Read Only. Provides a handle to the Repository object.</td>
</tr>
<tr>
<td>Visible</td>
<td>Boolean</td>
<td>Read/Write. Whether or not the application is visible.</td>
</tr>
</tbody>
</table>

GetObject() Support

The App object is creatable and a handle can be obtained by creating one. In addition, clients can use the equivalent of Visual Basic's GetObject() to obtain a reference to a currently running instance of Enterprise Architect.

Use this method to more quickly test changes to Add-Ins and external clients, as the Enterprise Architect application and data files do not have to be constantly re-loaded.

For example:

Dim App as EA.App
Set App = GetObject(,”EA.App”)  

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

16.7.2.3 Enumerations

These enumerations are defined by the Automation Interface:

- ConstLayoutStyles Enum
- EnumRelationSetType Enum
- MDGMenus Enum
- ObjectType Enum
- PropType Enum
- ReloadType Enum
- XMIType Enum
### 16.7.2.3.1 ConstLayoutStyles Enum

The enum values defined here are used exclusively for the *Lay Out a Diagram* method. They enable you to define the layout options as depicted in the *Layout a Diagram* menu option (for further information, see the *Enterprise Architect User Guide, Lay Out a Diagram* topic).

<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsCrossReduceAggressive</td>
<td>Perform aggressive Cross-reduction in the layout process (time consuming).</td>
</tr>
<tr>
<td>lsCycleRemoveDFS</td>
<td>Use the Depth First Cycle Removal algorithm.</td>
</tr>
<tr>
<td>lsCycleRemoveGreedy</td>
<td>Use the Greedy Cycle Removal algorithm.</td>
</tr>
<tr>
<td>lsDiagramDefault</td>
<td>Use existing layout options specified for this diagram.</td>
</tr>
<tr>
<td>lsInitializeDFSIn</td>
<td>Initialize the layout using the Depth First Search Inward algorithm.</td>
</tr>
<tr>
<td>lsInitializeNaive</td>
<td>Initialize the layout using the Naïve Initialize Indices algorithm.</td>
</tr>
<tr>
<td>lsInitializeDFSOut</td>
<td>Initialize the layout using the Depth First Search Outward algorithm.</td>
</tr>
<tr>
<td>lsLayeringLongestPathSink</td>
<td>Layer the diagram using the Longest Path Sink algorithm.</td>
</tr>
<tr>
<td>lsLayeringLongestPathSource</td>
<td>Layer the diagram using the Longest Path Source algorithm.</td>
</tr>
<tr>
<td>lsLayeringOptimalLinkLength</td>
<td>Layer the diagram using the Optimal Link Length algorithm.</td>
</tr>
<tr>
<td>lsLayoutDirectionDown</td>
<td>Direct connectors to point downwards.</td>
</tr>
<tr>
<td>lsLayoutDirectionLeft</td>
<td>Direct connectors to point leftwards.</td>
</tr>
<tr>
<td>lsLayoutDirectionRight</td>
<td>Direct connectors to point rightwards.</td>
</tr>
<tr>
<td>lsLayoutDirectionUp</td>
<td>Direct connectors to point upwards.</td>
</tr>
<tr>
<td>lsProgramDefault</td>
<td>Use factory default layout options as specified by Enterprise Architect.</td>
</tr>
</tbody>
</table>

### 16.7.2.3.2 EnumRelationSetType Enum

This enumeration represents values returned from the *GetRelationSet* method of the *Element* object.

<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsDependEnd</td>
<td>List of elements that depend on the current element.</td>
</tr>
<tr>
<td>rsDependStart</td>
<td>List of elements that the current element depends on.</td>
</tr>
<tr>
<td>rsGeneralizeEnd</td>
<td>List of elements that are generalized by the current element.</td>
</tr>
<tr>
<td>rsGeneralizeStart</td>
<td>List of elements that the current element generalizes.</td>
</tr>
<tr>
<td>rsParents</td>
<td>List of all parent elements of the current element.</td>
</tr>
<tr>
<td>rsRealizeEnd</td>
<td>List of elements that are realized by the current element.</td>
</tr>
<tr>
<td>rsRealizeStart</td>
<td>List of elements that the current element realizes.</td>
</tr>
</tbody>
</table>

### 16.7.2.3.3 MDGMenus Enum

Use this enumeration when providing the *HiddenMenus* property to *MDG_GetProperty*.

These options are exclusive of one another and can be read or added to hide more than one menu.

See the *MDG_GetProperty* topic for an example of use.

<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgBuildProject</td>
<td>Hide Build Project menu option.</td>
</tr>
<tr>
<td>mgMerge</td>
<td>Hide Merge menu option.</td>
</tr>
<tr>
<td>mgRun</td>
<td>Hide Run menu option.</td>
</tr>
</tbody>
</table>
16.7.2.3.4 ObjectType Enum

The ObjectType enumeration identifies Enterprise Architect object types even when referenced through a Dispatch interface.

For example:

```cpp
Object ob = Repository.GetElementByID(13);
if ( ob.ObjectType == otElement )
    ;
else if ( ob.ObjectType == otAuthor )
    ...
```

All of the following are valid enumeration values:

- `otNone`
- `otProject`
- `otRepository`
- `otCollection`
- `otElement`
- `otPackage`
- `otModel`
- `otConnector`
- `otDiagram`
- `otRequirement`
- `otScenario`
- `otConstraint`
- `otTaggedValue`
- `otFile`
- `otEffort`
- `otMetric`
- `otIssue`
- `otRisk`
- `otTest`
- `otDiagramObject`
- `otDiagramLink`
- `otResource`
- `otConnectorEnd`
- `otAttribute`
- `otMethod`
- `otParameter`
- `otClient`
- `otAuthor`
- `otDatatype`
- `otStereotype`
- `otTaskOfTerm`
- `otProjectIssues`
- `otAttributeConstraint`
- `otAttributeTag`
- `otMethodConstraint`
- `otMethodTag`
- `otConnectorConstraint`
- `otConnectorTag`
- `otProjectResource`
- `otReference`
- `otRoleTag`
- `atCustomProperty`
- `atPartition`
- `atTransition`
- `atEventProperty`
- `atEventProperties`
- `atPropertyType`
- `atProperties`
- `atProperty`
- `atSwimlaneDef`
- `atSwimlanes`
- `atSwimlane`

16.7.2.3.5 PropType Enum

The PropType enumeration gives the automation programmer an indication of what sort of data is going to be stored by this property.

<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ptArray</code></td>
<td>An array containing values of any type.</td>
</tr>
<tr>
<td><code>ptBoolean</code></td>
<td>True or False.</td>
</tr>
<tr>
<td><code>ptEnum</code></td>
<td>A string being an entry in the semi-colon separated list specified in the validation field of the Property.</td>
</tr>
<tr>
<td><code>ptFloatingPoint</code></td>
<td>4 or 8 byte floating point value.</td>
</tr>
<tr>
<td><code>ptInteger</code></td>
<td>16-bit or 32-bit signed integer.</td>
</tr>
<tr>
<td><code>ptString</code></td>
<td>Unicode string.</td>
</tr>
</tbody>
</table>
16.7.2.3.6 ReloadType Enum

This enumeration represents values returned from the `GetReloadItem` and `PeekReloadItem` methods of the `ModelWatcher` Class. It has four possible values, which define the type of change that was made to a model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rtElement</code></td>
<td>The <code>Item</code> parameter represents a particular element that must be reloaded.</td>
</tr>
<tr>
<td><code>rtEntireModel</code></td>
<td>Entire model must be reloaded to ensure that all changes are reloaded.</td>
</tr>
<tr>
<td><code>rtNone</code></td>
<td>No change in the model.</td>
</tr>
<tr>
<td><code>rtPackage</code></td>
<td>The <code>Item</code> parameter represents a particular package that must be reloaded.</td>
</tr>
</tbody>
</table>

16.7.2.3.7 XMIType Enum

The following enumeration values are used in the `Project.ExportPackageXMI()` method. They enable specification of the XMI export type.

- xmiEADefault
- xmiRoseDefault
- xmiEA10
- xmiEA11
- xmiEA12
- xmiRose10
- xmiRose11
- xmiRose12
- xmiMOF13
- xmiMOF14
- xmiEA20
- xmiEA21

16.7.2.4 Repository

**public Package**

The `Repository` package contains the high level system objects and entry point into the model itself using the `Models` collection and the other system level collections.

This diagram illustrates the `Repository` and its first level functions and collections.
16.7.2.4.1 Repository

public Class

The Repository is the main container of all structures such as models, packages and elements. You can iteratively begin accessing the model using the Models collection. It also has some convenient methods to directly access the structures without having to locate them in the hierarchy first.

Associated table in .EAP file: <none>

Repository Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Collection</td>
<td>Read only. The system Authors collection. Contains 0 or more Author objects, each of which can be associated with, for example, elements or diagrams as the item author or owner. Use AddNew, Delete and GetAt to manage Authors.</td>
</tr>
<tr>
<td>BatchAppend</td>
<td>Boolean</td>
<td>Read/Write. Set this property to true when your automation client has to rapidly insert many elements, operations, attributes and/or operation parameters. Set to false when work is complete.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Clients</td>
<td>Collection</td>
<td>Read only. A list of Clients associated with the project. You can modify, delete and add new Client objects using this collection.</td>
</tr>
<tr>
<td>ConnectionString</td>
<td>String</td>
<td>Read only. The filename/connection string of the current Repository.</td>
</tr>
<tr>
<td>Datatypes</td>
<td>Collection</td>
<td>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.</td>
</tr>
<tr>
<td>EnableCache</td>
<td>Boolean</td>
<td>Read/Write: An optimization for pre-loading package objects when dealing with large sets of automation objects.</td>
</tr>
<tr>
<td>EnableUIUpdates</td>
<td>Boolean</td>
<td>Read/Write. Set this property to false to improve the performance of changes to the model; e.g. bulk addition of elements to a package. To reveal the changes to the user, call Repository.RefreshModelView().</td>
</tr>
<tr>
<td>FlagUpdate</td>
<td>Boolean</td>
<td>Read/Write. Instructs Enterprise Architect to update the Repository with the LastUpdate value.</td>
</tr>
<tr>
<td>InstanceGUID</td>
<td>String</td>
<td>Read only: The identifier string identifying the Enterprise Architect runtime session.</td>
</tr>
<tr>
<td>Issues</td>
<td>Collection</td>
<td>Read only. The System Issues list. Contains ProjectIssues objects, each detailing a particular issue as it relates to the project as a whole.</td>
</tr>
<tr>
<td>LastUpdate</td>
<td>String</td>
<td>Read only. The identifier string identifying the Enterprise Architect runtime session and the timestamp for when it was set.</td>
</tr>
<tr>
<td>LibraryVersion</td>
<td>Long</td>
<td>Read only. The build number of the Enterprise Architect runtime.</td>
</tr>
<tr>
<td>Models</td>
<td>Collection</td>
<td>Read only. Models are of type package and belong to a collection of packages. This is the top level entry point to an Enterprise Architect project file. Each model is a root node in the Project Browser and can contain items such as Views and packages. 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 can also use the AddNew function to create a new model. A model can be deleted, but remember that everything contained in the model is deleted as well.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through the Dispatch interface.</td>
</tr>
<tr>
<td>ProjectGUID</td>
<td>String</td>
<td>Read only: Returns a unique ID for the project.</td>
</tr>
<tr>
<td>PropertyTypes</td>
<td>Collection</td>
<td>Read only. Collection of Property Types available to the Repository.</td>
</tr>
<tr>
<td>Resources</td>
<td>Collection</td>
<td>Read only. Contains available ProjectResource objects to assign to work items within the project. Use the add new, modify and delete functions to manage resources.</td>
</tr>
<tr>
<td>Stereotypes</td>
<td>Collection</td>
<td>Read only. The Stereotypes collection. A list of Stereotype objects that contain information on a stereotype and which elements it can be applied to.</td>
</tr>
<tr>
<td>SuppressEADialogs</td>
<td>Boolean</td>
<td>Read/Write. Set this property in the EA_OnPostNewElement or EA_OnPostNewConnector broadcast events to control whether Enterprise Architect should suppress</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td>showing the default Properties dialogs to the user when an element or connector is created.</td>
</tr>
<tr>
<td>Tasks</td>
<td>Collection</td>
<td>Read only. A list of system tasks (to do list). Each entry is a Task Item; you can modify, delete and add new tasks.</td>
</tr>
<tr>
<td>Terms</td>
<td>Collection</td>
<td>Read only. The project Glossary. Each Term object is an entry in the Glossary. Add, modify and delete Terms to maintain the Glossary.</td>
</tr>
</tbody>
</table>

**Repository Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| ActivateDiagram (long DiagramID)   |          | Activates an already open diagram (i.e. makes it the active tab) in the main Enterprise Architect user interface. Parameters:  
  - DiagramID: Long - the ID of the diagram to make active. |
| ActivatePerspective (string, long) | Boolean  | Deprecated - no longer in use. |
| ActivateTab (string Name)          |          | Activates an open Enterprise Architect tabbed view. Parameters:  
  - Name: String - the name of the view to activate. |
| ActivateToolbox (string Toolbox, long Options) | Boolean | Activates a Toolbox page in the GUI. Parameters:  
  - Toolbox: String - the name of the Toolbox page to activate.  
  - Options: Long - reserved for future use. Returns true if the specified Toolbox page is successfully activated, otherwise returns false. |
| AddDefinedSearches (string sXML)   |          | Enables you to enter a set of defined searches that last in Enterprise Architect for the life of the application. When Enterprise Architect loads again they must be inserted again by your Add-In. Parameters:  
  - sXML: String - the XML of the defined searches; you can get this XML by performing an export of the searches from the Manage Searches dialog in Enterprise Architect. |
| AddPerspective (string Perspective, long Options) | Boolean | Deprecated - no longer in use. |
| AddTab (string TabName, string ControlID) | activeX custom control | Adds an ActiveX custom control as a tabbed window. Enterprise Architect creates a control and, if successful, returns its Unknown pointer, which can be used by the caller to manipulate the control. Parameters:  
  - TabName: String - used as the tab caption.  
  - ControlID: String - the ProgID of the control. e.g. Project1 UserControl1. |
<p>| AdviseConnectorChange (long ConnectorID) |          | Provides an Add-In or automation client with the ability to advise the Enterprise Architect 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. Parameters: |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdviseElementChange (long ObjectID)</td>
<td></td>
<td>Provides an Add-In or automation client with the ability to advise the Enterprise Architect 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. Parameters:</td>
</tr>
<tr>
<td>ChangeLoginUser (string Name, string Password)</td>
<td>Boolean</td>
<td>Sets the currently logged on user to be that specified by a name and password. This logs the user into the repository when security is enabled. If security is not enabled an exception (Security not enabled) is thrown. Parameters:</td>
</tr>
<tr>
<td>ClearAuditLogs (Object StartDateTime, Object EndDateTime)</td>
<td>Boolean</td>
<td>Clears all Audit Logs from the model. Parameters:</td>
</tr>
<tr>
<td>ClearOutput (string Name)</td>
<td></td>
<td>Removes all the text from a tab in the Output window. Parameters:</td>
</tr>
<tr>
<td>CloseAddins ()</td>
<td></td>
<td>Called by automation controllers to ensure that Add-Ins created in .NET do not linger after all controller references to Enterprise Architect have been cleared.</td>
</tr>
<tr>
<td>CloseDiagram (long DiagramID)</td>
<td></td>
<td>Closes a diagram in the current list of diagrams that Enterprise Architect has open. Parameters:</td>
</tr>
<tr>
<td>CloseFile ()</td>
<td></td>
<td>Closes any open file.</td>
</tr>
<tr>
<td>CreateOutputTab (string Name)</td>
<td></td>
<td>Creates a tab in the Output window. Parameters:</td>
</tr>
<tr>
<td>DeletePerspective (string Perspective, long Options)</td>
<td>Boolean</td>
<td>Deprecated - no longer in use.</td>
</tr>
<tr>
<td>DeleteTechnology (string ID)</td>
<td>Boolean</td>
<td>Removes a specified MDG Technology resource from the repository. Parameters:</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>ID: String - the ID of the technology.</td>
<td></td>
<td>Returns true, if the technology is successfully removed from the model. Returns false otherwise.</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>This applies to technologies imported into pre-7.0 versions of Enterprise Architect (imported technologies), not to technologies referenced in version 7.0 and later (referenced technologies). See Deploying MDG Technologies (from Add-Ins).</td>
</tr>
<tr>
<td>EnsureOutputVisible (string Name)</td>
<td></td>
<td>Ensures that a specified tab in the Output window is visible to the user. The Output window is made visible if it is hidden. Parameters: Name: String - the name of the tab to make visible.</td>
</tr>
<tr>
<td>ExecutePackageBuildScript (long ScriptOptions, string PackageGuid)</td>
<td></td>
<td>Enables 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. Parameters: ScriptOptions: Long - the script type; can be any one of these numerical values: 1 = Build 2 = Test 3 = Run 4 = Create Workbench Instance 5 = Debug. PackageGuid: String - the ID of the package for which to run the script.</td>
</tr>
<tr>
<td>Exit</td>
<td></td>
<td>Shut down Enterprise Architect immediately. Used by DotNET programmers where the garbage collector does not immediately release all referenced COM objects.</td>
</tr>
<tr>
<td>GetActivePerspective ()</td>
<td>String</td>
<td>Deprecated - no longer in use.</td>
</tr>
<tr>
<td>GetAttributeByGuid (string Guid)</td>
<td>Attribute</td>
<td>Returns a pointer to an attribute in the repository, located by its GUID. This is usually found using the AttributeGUID property of an attribute. Parameters: Guid: String - the GUID of the attribute to locate.</td>
</tr>
<tr>
<td>GetAttributeById (string Id)</td>
<td>Attribute</td>
<td>Returns a pointer to an attribute in the repository, located by its ID. This is usually found using the AttributeID property of an attribute. Parameters: Id: String - the ID of the attribute to locate.</td>
</tr>
<tr>
<td>GetConnectorByGuid (string Guid)</td>
<td>Connector</td>
<td>Returns a pointer to a connector in the repository, located by its GUID. This is usually found using the ConnectorGUID property of a connector. Parameters: Guid: String - the GUID of the connector to locate.</td>
</tr>
<tr>
<td>GetConnectorById (long ConnectorId)</td>
<td>Connector</td>
<td>Searches the repository for a connector with a specific ID. Parameters: ConnectorId: Long - the ID of the connector to locate.</td>
</tr>
<tr>
<td>GetContextItem (object Item)</td>
<td>ObjectType</td>
<td>Sets a pointer to an item in context within Enterprise Architect.</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| GetContextItemType ()        | ObjectType      | Returns the ObjectType of an item in context within Enterprise Architect. A ContextItem is defined as an item selected anywhere within the Enterprise Architect GUI including:  
  - An item selected in the Project Browser  
  - An item selected in an open diagram  
  - An item selected in certain dialogs, such as the attribute Properties dialog.  
  The supported ObjectTypes can be any one of the following values:  
  - otElement  
  - otPackage  
  - otDiagram  
  - otAttribute  
  - otMethod  
  - otConnector                                                                                   |
| GetContextObject ()          | Object          | Returns the current context Object.                                                                                                                                                                  |
| GetCounts ()                 | String          | Returns a set of counts from a number of tables within the base Enterprise Architect repository. These can be used to determine whether records have been added or deleted from the tables for which information is retrieved.             |
| GetCurrentDiagram ()         | Diagram         | Returns a selected diagram.                                                                                                                                                                        |
| GetCurrentUser (GetGuid = false) | String       | If security is not enabled in the repository, an error is generated. If GetGuid is True, a GUID generated by Enterprise Architect representing the user is returned; otherwise the text as entered in System Users/User Details/Login is returned. |
| GetDiagramByGuid (string Guid) | Diagram     | 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 to open an diagram without using the collection GetAt() function.  
  Parameters:  
  - Guid: String - the GUID of the diagram to locate.                                             |
| GetDiagramByID (DiagramID)  | Diagram         | Gets 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 to open a diagram without using the collection GetAt() function.  
  Parameters:  
  - DiagramID: Long - the ID of the diagram to locate.                                             |
| GetElementByGuid (string Guid) | Element     | 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 to open an element without using the collection GetAt() function.  
  Parameters:                                                                                      |
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetElementById (long ElementID)</td>
<td>Element</td>
<td>Gets a pointer to an element using an absolute reference number (local ID). This is usually found using the <code>ElementID</code> property of an element, and stored for later use to open an element without using the collection <code>GetAt()</code> function. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· ElementID: Long - the ID of the element to locate.</td>
</tr>
<tr>
<td>GetElementsByQuery (string QueryName, string SearchTerm)</td>
<td>Collection</td>
<td>Enables you to run a search in Enterprise Architect, returning the result as a collection. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· QueryName: String - the name of the search to run, for example 'Simple'.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· SearchTerm: String - the term to search for. For example <code>GetElementsByQuery('Simple','Class1')</code>, where results contain elements with <code>Class1</code> in the Name and Notes fields.</td>
</tr>
<tr>
<td>GetElementSet ()</td>
<td>Collection</td>
<td>Returns a set of elements as a collection based on an input of element ID numbers in comma separated form. For example: <code>GetElementSet(&quot;34,56,21,5&quot;)</code>.</td>
</tr>
<tr>
<td>GetFieldFromFormat (string Format, string Text)</td>
<td>String</td>
<td>Converts a field from your preferred format to Enterprise Architect's internal format. Returns the field in Enterprise Architect's internal format. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Format: String - The format to convert the field from. Valid formats are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· HTML - Full HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· RTF - Rich Text Format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· TXT - Plain text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Text: String - The field to be converted.</td>
</tr>
<tr>
<td>GetFormatFromField (string Format, string Text)</td>
<td>String</td>
<td>After accessing a field that contains formatting, use this method to convert it to your preferred format. Returns the field in the format specified. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Format: String - The format to convert the field to. Valid formats are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· HTML - Full HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· RTF - Rich Text Format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· TXT - Plain text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Text: String - The field to be converted.</td>
</tr>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>GetMethodByGuid (string Guid)</td>
<td>Method</td>
<td>Returns a pointer to a method in the repository. This is usually found using the <code>MethodGUID</code> property of a method. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Guid: String - the GUID of the method to look for.</td>
</tr>
<tr>
<td>GetMethodById (string Id)</td>
<td>Method</td>
<td>Returns a pointer to a method in the repository. This is usually found using the <code>MethodID</code> property of a method. Parameters:</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| GetPackageByGuid (string Guid) | Package | Returns a pointer to a package in the repository using the package's GUID reference number (global ID). This is usually found using the PackageGUID property of the package. Parameters:  
  - Guid: String - the GUID of the package to look for. Each package in the model also has an associated element with the same GUID, so if you have an element with Type="Package" then you can load the package by calling: GetPackageByGuid(Element.ElementGUID). |
| GetPackageByID (long PackageID) | Package | Get a pointer to a package using an absolute reference number (local ID). This is usually found using the PackageID property of an package, and stored for later use to open a package without using the collection GetAt() function. Parameters:  
  - PackageID: Long - the ID of the package to locate. |
| GetProjectInterface () | Project | Return a pointer to the EA.Project interface (the XML-based automation server for Enterprise Architect). Use this interface to work with Enterprise Architect using XML, and also to access utility functions for loading diagrams, running reports and so on. |
| GetReferenceList (String Type) | Reference | Uses the list type to get a pointer to a Reference List object. Parameters:  
  - Type: String - specifies the list type to get; valid list types are:  
    - Diagram  
    - Element  
    - Constraint  
    - Requirement  
    - Connector  
    - Status  
    - Cardinality  
    - Effort  
    - Metric  
    - Scenario  
    - Status and  
    - Test. |
| GetTechnologyVersion (stringID) | String | Returns the version of a specified MDG Technology resource. Parameters:  
  - ID: String - the specified technology ID. |
| GetTreeSelectedItem (ObjectSelectedItem) | ObjectType | Gets 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. The object passed back through the parameter can be a package, element, diagram, attribute or operation object. Parameters:  
  - SelectedItem: Object - the object to get the variable and type for. |
<p>| GetTreeSelectedItemType () | ObjectType | Returns the type of the object currently selected in the tree. One of: |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetTreeSelectedObject ()</td>
<td>Object</td>
<td>The related method <code>GetTreeSelectedObject()</code> has an output parameter that is inaccessible by some scripting languages. As an alternative, this method provides the selected item through the return value.</td>
</tr>
<tr>
<td>GetTreeSelectedPackage ()</td>
<td>Package</td>
<td>Returns the package in which the currently selected tree view object is contained.</td>
</tr>
<tr>
<td>HasPerspective (string Perspective)</td>
<td>String</td>
<td>Deprecated - no longer in use.</td>
</tr>
<tr>
<td>ImportPackageBuildScripts (string PackageGuid, string BuildScriptXML)</td>
<td></td>
<td>Imports build scripts into a package in Enterprise Architect. Parameters: PackageGuid: String - the GUID of the package into which to import the build scripts. BuildScriptXML: String - the build script XML data, which you can export from within Enterprise Architect.</td>
</tr>
<tr>
<td>ImportTechnology (string Technology)</td>
<td>Boolean</td>
<td>Installs a given MDG Technology resource into the repository. Parameters: Technology: String - the contents of the technology resource file. Returns True, if the technology is successfully loaded into the model. Otherwise returns False. Note: This applies to technologies imported into pre-7.0 versions of Enterprise Architect (imported technologies), not to technologies referenced in version 7.0 and later (referenced technologies). See Deploying MDG Technologies (from Add-Ins).</td>
</tr>
<tr>
<td>IsTabOpen ()</td>
<td>String</td>
<td>Checks whether a named tab is open and active. Parameters: TabName: String - the name of the tab to check for. Returns: 2 to indicate that a tab is open and active (top-most) 1 to indicate that it is open but not top-most, or 0 to indicate that it is not visible at all. Note: TabName is case-sensitive.</td>
</tr>
<tr>
<td>IsTechnologyEnabled (string ID)</td>
<td>Boolean</td>
<td>Checks whether a specified technology is enabled in Enterprise Architect. Parameters: ID: String - the technology ID to check for. Returns True if the MDG Technology resource is enabled. Otherwise returns False.</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| IsTechnologyLoaded (string ID) | Boolean | Checks whether a specified technology is loaded into the repository. Parameters:  
- ID: String - the technology ID to check for. Returns True if the MDG Technology resource is loaded into the repository. Otherwise returns False. |
| OpenDiagram (long DiagramID) | | Provides a method for an automation client or Add-In to open a diagram. The diagram is added to the tabbed list of open diagrams in the main Enterprise Architect view. Parameters:  
- DiagramID: Long - the ID of the diagram to open. |
| OpenFile (string Filename) | Boolean | This is the main point for opening an Enterprise Architect project file from an automation client, and working with the contained objects. Parameters:  
- Filename: String - the filename of the Enterprise Architect project to open. If the required project is a DBMS repository, and you have created a shortcut .EAP file containing the database connection string, you can call this shortcut file to access the DBMS repository. You can also connect to a SQL database by passing in the connection string itself instead of a filename. A valid connection string can be obtained from the Open Project dialog by selecting a recently opened SQL repository. |
| OpenFile2 (string FilePath, string Username, string Password) | Boolean | As for OpenFile() except this enables the specification of a password. |
| RefreshModelView (long PackageID) | | Reloads a package or the entire model, updating the user interface. Parameters:  
- PackageID: Long - the ID of the package to reload: if 0, the entire model is reloaded; if a valid package ID, only that package is reloaded. |
| RefreshOpenDiagrams (boolean FullReload) | | Refreshes the diagram contents for all diagrams open in Enterprise Architect. Parameters:  
- FullReload: Boolean - if false the displayed contents of elements and connectors are refreshed in each diagram; if true each of the diagrams is completely reloaded from the repository. |
| ReloadDiagram (long DiagramID) | | Reloads a specified diagram. This would commonly be used to refresh a visible diagram after code import/export or other batch process where the diagram requires complete refreshing. Parameters:  
- DiagramID: Long - the ID of the diagram to be reloaded. |
| RemoveOutputTab (string Name) | | Removes a specified tab from the Output window. Parameters:  
- Name: String - the name of the tab to be removed. |
<p>| RunModelSearch (string sQueryName, string) | | Runs a search, displaying the results in Enterprise Architect's Model Search window. |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| sSearchTerm, string sSearchOptions, string sSearchData) | | Parameters:  
  - sQueryName: String - the name of the search to run, for example Simple.  
  - sSearchTerm: String - the term to search for.  
  - sSearchOptions: String - currently not being used.  
  - sSearchData: String - enables you to supply a list of results in the form of XML, which is appended onto the result list in Enterprise Architect. See XML Format; this parameter is not mandatory so pass in an empty string to run the search as per normal. |
| SaveAllDiagrams () | | Saves all open diagrams. |
| SaveAuditLogs (string FilePath, object StartDateTime, object EndDateTime) | Boolean | Saves the Audit Logs contained within a model to a specified file.  
Parameters:  
  - FilePath: String - the file to save the Audit Logs to.  
  - StartDateTime: Variant [DateTime] - the earliest date and time of log entries to save.  
  - EndDateTime: Variant [DateTime] - the latest date and time of log entries to save.  
  
If StartDateTime and EndDateTime are not null then only log items that fall into this period are saved.  
Returns true for success, false for failure.  

**Note:**  
This might fail if the user logged into the model does not have the correct access permission. |
| SaveDiagram (long DiagramID) | | Saves an open diagram. Assumes the diagram is open in the main user interface Tab list.  
Parameters:  
  - DiagramID: Long - the ID of the diagram to save. |
| ShowDynamicHelp (string Topic) | | Shows a help topic as a view.  
Parameters:  
  - Topic: String - specifies the help topic. |
| ShowInProjectView (Object Item) | | Selects a specified object in the Project Browser.  
Parameters:  
  - Item: Object - the object to highlight.  
Accepted object types are Package, Element, Diagram, Attribute, and Method. An exception is thrown if the object is of an invalid type. |
| ShowProfileToolbox (string Technology, string Profile, boolean Show) | | Shows/hides the contents of a specified technology or profile in the Enterprise Architect UML Toolbox.  
Parameters:  
  - Technology: String - the ID of the technology.  
  - Profile: String - the ID of the profile.  
  - Show: Boolean - if true, show the technology or profile; if false, hide the technology or profile.  
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
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| ShowWindow (long Show) |      | Shows or hides Enterprise Architect. Parameters:  
|                      |      | • Show: Long.                                                                                                                          |
| SQLQuery (string SQL) | String | Enables execution of a SQL select statement against the current repository. Returns an XML formatted string value of the resulting recordset. Parameters:  
|                      |      | • SQL: String - contains the SQL Select statement.                                                                                     |
| WriteOutput (string Name, string String, long ID) |      | Writes text to a specified tab in the Output window, and associates the text with an ID. Parameters:  
|                      |      | • Name: String - specifies the tab on which to display the text.  
|                      |      | • String: String - specifies the text to display.  
|                      |      | • ID: Long - specifies the ID the text is associated with.                                                                             |

#### 16.7.2.4.2 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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. Author name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about the author.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Roles</td>
<td>String</td>
<td>Read/Write. Roles the author might play in this project.</td>
</tr>
</tbody>
</table>

**Author Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Updates the current Author object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

#### 16.7.2.4.3 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>String</td>
<td>Read/Write. Email address.</td>
</tr>
<tr>
<td>Fax</td>
<td>String</td>
<td>Read/Write. Fax number.</td>
</tr>
<tr>
<td>Mobile</td>
<td>String</td>
<td>Read/Write. Mobile phone if available.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. Client name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about client.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through the Dispatch interface.</td>
</tr>
<tr>
<td>Organization</td>
<td>String</td>
<td>Read/Write. Associated organization.</td>
</tr>
<tr>
<td>Phone1</td>
<td>String</td>
<td>Read/Write. Main phone number.</td>
</tr>
<tr>
<td>Phone2</td>
<td>String</td>
<td>Read/Write. Second phone number.</td>
</tr>
<tr>
<td>Roles</td>
<td>String</td>
<td>Read/Write. Roles this client might play in the project.</td>
</tr>
</tbody>
</table>

Client Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Updates the current Client object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.4.4 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 still contains a reference to the deleted item, which should not be called.

Each can be used to iterate through the collection for languages that support this type of construct.

Collection Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Short</td>
<td>Read only. The number of objects referenced by this list.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>
### Collection Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| AddNew (string Name, string Type)          | Object   | Adds 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 depends 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. Parameters:  
|                                             |          | Name: String  
|                                             |          | Type: String (up to 30 characters long)                             |
| Delete (short Index)                        | Void     | Deletes the item at the selected reference. Parameters:  
|                                             |          | Index: Short                                                      |
| DeleteAt (short Index, boolean Refresh)     | Void     | Deletes the item at the selected index. The second parameter is currently unused. Parameters:  
|                                             |          | Index: Short  
|                                             |          | Refresh: Boolean                                                  |
| GetAt (short Index)                         | Object   | Retrieves the array object using a numerical index. If the index is out of bounds, an error occurs. Parameters:  
|                                             |          | Index: Short                                                      |
| GetByName (string Name)                     | Object   | Gets an item in the current collection by Name. If the collection does not contain any items, the method returns a null value. If the collection contains items, but it was unable to find an object with the specified name, the method raises an exception. Only supported for the following collections: Models, Packages, Elements, Diagrams, TaggedValues. Parameters:  
|                                             |          | Name: String                                                      |
| GetLastError ()                             | String   | Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.  
|                                             |          | Name: String                                                      |
| Refresh ()                                  | Void     | Refreshes the collection by re-querying the model and reloading the collection. Should be called after adding a new item or after deleting an item.  

#### 16.7.2.4.5 Datatype

**public Class**

A *Datatype* is a named type that can 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatatypeID</td>
<td>Long</td>
<td>Read/Write. Instance ID for this datatype within the current model. System maintained.</td>
</tr>
<tr>
<td>DefaultLen</td>
<td>Long</td>
<td>Read/Write. Default length (DDL only.)</td>
</tr>
<tr>
<td>DefaultPrec</td>
<td>Long</td>
<td>Read/Write. Default precision (DDL only).</td>
</tr>
<tr>
<td>DefaultScale</td>
<td>Long</td>
<td>Read/Write. Default scale (DDL only).</td>
</tr>
<tr>
<td>GenericType</td>
<td>String</td>
<td>Read/Write. The associated generic type for this data type.</td>
</tr>
<tr>
<td>HasLength</td>
<td>String</td>
<td>Read/Write. Indicates datatype has a length component.</td>
</tr>
<tr>
<td>MaxLen</td>
<td>Long</td>
<td>Read/Write. Maximum length (DDL only).</td>
</tr>
<tr>
<td>MaxPrec</td>
<td>Long</td>
<td>Read/Write. Maximum precision (DDL only).</td>
</tr>
<tr>
<td>MaxScale</td>
<td>Long</td>
<td>Read/Write. Maximum scale (DDL only).</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The datatype name (e.g. integer). This appears in the related drop-down datatype lists where appropriate.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Product</td>
<td>String</td>
<td>Read/Write. The datatype product - e.g. Java, C++, Oracle.</td>
</tr>
<tr>
<td>Size</td>
<td>Long</td>
<td>Read/Write. The datatype size.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The type can be DDL for database datatype or Code for language datatypes.</td>
</tr>
<tr>
<td>UserDefined</td>
<td>Long</td>
<td>Read/Write. Indicates if datatype is a user defined type or system generated. Datatypes distributed with Enterprise Architect are all system generated. Datatypes created in the Datatype dialog are marked 1 (true).</td>
</tr>
</tbody>
</table>

Datatype Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Updates the current Datatype object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.4.6 EventProperties

An EventProperties object is passed to BroadcastFunctions to facilitate parameter passing.

EventProperties Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Read only. Number of parameters being passed to this broadcast event.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>
EventProperties Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get(objectIndex)</td>
<td>EventProperty</td>
<td>Read only. Returns an EventProperty in the list, raising an error if Index is out of range. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Index: Variant - can either be a number representing a zero-based index into the array, or a string representing the name of the EventProperty, e.g. Props.Get(3) or Props.Get(&quot;ObjectID&quot;).</td>
</tr>
</tbody>
</table>

16.7.2.4.7 EventProperty

EventProperty objects are always part of an EventProperties collection, and are passed to Add-In methods responding to broadcast events.

EventProperty Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>String</td>
<td>Explanation of what this property represents.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>A string distinguishing this property from others in the list.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Value</td>
<td>Variant</td>
<td>A string, number or object reference representing the property value.</td>
</tr>
</tbody>
</table>

16.7.2.4.8 ModelWatcher

public Class

The ModelWatcher object enables an automation client to track changes in a particular model.

Note:

After your model has been loaded, you only create the ModelWatcher once. If you reload the model, or load another model, the created ModelWatcher is still valid.

ModelWatcher Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>

ModelWatcher Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetReloadItem (object Index)</td>
<td>ReloadType</td>
<td>The object that must be reloaded in order to see all changes is returned through the Item parameter. If there are no changes or the entire model must be reloaded, this value is returned as null (C#) or Nothing (VB). Calling this method clears the records so that the next time it is called the return values refer only to new changes. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Item: Object</td>
</tr>
<tr>
<td>PeekReloadItem</td>
<td>ReloadType</td>
<td>This method behaves identically to GetReloadItem() but does not clear the</td>
</tr>
</tbody>
</table>
16.7.2.4.9 Package

public Class

A Package object corresponds to a Package element in the Enterprise Architect 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 Enterprise Architect 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>String</td>
<td>Read only. Alias.</td>
</tr>
<tr>
<td>BatchLoad</td>
<td>Long</td>
<td>Read/Write. Flag to indicate that the package is batch loaded during batch import from controlled packages. Not yet implemented.</td>
</tr>
<tr>
<td>BatchSave</td>
<td>Long</td>
<td>Read/Write. Boolean value to indicate whether the package is included in the batch XML export list or not.</td>
</tr>
<tr>
<td>CodePath</td>
<td>String</td>
<td>Read/Write. The path to where associated source code is found. Not currently used.</td>
</tr>
<tr>
<td>Connectors</td>
<td>Collection</td>
<td>Read only. Collection of connectors.</td>
</tr>
<tr>
<td>Created</td>
<td>Date</td>
<td>Read/Write. Date the package was created.</td>
</tr>
<tr>
<td>Diagrams</td>
<td>Collection</td>
<td>Read only. A collection of diagrams contained in this package.</td>
</tr>
<tr>
<td>Element</td>
<td>Object</td>
<td>Read only. The associated element object. Use to set element type information for a package, including Stereotype, Complexity, Alias, Author, Constraints and Scenarios.</td>
</tr>
<tr>
<td>Elements</td>
<td>Collection</td>
<td>Read only. A collection of elements that belong to this package.</td>
</tr>
<tr>
<td>Flags</td>
<td>String</td>
<td>Read/Write. Extended information about the package.</td>
</tr>
<tr>
<td>IsControlled</td>
<td>Boolean</td>
<td>Read/Write. Indicates if the package has been marked as Controlled.</td>
</tr>
<tr>
<td>IsModel</td>
<td>Boolean</td>
<td>Read only. Indicates if the package is a model or a package.</td>
</tr>
<tr>
<td>IsNamespace</td>
<td>Boolean</td>
<td>Read/Write. True is 'package is a Namespace root'. Use 0 and 1 to set False and True.</td>
</tr>
<tr>
<td>IsProtected</td>
<td>Boolean</td>
<td>Read/Write. Indicates if the package has been marked as Protected.</td>
</tr>
<tr>
<td>IsVersionControlled</td>
<td>Boolean</td>
<td>Read. Indicates whether or not this package is under version control.</td>
</tr>
<tr>
<td>LastLoadDate</td>
<td>Date</td>
<td>Read/Write. The date XML was last loaded for the package.</td>
</tr>
<tr>
<td>LastSaveDate</td>
<td>Date</td>
<td>Read/Write. The date XML was last saved from the package.</td>
</tr>
<tr>
<td>LogXML</td>
<td>Boolean</td>
<td>Read/Write. Indicates if XMI export information is to be logged.</td>
</tr>
<tr>
<td>Modified</td>
<td>Date</td>
<td>Read/Write. Date the package was last modified.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The name of the package.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about this package.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Owner</td>
<td>String</td>
<td>Read/Write. The package owner when using controlled packages.</td>
</tr>
<tr>
<td>PackageGUID</td>
<td>Variant</td>
<td>Read only. The global Package ID. Valid across models.</td>
</tr>
<tr>
<td>PackageID</td>
<td>Long</td>
<td>Read only. The local Package ID number. Valid only in this model file.</td>
</tr>
<tr>
<td>Packages</td>
<td>Collection</td>
<td>Read only. A collection of contained packages that can be walked through.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read/Write. The ID of the package that is the parent of this one. 0 indicates this package is a model (i.e. it has no parent).</td>
</tr>
<tr>
<td>TreePos</td>
<td>Long</td>
<td>Read/Write. The relative position in the tree compared to other packages (use to sort packages).</td>
</tr>
<tr>
<td>UMLVersion</td>
<td>String</td>
<td>Read/Write. The UML version for XMI export purposes.</td>
</tr>
<tr>
<td>UseDTD</td>
<td>Boolean</td>
<td>Read/Write. Indicates if a DTD is to be used when exporting XMI.</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>Read/Write. The version of the package.</td>
</tr>
<tr>
<td>XMLPath</td>
<td>String</td>
<td>Read/Write. The path to where the XML is saved when using controlled packages.</td>
</tr>
</tbody>
</table>

### Package Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyGroupLock (string aGroupName)</td>
<td>Boolean</td>
<td>Applies a group lock to the package object, for the specified group, on behalf of the current user. Throws an exception if the operation fails. Use GetLastError() to retrieve error information. Parameter:    - aGroupName: String - The name of the security group for which to apply the lock.</td>
</tr>
<tr>
<td>ApplyUserLock ()</td>
<td>Boolean</td>
<td>Applies a user lock to the package object for the current user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</td>
</tr>
<tr>
<td>Clone</td>
<td>LDISPATCH</td>
<td>Inserts a copy of the package into the same parent as the original package. Returns the newly-created package.</td>
</tr>
<tr>
<td>FindObject (string DottedID)</td>
<td>LPDISPATCH</td>
<td>Returns a package, element, attribute or operation matching the parameter DottedID. Parameter:    - DottedID: String - Is in the form object.object.object where object is replaced by the name of a package, element attribute or operation. Examples include MyNamespace.Class1, CStudent.m_Name, MathClass.DoubleIt(int) If the DottedID is not found, an error is returned: Can’t find matching object.</td>
</tr>
<tr>
<td>GetLastErrror ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object.</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>ReleaseLock ()</td>
<td>Boolean</td>
<td>Removes an existing User or Group lock from the package object. Throws an exception if the operation fails. Use GetLastError( ) to retrieve error information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update the current package object after modification or appending a new item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If false is returned, check the GetLastError function for more information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that a package object also has an element component that must be taken into account. The package object contains information about the package attributes such as hierarchy or contents. The element attribute contains information about, for example, Stereotype, Constraints or Files - all the attributes of a typical element.</td>
</tr>
<tr>
<td>VersionControlAdd</td>
<td>Void</td>
<td>Places the package under version control, using the specified Version Control Configuration and the specified XMI filename. Throws an exception if the operation fails. Use GetLastError( ) to retrieve error information.</td>
</tr>
<tr>
<td>(string ConfigGuid,</td>
<td></td>
<td>It is recommended that the package be saved using Update() before calling VersionControlAdd(), so that any outstanding changes are not lost.</td>
</tr>
<tr>
<td>string XMLFile, bool</td>
<td></td>
<td>KeepCheckedOut)</td>
</tr>
<tr>
<td>VersionControlCheckin</td>
<td>Void</td>
<td>Perform checkin of the version controlled package. Throws an exception if the operation fails. Use GetLastError( ) to retrieve error information.</td>
</tr>
<tr>
<td>(string Comment)</td>
<td></td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· ConfigGuid: String - Name corresponding to the Unique ID of the version control configuration to use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· XMLFile: String - Name of the XML file to use for this package. This filename is relative to the Working Copy folder specified for the Config.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· KeepCheckedOut: Boolean - Specify True to add to version control and keep package checked-out.</td>
</tr>
<tr>
<td>VersionControlCheckout</td>
<td>Void</td>
<td>Perform checkout of the version controlled package. Throws an exception if the operation fails. Use GetLastError( ) to retrieve error information.</td>
</tr>
<tr>
<td>(string Comment)</td>
<td></td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Comment: String - Log message that is added to the version controlled file's history (where applicable).</td>
</tr>
<tr>
<td>VersionControlGetStatus</td>
<td>Long</td>
<td>Returns the version control status of the package. Throws an exception if the operation fails. Use GetLastError( ) to retrieve error information.</td>
</tr>
<tr>
<td>()</td>
<td></td>
<td>Return value maps to the following enumerated type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enum EnumCheckOutStatus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· csUncontrolled = 0,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· csCheckedIn,</td>
</tr>
</tbody>
</table>
Method | Type | Notes
--- | --- | ---
| | | 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 Enterprise Architect can not currently connect to the version control config and the package was not previously checked out to this user.

csDeleted - The package file has been deleted from version control.

VersionControlRemove () Void Removes version control from the package.

Throws an exception if the operation fails. Use GetLastError () to retrieve error information.

16.7.2.4.10 ProjectIssues

public Class


Associated table in .EAP file: t_issues

ProjectIssues Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>String</td>
<td>Read/Write. The category this issue belongs to.</td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
<td>Read/Write. Date created.</td>
</tr>
<tr>
<td>DateResolved</td>
<td>Date</td>
<td>Read/Write. Date issue resolved.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. Issue name (i.e. the issue itself).</td>
</tr>
<tr>
<td>IssueID</td>
<td>Long</td>
<td>Read only. The ID of this issue.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Associated description of issue.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Owner</td>
<td>String</td>
<td>Read/Write. Owner of issue.</td>
</tr>
<tr>
<td>Priority</td>
<td>String</td>
<td>Read/Write. Issue priority. Generally should use Low, Medium or High.</td>
</tr>
<tr>
<td>Resolution</td>
<td>String</td>
<td>Read/Write. Description of resolution.</td>
</tr>
<tr>
<td>Resolver</td>
<td>String</td>
<td>Read/Write. Person resolving issue.</td>
</tr>
</tbody>
</table>
### ProjectIssues Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Issue object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.4.11 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>String</td>
<td>Email address.</td>
</tr>
<tr>
<td>Fax</td>
<td>String</td>
<td>Fax number.</td>
</tr>
<tr>
<td>Mobile</td>
<td>Variant</td>
<td>Mobile number if available.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Name of resource.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>A description if appropriate.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Organization</td>
<td>Package, String</td>
<td>Organization resource associated with.</td>
</tr>
<tr>
<td>Phone1</td>
<td>Variant</td>
<td>Main phone.</td>
</tr>
<tr>
<td>Phone2</td>
<td>Variant</td>
<td>Alternative phone.</td>
</tr>
<tr>
<td>Roles</td>
<td>String</td>
<td>The roles this resource can play in the current project.</td>
</tr>
</tbody>
</table>

### ProjectResource Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Resource object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>
16.7.2.4.12 PropertyType

public Class

A PropertyType object represents a defined property that can 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 for the model.

Associated table in .EAP file: t_propertytypes

Author Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>String</td>
<td>Read/Write. Short description for the property.</td>
</tr>
<tr>
<td>Detail</td>
<td>String</td>
<td>Read/Write. Configuration information for the property.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Tag</td>
<td>String</td>
<td>Read/Write. Name of the property (Tag Name).</td>
</tr>
</tbody>
</table>

Author Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current PropertyType object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.4.13 Reference

public Class

This Interface provides access to the various lookup tables within Enterprise Architect. 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Short</td>
<td>Count of items in the list.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>
Attribute | Type | Notes
--- | --- | ---
Type | String | The list type (e.g. Diagram Types).

### Reference Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| GetAt (shortIndex) | String | Get the item at the specified index. Parameters:  
- Index: Short - The index of the item to retrieve from the list. |
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object.  
This function is rarely used as an exception is thrown when an error occurs. |
| Refresh () | Short | Refresh the current list and return the count of items. |

### 16.7.2.4.14 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppliesTo</td>
<td>String</td>
<td>Read/Write. A reference to the stereotype <em>Base Class</em>, i.e. which element it applies to.</td>
</tr>
<tr>
<td>MetafileLoadPath</td>
<td>String</td>
<td>Read/Write. Path to an associated metafile. The automation interface does not yet support loading metafiles. To do this you must use the Stereotype tab of the UML Types dialog in Enterprise Architect.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about the stereotype.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The stereotype name. Appears in the Stereotype drop list for elements that match the AppliesTo attribute.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>StereotypeGUID</td>
<td>String</td>
<td>Read/Write. Unique identifier for stereotype, generally set and maintained by Enterprise Architect.</td>
</tr>
<tr>
<td>Style</td>
<td>String</td>
<td>Read/Write. Additional style specifier for stereotype.</td>
</tr>
<tr>
<td>VisualType</td>
<td>String</td>
<td>Read/Write. Indicates an inbuilt visual style associated with a stereotype. Not currently implemented.</td>
</tr>
</tbody>
</table>

### Stereotype Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| GetLastError () | String | Returns a string value describing the most recent error that occurred in relation to this object.  
This function is rarely used as an exception is thrown when an error occurs. |
Method | Type | Notes
--- | --- | ---
Update () | Boolean | Update the current stereotype object after modification or appending a new item. If \textbf{false} is returned, check the \texttt{GetLastError} function for more information.

16.7.2.4.15 Task

public Class

A Task is an entry in the System ToDo list. Accessed using the Repository \textit{Tasks} collection.

Associated table in \texttt{.EAP} file: \texttt{t\_tasks}

Task Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActualTime</td>
<td>\texttt{Long}</td>
<td>Read/Write. Time already expended on task, in hours, days or other units.</td>
</tr>
<tr>
<td>AssignedTo</td>
<td>\texttt{String}</td>
<td>Read/Write. Person this task is assigned to; i.e. the responsible resource.</td>
</tr>
<tr>
<td>EndDate</td>
<td>\texttt{Date}</td>
<td>Read/Write. Date task scheduled to finish.</td>
</tr>
<tr>
<td>History</td>
<td>\texttt{String}</td>
<td>Read/Write. Memo field to hold, for example, task history or notes.</td>
</tr>
<tr>
<td>Name</td>
<td>\texttt{Variant}</td>
<td>Read/Write. Task name.</td>
</tr>
<tr>
<td>Notes</td>
<td>\texttt{Variant}</td>
<td>Read/Write. Description of the task.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>\texttt{ObjectType}</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Owner</td>
<td>\texttt{String}</td>
<td>Read/Write. The task owner.</td>
</tr>
<tr>
<td>Percent</td>
<td>\texttt{Long}</td>
<td>Read/Write. Percent the task is complete.</td>
</tr>
<tr>
<td>Phase</td>
<td>\texttt{String}</td>
<td>Read/Write. The phase of the project the task relates to.</td>
</tr>
<tr>
<td>Priority</td>
<td>\texttt{String}</td>
<td>Read/Write. Priority associated with this task.</td>
</tr>
<tr>
<td>StartDate</td>
<td>\texttt{Date}</td>
<td>Read/Write. Date task is to start.</td>
</tr>
<tr>
<td>Status</td>
<td>\texttt{Variant}</td>
<td>Read/Write. Current task status.</td>
</tr>
<tr>
<td>TaskID</td>
<td>\texttt{Long}</td>
<td>Read only. Local ID of task.</td>
</tr>
<tr>
<td>TotalTime</td>
<td>\texttt{Long}</td>
<td>Read/Write. The total expected time the task might run - in hours, days or some other unit.</td>
</tr>
<tr>
<td>Type</td>
<td>\texttt{String}</td>
<td>Read/Write. Sets or returns string representing the type.</td>
</tr>
</tbody>
</table>

Task Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>\texttt{String}</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>\texttt{Boolean}</td>
<td>Update the current Task object after modification or appending a new item. If \textbf{false} is returned, check the \texttt{GetLastError} function for more information.</td>
</tr>
</tbody>
</table>
16.7.2.4.16 Term

**public Class**

A *Term* object represents one entry in the system glossary. Accessed using the Repository *Terms* collection.

Associated table in .EAP file: *t_glossary*

**Term Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meaning</td>
<td>String</td>
<td>Read/Write. The description of the term; its meaning.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Term</td>
<td>String</td>
<td>Read/Write. The glossary item name.</td>
</tr>
<tr>
<td>TermID</td>
<td>Long</td>
<td>Read only. A local ID number to identify the term in the model.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The type this term applies to (e.g. business or technical).</td>
</tr>
</tbody>
</table>

**Term Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Term object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.5 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 Enterprise Architect model. Classes, Use Cases and Components are all different types of UML element.

The diagram below illustrates the relationships between an *element* and its associated extended information. The related information is accessed through the collections owned by the element (e.g. Scenarios and Tests). It also includes a full description of the element object (the basic model structural unit).
### 16.7.2.5.1 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The name of the constraint (i.e. the constraint).</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about the constraint.</td>
</tr>
<tr>
<td>ObjectType</td>
<td><code>ObjectType</code></td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read only. The <code>ElementID</code> of the element to which this constraint applies.</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Read/Write. Current status.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. Constraint type.</td>
</tr>
<tr>
<td>Weight</td>
<td>Long</td>
<td>Read/Write. A weighting factor.</td>
</tr>
</tbody>
</table>

#### Constraint Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| GetLastError () | String     | Returns a string value describing the most recent error that occurred in relation to this object.  
|                |            | This function is rarely used as an exception is thrown when an error occurs.          |
| 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. |

### 16.7.2.5.2 Effort

**public Class**

An *Effort* is a named item with a weighting that can be associated with an element for purposes of building metrics about the model. Accessed through the Element *Efforts* collection.

Associated table in .EAP file: `t_objecteffort`

#### Effort Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The name of the effort.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about the effort.</td>
</tr>
<tr>
<td>ObjectType</td>
<td><code>ObjectType</code></td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The effort type.</td>
</tr>
<tr>
<td>Weight</td>
<td>Long</td>
<td>Read/Write. A weighting factor.</td>
</tr>
</tbody>
</table>
### Effort Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Saves the effort to the model.</td>
</tr>
</tbody>
</table>

### 16.7.2.5.3 Element

**public Class**

An *Element* is the main modeling unit. It corresponds to (for example) Class, Use Case, Node or Component. 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 the 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 addition information about the element, including things such as Tagged Values, Issues, Constraints and Requirements.

Associated table in .EAP file: *t_object*

### Element Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>String</td>
<td>Read/Write. Indicates if the element is Abstract (1) or Concrete (0).</td>
</tr>
<tr>
<td>ActionFlags</td>
<td>String</td>
<td>Read/Write. A structure to hold flags concerned with Action semantics.</td>
</tr>
<tr>
<td>Alias</td>
<td>String</td>
<td>Read/Write. An optional alias for this element.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Collection</td>
<td>Read only. Collection of Attribute objects for current element. Use the AddNew and Delete functions to manage attributes.</td>
</tr>
<tr>
<td>AttributesEx</td>
<td>Collection</td>
<td>Read only. Collection of Attribute objects belonging to the current element and its parent elements.</td>
</tr>
<tr>
<td>Author</td>
<td>String</td>
<td>Read/Write. The element author (see the Repository: Authors list for more details).</td>
</tr>
<tr>
<td>BaseClasses</td>
<td>Collection</td>
<td>Read only. List of Base Classes for this element presented as a collection for convenience.</td>
</tr>
<tr>
<td>ClassifierID</td>
<td>Long</td>
<td>Deprecated. See ClassifierID.</td>
</tr>
<tr>
<td>ClassifierID</td>
<td>Long</td>
<td>Read/Write. ElementID of a Classifier associated with this element; that is, the base type. Only valid for instance type elements (e.g. Object, Sequence).</td>
</tr>
<tr>
<td>ClassifierName</td>
<td>String</td>
<td>Read/Write. Name of associated Classifier (if any).</td>
</tr>
<tr>
<td>ClassifierType</td>
<td>String</td>
<td>Read only. Type of associated classifier.</td>
</tr>
<tr>
<td>Complexity</td>
<td>String</td>
<td>Read/Write. A complexity value indicating how difficult the element is. Can be used for metric reporting and estimation. Valid values are: 1 for Easy, 2 for Medium, 3 for Difficult.</td>
</tr>
<tr>
<td>CompositeDiagram</td>
<td>Diagram</td>
<td>Read only. If the element is Composite, returns its associated diagram; otherwise returns null.</td>
</tr>
<tr>
<td>Connectors</td>
<td>Collection</td>
<td>Read only. Returns a collection containing the connectors to other elements.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Constraints</td>
<td>Collection</td>
<td>Read only. Collection of Constraint objects.</td>
</tr>
<tr>
<td>ConstraintsEx</td>
<td>Collection</td>
<td>Read only. Collection of Constraint objects belonging to the current element and its parent elements.</td>
</tr>
<tr>
<td>Created</td>
<td>Date</td>
<td>Read/Write. The date the element was created.</td>
</tr>
<tr>
<td>CustomProperties</td>
<td>Collection</td>
<td>Read only. List of advanced properties for an element. The collection of advanced properties differs depending on element type; for example, an Action and an Activity have different advanced properties. Currently only editable from the user interface.</td>
</tr>
<tr>
<td>Diagrams</td>
<td>Collection</td>
<td>Read only. Returns a collection of sub-diagrams (child diagrams) attached to this element as seen in the tree view.</td>
</tr>
<tr>
<td>Difficulty</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Efforts</td>
<td>Collection</td>
<td>Read only. Collection of Effort objects.</td>
</tr>
<tr>
<td>ElementGUID</td>
<td>String</td>
<td>Read only. A globally unique ID for this element; i.e. unique across all model files. If you have 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 Enterprise Architect expects.</td>
</tr>
<tr>
<td>ElementID</td>
<td>Long</td>
<td>Read only. The local ID of the Element. Valid for this file only.</td>
</tr>
<tr>
<td>Elements</td>
<td>Collection</td>
<td>Read only. Returns a collection of child elements (sub-elements) attached to this element as seen in the tree view.</td>
</tr>
<tr>
<td>EmbeddedElements</td>
<td>Collection</td>
<td>Read only. List of elements that are embedded into this element, such as Ports, Parts, Pins and Parameter Sets.</td>
</tr>
<tr>
<td>EventFlags</td>
<td>String</td>
<td>Read/Write. A structure to hold a variety of flags to do with signals or events.</td>
</tr>
<tr>
<td>ExtensionPoints</td>
<td>String</td>
<td>Read/Write. Optional extension points for a Use Case as a comma-separated list.</td>
</tr>
<tr>
<td>Files</td>
<td>Collection</td>
<td>Read only. Collection of File objects.</td>
</tr>
<tr>
<td>GenFile</td>
<td>String</td>
<td>Read/Write. The file associated with this element for code generation and synchronization purposes. Can include macro expansion tags for local conversion to full path.</td>
</tr>
<tr>
<td>Genlinks</td>
<td>String</td>
<td>Read/Write. Links to other Classes discovered at code reversing time; Parents and Implements connectors only.</td>
</tr>
<tr>
<td>GenType</td>
<td>String</td>
<td>Read/Write. The code generation type; e.g. Java, C++, C#, VBNet, Visual Basic, Delphi.</td>
</tr>
<tr>
<td>Header1</td>
<td>Variant</td>
<td>Read/Write. A user defined string for inclusion as header in the source files generated.</td>
</tr>
<tr>
<td>Header2</td>
<td>Variant</td>
<td>Read/Write. Same as for Header1, but used in the CPP source file.</td>
</tr>
<tr>
<td>IsActive</td>
<td>Boolean</td>
<td>Read/Write. Boolean value indicating whether the element is active or not. 1 = True, 0 = False.</td>
</tr>
<tr>
<td>IsLeaf</td>
<td>Boolean</td>
<td>Read/Write. Boolean value indicating whether the element is in leaf node or not. 1 = True, 0 = False.</td>
</tr>
<tr>
<td>isNew</td>
<td>Boolean</td>
<td>Read/Write. Boolean value indicating whether the element is new or not.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsSpec</td>
<td>Boolean</td>
<td>Read/Write. Boolean value indicating whether the element is a specification or not. 1 = True, 0 = False.</td>
</tr>
<tr>
<td>Issues</td>
<td>Collection</td>
<td>Read only. Collection of Issue objects.</td>
</tr>
<tr>
<td>Locked</td>
<td>Boolean</td>
<td>Read/Write. Indicates if the element has been locked against further change.</td>
</tr>
<tr>
<td>MetaType</td>
<td>String</td>
<td>Read only: The element's domain-specific meta type, as defined by an applied stereotype from an MDG Technology.</td>
</tr>
<tr>
<td>Methods</td>
<td>Collection</td>
<td>Read only. Collection of Method objects for current element.</td>
</tr>
<tr>
<td>MethodsEx</td>
<td>Collection</td>
<td>Read only. Collection of Method objects belonging to the current element and its parent elements.</td>
</tr>
<tr>
<td>Metrics</td>
<td>Collection</td>
<td>Read only. Collection of Metric elements for current element.</td>
</tr>
<tr>
<td>MiscData</td>
<td>String</td>
<td>Read only. This low-level property provides information about the contents of the PData fields. These database fields are not documented and developers must gain understanding of these fields through their own endeavors to use this property. MiscData is zero based, therefore: MiscData(0) corresponds to PDATA1 MiscData(1) to PDATA2 and so on.</td>
</tr>
<tr>
<td>Modified</td>
<td>Date</td>
<td>Read/Write. The date the element was last modified.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>String</td>
<td>Read/Write. Multiplicity value for this element.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The element name; should be unique within the current package.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Further descriptive text about the element.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>PackageID</td>
<td>Long</td>
<td>Read/Write. A local ID for the package containing this element.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read/Write. If this element is a child of another, used to set or retrieve the ElementID of the other element. If not, returns 0.</td>
</tr>
<tr>
<td>Partitions</td>
<td>Collection</td>
<td>Read Only. List of logical partitions into which an element can be divided. Only valid for elements that support partitions, such as Activities and States.</td>
</tr>
<tr>
<td>Persistence</td>
<td>String</td>
<td>Read/Write. The persistence associated with this element. Can be Persistent or Transient.</td>
</tr>
<tr>
<td>Phase</td>
<td>String</td>
<td>Read/Write. Phase this element scheduled to be constructed in. Any string value.</td>
</tr>
<tr>
<td>Priority</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Properties</td>
<td>Properties</td>
<td>Returns a list of specialized properties that apply to the element that might 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.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>PropertyType</td>
<td>Long</td>
<td>Read/Write. The GUID of the type which defines either a Port or a Part.</td>
</tr>
<tr>
<td>Realizes</td>
<td>Collection</td>
<td>Read only. List of Interfaces realized by this element for convenience.</td>
</tr>
<tr>
<td>Requirements</td>
<td>Collection</td>
<td>Read only. Collection of Requirement objects.</td>
</tr>
<tr>
<td>RequirementsEx</td>
<td>Collection</td>
<td>Read only. Collection of Requirement objects belonging to the current element and its parent elements.</td>
</tr>
<tr>
<td>Resources</td>
<td>Collection</td>
<td>Read only. Collection of Resource objects for current element.</td>
</tr>
<tr>
<td>Risks</td>
<td>Collection</td>
<td>Read only. Collection of Risk objects.</td>
</tr>
<tr>
<td>RunState</td>
<td>String</td>
<td>Read/Write. The object's runstate list as a string.</td>
</tr>
<tr>
<td>Scenarios</td>
<td>Collection</td>
<td>Read only. Collection of Scenario objects for current element.</td>
</tr>
<tr>
<td>StateTransitions</td>
<td>Collection</td>
<td>Read only. List of State Transitions that an element can support. Applies in particular to Timing elements.</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Read/Write. Sets or gets the status, such as Proposed or Approved.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>String</td>
<td>Read/Write. The primary element stereotype. This is the first of the list of stereotypes you can access using the StereotypeEx attribute.</td>
</tr>
<tr>
<td>StereotypeEx</td>
<td>String</td>
<td>Read/Write. All the applied stereotypes of the element in a comma-separated list.</td>
</tr>
<tr>
<td>StyleEx</td>
<td>String</td>
<td>Read/Write. Advanced style settings. Not currently used.</td>
</tr>
<tr>
<td>Subtype</td>
<td>Long</td>
<td>Read/Write. A numeric subtype that qualifies the Type of the main element. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For Event: 0 = Receiver, 1 = Sender</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For Class: 1 = Parameterised, 2 = Instantiated, 3 = Both, 0 = Neither, 17 = Association Class</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If 17, because an Association Class has been created through the user interface, MiscData(3) will contain the ID of the related Association. As MiscData is read-only, you cannot create an Association Class through the Automation Interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For Note: 1 = Note linked to connector, 2 = Constraint linked to connector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For StateNode: 100 = ActivityInitial, 101 = ActivityFinal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For Activity: 0 = Activity, 8 = composite Activity (also set to 8 for other composite elements such as Use Cases)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For Synchronization: 0 = Horizontal, 1 = Vertical.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that there are many more Types than indicated in the above examples.</td>
</tr>
<tr>
<td>Tablespace</td>
<td>String</td>
<td>Read/Write. Associated tablespace for a Table element.</td>
</tr>
<tr>
<td>Tag</td>
<td>String</td>
<td>Read/Write. Corresponds to the Keywords field in the Enterprise Architect user interface. See the General Settings topic.</td>
</tr>
<tr>
<td>TaggedValues</td>
<td>Collection</td>
<td>Read only. Returns a collection of TaggedValue objects.</td>
</tr>
</tbody>
</table>
## Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaggedValuesEx</td>
<td>Collection of type TaggedValue</td>
<td>Read only. Returns a collection of TaggedValue objects belonging to the current element and the elements specialized or realized by the current element.</td>
</tr>
<tr>
<td>Tests</td>
<td>Collection</td>
<td>Read only. Collection of Test objects for current element.</td>
</tr>
<tr>
<td>TreePos</td>
<td>Long</td>
<td>Read/Write. Sets or gets the tree position.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The element type (e.g. Class, Component). Note that Type is case sensitive inside Enterprise Architect and should be provided with an initial capital (proper case). Valid types are:</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>Read/Write. The version of the element.</td>
</tr>
<tr>
<td>Visibility</td>
<td>String</td>
<td>Read/Write. The Scope of this element within the current package. Valid values are: Public, Private, Protected or Package.</td>
</tr>
</tbody>
</table>

## Element Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyGroupLock (string aGroupName)</td>
<td>Boolean</td>
<td>Applies a group lock to the element object, for the specified group, on behalf of the current user. Throws an exception if the operation fails. Use GetLastError() to retrieve error information. Parameter: aGroupName: String - the name of the user group for which to set the group lock.</td>
</tr>
<tr>
<td>ApplyUserLock ()</td>
<td>Boolean</td>
<td>Applies a user lock to the element object for the current user. Throws an exception if the operation fails. Use GetLastError() to</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetLastErrorMessage()</td>
<td>String</td>
<td>retrieve error information.</td>
</tr>
<tr>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetLinkedDocument()</td>
<td>String</td>
<td>Returns a string value containing the element's linked document contents, in RTF format. If the element contains no linked document, an empty string is returned.</td>
</tr>
<tr>
<td>GetRelationSet(EnumRelationSetType Type)</td>
<td>String</td>
<td>Returns a string containing a comma-separated list of Element IDs of directly- and indirectly-related elements based on the given type. See Enum Relation Set Type. Recurses using the same relation type on all elements it finds, retrieving all dependencies and sub-dependencies of the current element; for example, Object1 depends on Object2, which depends on Object3. Therefore this method returns Object2 and Object3. To obtain only the direct relationships of the element, use the Connector collection instead.</td>
</tr>
<tr>
<td>GetStereotypeList()</td>
<td>String</td>
<td>Returns a comma-separated list of stereotypes allied to this element.</td>
</tr>
<tr>
<td>LoadLinkedDocument(string Filename)</td>
<td>Boolean</td>
<td>Loads the RTF document from the specified file into the element's linked document. Parameter: FileName: String - the name of the file from which to load the RTF document.</td>
</tr>
<tr>
<td>Refresh()</td>
<td>Void</td>
<td>Refreshes the element features in the Project Browser. Usually called after adding or deleting attributes or methods, when the user interface is required to be updated as well.</td>
</tr>
<tr>
<td>ReleaseLock()</td>
<td>Boolean</td>
<td>Releases a user lock or group lock on the element object. Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</td>
</tr>
<tr>
<td>SaveLinkedDocument(string Filename)</td>
<td>Boolean</td>
<td>Saves the linked document for this element to the specified RTF file. Parameter: FileName: String - the name of the RTF file to which to save the linked document.</td>
</tr>
<tr>
<td>SetAppearance(long Scope, long Item, long Value)</td>
<td>Void</td>
<td>Sets the visual appearance of the element. Parameter: Scope: Long - Scope of appearance set to modify 0 – Local (Diagram-local appearance) 1 – Base (Default appearance across entire model) Item: Long - Appearance item to modify 0 – Background color 1 – Font Color 2 – Border Color 3 – Border Width Value: Long - Value to set appearance to.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current element object after modification or appending a new item.</td>
</tr>
</tbody>
</table>
### 16.7.2.5.4 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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileDate</td>
<td>String</td>
<td>Read/Write. The file date when entry is created.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The file name can be a logical file or a reference to a web address (using http://).</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about the file.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Size</td>
<td>String</td>
<td>Read/Write. The file size.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. File type.</td>
</tr>
</tbody>
</table>

**File Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current File object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.5.5 Issue (Maintenance)

**public Class**

An Issue is either a Change or a Defect, is associated with the containing element, and is accessed through the Issues collection of an element.

Associated table in .EAP file: `t_objectproblems`

**Issue Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateReported</td>
<td>Date</td>
<td>Read/Write. Date issue reported.</td>
</tr>
<tr>
<td>DateResolved</td>
<td>Date</td>
<td>Read/Write. Date issue resolved.</td>
</tr>
<tr>
<td>ElementID</td>
<td>Long</td>
<td>Read/Write. ID of element associated with this issue.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The Issue name - i.e. the Issue itself.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Issue Description.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Priority</td>
<td>String</td>
<td>Read/Write. Issue priority. Generally should use <strong>Low</strong>, <strong>Medium</strong> and <strong>High</strong>.</td>
</tr>
<tr>
<td>Reporter</td>
<td>String</td>
<td>Read/Write. Person reporting issue.</td>
</tr>
<tr>
<td>Resolver</td>
<td>String</td>
<td>Read/Write. Person resolving issue.</td>
</tr>
<tr>
<td>ResolverNotes</td>
<td>String</td>
<td>Read/Write. Notes entered by resolver about resolution.</td>
</tr>
<tr>
<td>Severity</td>
<td>String</td>
<td>Read/Write. Issue severity. Should be marked as <strong>Low</strong>, <strong>Medium</strong> or <strong>High</strong>.</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Read/Write. The current status of the issue.</td>
</tr>
<tr>
<td>Type</td>
<td>Variant</td>
<td>Read/Write. Issue type - can be <strong>Defect</strong>, <strong>Change</strong>, <strong>Issue</strong> and <strong>ToDo</strong>.</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>Read/Write. Version associated with issue. Note that this method is only available through a Dispatch interface. e.g. Object ob = Issue; Print ob.Version;</td>
</tr>
</tbody>
</table>

### Issue Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Issue object after modification or appending a new item. If <strong>false</strong> is returned, check the <strong>GetLastError</strong> function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.5.6 Metric

**public Class**

A **Metric** is a named item with a weighting that can 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The name of the metric.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about this metric.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The metric type.</td>
</tr>
<tr>
<td>Weight</td>
<td>Long</td>
<td>Read/Write. A user defined weighting for estimation or metric purposes.</td>
</tr>
</tbody>
</table>

### Metric Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Metric object after modification or appending a new item. If <strong>false</strong> is returned, check the <strong>GetLastError</strong> function for more information.</td>
</tr>
</tbody>
</table>
16.7.2.5.7 Requirement

**public Class**


Associated table in .EAP file: t_objectrequires

**Requirement Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty</td>
<td>String</td>
<td>Read/Write. Estimated difficulty to implement.</td>
</tr>
<tr>
<td>LastUpdate</td>
<td>Date</td>
<td>Read/Write. Date requirement last updated.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The requirement itself.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Further notes about requirement.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read only. The ElementID of the element to which this requirement applies.</td>
</tr>
<tr>
<td>Priority</td>
<td>String</td>
<td>Read/Write. Assigned priority of the requirement.</td>
</tr>
<tr>
<td>RequirementID</td>
<td>Long</td>
<td>Read only. A local ID for this requirement.</td>
</tr>
<tr>
<td>Stability</td>
<td>String</td>
<td>Read/Write. Estimated stability of the requirement.</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Read/Write. Current status of the requirement.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. Requirement type.</td>
</tr>
</tbody>
</table>

**Requirement Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Requirement object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.5.8 Resource

**public 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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActualHours</td>
<td>Long</td>
<td>Read/Write. Time already expended on the task, in hours, days or other</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>DateEnd</td>
<td>Date</td>
<td>Read/Write. Expected end date.</td>
</tr>
<tr>
<td>DateStart</td>
<td>Date</td>
<td>Read/Write. Date to start work.</td>
</tr>
<tr>
<td>ExpectedHours</td>
<td>Long</td>
<td>Read/Write. The total expected time the task might run, in hours, days or other units.</td>
</tr>
<tr>
<td>History</td>
<td>String</td>
<td>Read/Write. Gets or sets history text.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. Name of resource (e.g. person's name).</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>PercentComplete</td>
<td>Long</td>
<td>Read/Write. Current percent complete figure.</td>
</tr>
<tr>
<td>Role</td>
<td>String</td>
<td>Read/Write. Role they play in implementing the element.</td>
</tr>
<tr>
<td>Time</td>
<td>Long</td>
<td>Read/Write. Time expected; numeric indicating number of days.</td>
</tr>
</tbody>
</table>

**Resource Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Resource object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

**16.7.2.5.9 Risk**

**public Class**

A Risk object represents a named risk associated with an element and is used for project management purposes. Accessed through the Element Risks collection.

Associated table in .EAP file: t_objectrisks

**Risk Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The risk.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Further notes describing the risk.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The risk type associated with this element.</td>
</tr>
<tr>
<td>Weight</td>
<td>Long</td>
<td>Read/Write. A weighting for estimation or metric purposes.</td>
</tr>
</tbody>
</table>

**Risk Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object.</td>
</tr>
</tbody>
</table>
### 16.7.2.5.10 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 can be added to using the Element *Scenarios* collection.

Associated table in .EAP file: `t_objectscenarios`

#### Scenario Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The Scenario name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Description of the Scenario. Usually contains the steps to execute the scenario.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>ScenarioGUID</td>
<td>String</td>
<td>Read/Write. A unique ID for the scenario. Used to identify the scenario unambiguously within a model.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The Scenario type (e.g. Basic Path).</td>
</tr>
<tr>
<td>Weight</td>
<td>Long</td>
<td>Read/Write. Currently used to position scenarios in the scenario list (i.e. List Position).</td>
</tr>
<tr>
<td>XMLContent</td>
<td>String</td>
<td>Read/Write. A structured field that can contain scenario details in XML format. <em>Not currently used.</em></td>
</tr>
</tbody>
</table>

#### Scenario Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current Scenario object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.5.11 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElementID</td>
<td>Long</td>
<td>Read/Write. The local ID of the associated element.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. Name of the property (Tag).</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Further descriptive notes.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>PropertyGUID</td>
<td>String</td>
<td>Read/Write. A global ID for the property.</td>
</tr>
<tr>
<td>PropertyID</td>
<td>Long</td>
<td>Read only. A local ID for the property.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Read/Write. The value assigned in this instance.</td>
</tr>
</tbody>
</table>

TaggedValue Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current TaggedValue object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.5.12 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptanceCriteria</td>
<td>String</td>
<td>Read/Write. The acceptance criteria for successful execution.</td>
</tr>
<tr>
<td>CheckedBy</td>
<td>String</td>
<td>Read/Write. Results confirmed by.</td>
</tr>
<tr>
<td>Class</td>
<td>Long</td>
<td>Read/Write. The test Class:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Unit Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = Integration Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 = System Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 = Acceptance Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 = Scenario Test</td>
</tr>
<tr>
<td>DateRun</td>
<td>Date</td>
<td>Read/Write. Date last run.</td>
</tr>
<tr>
<td>Input</td>
<td>String</td>
<td>Read/Write. Input data.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The test name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Detailed notes about test to be carried out.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>RunBy</td>
<td>String</td>
<td>Read/Write. Person conducting test.</td>
</tr>
<tr>
<td>TestResults</td>
<td>Variant</td>
<td>Read/Write. Results of test.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The test type, such as Load or Regression.</td>
</tr>
</tbody>
</table>
16.7.2.6 Element Features

public Package

The ElementFeatures package contains descriptions of the model interfaces that enable access to operations and attributes, and their associated Tagged Values and constraints. 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 that defines the arguments associated with an operation (method).
16.7.2.6.1 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowDuplicates</td>
<td>Boolean</td>
<td>Read/Write. Indicates if duplicates are allowed in the collection. If the attribute represents a database column, this when set represents the <strong>Not Null</strong> option.</td>
</tr>
<tr>
<td>AttributeGUID</td>
<td>String</td>
<td>Read/Write. A globally unique ID for the current attribute. System generated.</td>
</tr>
<tr>
<td>AttributeID</td>
<td>Long</td>
<td>Read only. Local ID number of the attribute.</td>
</tr>
<tr>
<td>ClassifierID</td>
<td>Long</td>
<td>Read/Write. Classifier ID, if appropriate; indicates the base type associated with attribute, if not a primitive type.</td>
</tr>
<tr>
<td>Container</td>
<td>String</td>
<td>Read/Write. The container type.</td>
</tr>
<tr>
<td>Containment</td>
<td>String</td>
<td>Read/Write. Type of containment. Can be <strong>Not Specified</strong>, <strong>By Reference</strong> or <strong>By Value</strong>.</td>
</tr>
<tr>
<td>Constraints</td>
<td><strong>Collection</strong></td>
<td>Read only. A collection of <strong>AttributeConstraint</strong> objects. Used to access and manage constraints associated with this attribute.</td>
</tr>
<tr>
<td>Default</td>
<td>String</td>
<td>Read/Write. Initial value assigned to this attribute.</td>
</tr>
<tr>
<td>IsCollection</td>
<td>Boolean</td>
<td>Read/Write. Indicates if the current feature is a collection or not. If the attribute represents a database column, this when set represents a <strong>Foreign Key</strong>.</td>
</tr>
<tr>
<td>IsConst</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating if the attribute is <strong>Const</strong> or not.</td>
</tr>
<tr>
<td>IsDerived</td>
<td>Boolean</td>
<td>Read/Write. Indicates if the attribute is derived (e.g. a calculated value).</td>
</tr>
<tr>
<td>IsOrdered</td>
<td>Boolean</td>
<td>Read/Write. Indicates if a collection is ordered or not. If the attribute represents a database column, this when set represents a <strong>Primary Key</strong>.</td>
</tr>
<tr>
<td>IsStatic</td>
<td>Boolean</td>
<td>Read/Write. Indicates if the current attribute is a static feature or not. If the attribute represents a database column, this when set represents the <strong>Unique</strong> option.</td>
</tr>
<tr>
<td>Length</td>
<td>String</td>
<td>Read/Write. The attribute length, where applicable.</td>
</tr>
<tr>
<td>LowerBound</td>
<td>String</td>
<td>Read/Write. A value for the collection lower bound.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The attribute name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Further notes about this attribute.</td>
</tr>
<tr>
<td>ObjectType</td>
<td><strong>ObjectType</strong></td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read only. Returns the <strong>ElementID</strong> of the element that this attribute is a part of.</td>
</tr>
<tr>
<td>Pos</td>
<td>Long</td>
<td>Read/Write. Position of the attribute in the Class attribute list.</td>
</tr>
<tr>
<td>Precision</td>
<td>String</td>
<td>Read/Write. Precision value.</td>
</tr>
<tr>
<td>Scale</td>
<td>String</td>
<td>Read/Write. Scale value.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>String</td>
<td>Read/Write. Sets or gets the stereotype for this attribute.</td>
</tr>
<tr>
<td>StereotypeEx</td>
<td>String</td>
<td>Read/Write. All the applied stereotypes of the attribute in a comma-separated list.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Style</td>
<td>String</td>
<td>Read/Write. Contains the Alias property for this attribute.</td>
</tr>
<tr>
<td>StyleEx</td>
<td>String</td>
<td>Read/Write. Advanced style settings. Use with care.</td>
</tr>
<tr>
<td>TaggedValues</td>
<td>Collection</td>
<td>Read only. A collection of AttributeTag objects. Use to access and manage Tagged Values associated with this attribute.</td>
</tr>
<tr>
<td>TaggedValuesEx</td>
<td>Collection</td>
<td>Read only. Collection of TaggedValue objects belonging to the current attribute and the TaggedValuesEx property of its classifier.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The attribute type (by name; also see ClassifierID).</td>
</tr>
<tr>
<td>UpperBound</td>
<td>String</td>
<td>Read/Write. A value for the collection upper bound.</td>
</tr>
<tr>
<td>Visibility</td>
<td>String</td>
<td>Read/Write. The scope of the attribute. Can be Private, Protected, Public or Package.</td>
</tr>
</tbody>
</table>

**Attribute Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Updates the current attribute object after modifying or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

**16.7.2.6.2 AttributeConstraint**

**public Class**

An AttributeConstraint is a constraint associated with the current Attribute.

Associated table in .EAP file: `t_attributeconstraints`

**AttributeConstraint Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeID</td>
<td>Long</td>
<td>Read/Write. ID of the attribute this constraint applies to.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The constraint.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes about constraint.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. Type of constraint.</td>
</tr>
</tbody>
</table>

**AttributeConstraint Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current AttributeConstraint object after modification or...</td>
</tr>
</tbody>
</table>
### 16.7.2.6.3 AttributeTag

**public Class**

An AttributeTag represents a Tagged Value associated with an attribute.

Associated table in .EAP file: `t_attributetag`

**AttributeTag Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeID</td>
<td>Long</td>
<td>Read/Write. Local ID of attribute associated with this Tagged Value.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. Name of tag.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>TagGUID</td>
<td>String</td>
<td>Read/Write. A globally unique ID for this Tagged Value.</td>
</tr>
<tr>
<td>TagID</td>
<td>Long</td>
<td>Read only. Local ID to identify Tagged Value.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Read/Write. Value associated with this tag.</td>
</tr>
</tbody>
</table>

**AttributeTag Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current AttributeTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.6.4 CustomProperties

**public Collection**

The CustomProperties collection contains 0 or more Cust Properties associated with the current element. These properties provide advanced UML configuration options, and must not be added to or deleted. The value of each property can be set.

**Note:**

The number and type of properties vary depending on the actual element.

**CustomProperty**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read-only. The CustomProperty name.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>
16.7.2.6.5 EmbeddedElements

**public Collection**

In UML 2.1 an element can have one or more embedded elements such as Ports, Pins, Parameters or ObjectNodes. These are attached to the boundary of the host element and cannot 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 can have their own embedded element collection (e.g. Ports can have Interfaces embedded on them).

The EmbeddedElements collection contains Element objects.

16.7.2.6.6 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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating if the method is abstract (1) or not (0).</td>
</tr>
<tr>
<td>Behavior</td>
<td>String</td>
<td>Read/Write. Some further explanatory behavior notes (e.g. pseudocode).</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
<td>In earlier releases of Enterprise Architect this attribute had the UK/ Australian spelling ‘Behaviour’; this is still present for backwards compatibility, but please now use the ‘Behavior’ attribute for consistency.</td>
</tr>
<tr>
<td>ClassifierID</td>
<td>String</td>
<td>Read/Write. Classifier ID that applies to the ReturnType.</td>
</tr>
<tr>
<td>Code</td>
<td>String</td>
<td>Read/Write. Optional field to hold the method Code (used for the Initial Code field).</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Variant</td>
<td>Read/Write. Concurrency type of method.</td>
</tr>
<tr>
<td>IsConst</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating the method is Const.</td>
</tr>
<tr>
<td>IsLeaf</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate if the method is Leaf (cannot be overridden).</td>
</tr>
<tr>
<td>IsPure</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating the method is defined as Pure in C++.</td>
</tr>
<tr>
<td>IsQuery</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate if the method is a query (i.e. does not alter Class variables).</td>
</tr>
<tr>
<td>IsRoot</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate if the method is Root.</td>
</tr>
<tr>
<td>IsStatic</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate a static method.</td>
</tr>
<tr>
<td>IsSynchronized</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating a Synchronized method call.</td>
</tr>
<tr>
<td>MethodID</td>
<td>Long</td>
<td>Read only. A local ID for the current method, only valid within this .EAP file.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The method name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes about the method.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Collection</td>
<td>Read only. The Parameters collection for the current method. Use to add and access parameter objects for the current method.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read only. An optional ID of an element that 'owns' this diagram; e.g. a Sequence diagram owned by a Use Case.</td>
</tr>
<tr>
<td>Pos</td>
<td>Long</td>
<td>Read/Write. Specifies the position of the method within the set of operations defined for a Class.</td>
</tr>
<tr>
<td>PostConditions</td>
<td>Collection</td>
<td>Read only. PostConditions (constraints) as they apply to this method. Returns a MethodConstraint object of type post.</td>
</tr>
<tr>
<td>PreConditions</td>
<td>Collection</td>
<td>Read only. PreConditions (constraints) as they apply to this method. Returns a MethodConstraint object of type pre.</td>
</tr>
<tr>
<td>ReturnIsArray</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate the return value is an array.</td>
</tr>
<tr>
<td>ReturnType</td>
<td>String</td>
<td>Read/Write. Return type for the method; can be a primitive data type or a Class or Interface type.</td>
</tr>
<tr>
<td>StateFlags</td>
<td>String</td>
<td>Read/Write. Some flags as applied to methods in State elements.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>String</td>
<td>Read/Write. The method stereotype (optional).</td>
</tr>
<tr>
<td>StereotypeEx</td>
<td>String</td>
<td>Read/Write. All the applied stereotypes of the method in a comma-separated list.</td>
</tr>
<tr>
<td>Style</td>
<td>String</td>
<td>Read/Write. Contains the Alias property for this method.</td>
</tr>
<tr>
<td>StyleEx</td>
<td>String</td>
<td>Read/Write. Advanced style settings. Not currently used.</td>
</tr>
<tr>
<td>TaggedValues</td>
<td>Collection</td>
<td>Read only. TaggedValues collection for the current method. Accesses a list of MethodTag objects.</td>
</tr>
</tbody>
</table>
### Method Tag

**public Class**

A `MethodTag` is a Tagged Value associated with a method.

Associated table in .EAP file: `t_operationtag`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodID</td>
<td>Long</td>
<td>Read/Write. The local ID of the associated method.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The name of the constraint.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes about this constraint.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The constraint type.</td>
</tr>
</tbody>
</table>

### MethodConstraint

**16.7.2.6.7 MethodConstraint**

**public Class**

A `MethodConstraint` 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`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodID</td>
<td>Long</td>
<td>Read/Write. The local ID of the associated method.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The name of the constraint.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes about this constraint.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. The constraint type.</td>
</tr>
</tbody>
</table>

### MethodConstraint Methods

**16.7.2.6.8 MethodTag**

**public Class**

A `MethodTag` is a Tagged Value associated with a method.

Associated table in .EAP file: `t_operationtag`
**MethodTag Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodID</td>
<td>Long</td>
<td>Read/Write. The ID of the associated method.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The tag or name of the property.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes about this item.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>TagGUID</td>
<td>String</td>
<td>Read/Write. A unique ID for this Tagged Value.</td>
</tr>
<tr>
<td>TagID</td>
<td>Long</td>
<td>Read only. A unique ID for this Tagged Value.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Read/Write. A value to apply to this tag.</td>
</tr>
</tbody>
</table>

**MethodTag Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current MethodTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.6.9 **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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>String</td>
<td>Read/Write. An optional alias for this parameter.</td>
</tr>
<tr>
<td>ClassifierID</td>
<td>String</td>
<td>Read/Write. A ClassifierID for the parameter, if known.</td>
</tr>
<tr>
<td>Default</td>
<td>String</td>
<td>Read/Write. A default value for this parameter.</td>
</tr>
<tr>
<td>IsConst</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating the parameter is Const (cannot be altered).</td>
</tr>
<tr>
<td>Kind</td>
<td>String</td>
<td>Read/Write. The parameter kind - in, inout, out, return.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The parameter name; must be unique for a single method.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>OperationID</td>
<td>Long</td>
<td>Read only. ID of the method associated with this parameter.</td>
</tr>
<tr>
<td>ParameterGUID</td>
<td>String</td>
<td>Read/Write. A globally unique ID for the current Parameter. System generated.</td>
</tr>
<tr>
<td>Position</td>
<td>Long</td>
<td>Read/Write. The position in the argument list.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>String</td>
<td>Read/Write. The first stereotype of the parameter.</td>
</tr>
</tbody>
</table>
### Attribute Types

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>StereotypeEx</td>
<td>String</td>
<td>Read/Write. All the applied stereotypes of the parameter in a comma-separated list.</td>
</tr>
<tr>
<td>Style</td>
<td>String</td>
<td>Read/Write. Some style information.</td>
</tr>
<tr>
<td>StyleEx</td>
<td>String</td>
<td>Read/Write. Advanced style settings. Not currently used.</td>
</tr>
<tr>
<td>Type</td>
<td>Variant</td>
<td>Read/Write. The parameter type; can be a primitive type or defined classifier.</td>
</tr>
</tbody>
</table>

### Parameter Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastErr()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current Parameter object after modifying or appending a new item. If false is returned, check the GetLastErr() function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.6.10 Partitions

#### public Collection

A collection of internal element partitions (regions). This is commonly seen in Activity, State, 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The partition name; can represent a condition or constraint in some cases.</td>
</tr>
<tr>
<td>Note</td>
<td>String</td>
<td>Read/Write. A free text note associated with this partition.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Operator</td>
<td>String</td>
<td>Read/Write. An optional operator value that specifies the partition type.</td>
</tr>
<tr>
<td>Size</td>
<td>String</td>
<td>Read/Write. Vertical or horizontal width of partition in pixels.</td>
</tr>
</tbody>
</table>

### 16.7.2.6.11 Properties

#### Properties

#### Properties Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>The number of properties that are available for this object.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>
### Properties Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item (variant Index)</td>
<td>Property</td>
<td>Returns a property either by name or by zero-based integer offset into the list of properties.</td>
</tr>
</tbody>
</table>

### Property

#### Property Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Read only. Identifies the property. The object to which the properties list applies can 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.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Type</td>
<td>PropType</td>
<td>Read only. Provides an indication of what sort of data is going to be stored by this property. This restriction can be further defined by the Validation attribute.</td>
</tr>
<tr>
<td>Validation</td>
<td>String</td>
<td>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 Enterprise Architect to ensure that the submitted data is appropriate.</td>
</tr>
<tr>
<td>Value</td>
<td>Variant</td>
<td>Read/write. The value of the property as defined in the other fields.</td>
</tr>
</tbody>
</table>

#### 16.7.2.6.12 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 enables you to access the transition set. You can also access additional information by referring to the connectors associated with the Timeline, and by 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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DurationConstraint</td>
<td>String</td>
<td>Read/Write. A constraint on the time duration that the transition takes.</td>
</tr>
<tr>
<td>Event</td>
<td>String</td>
<td>Read/Write. Event (optional) that initiated transition.</td>
</tr>
<tr>
<td>Note</td>
<td>String</td>
<td>Read/Write. A free text note.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>TimeConstraint</td>
<td>String</td>
<td>Read/Write. A constraint on when the transition has to be complete by.</td>
</tr>
<tr>
<td>TxState</td>
<td>String</td>
<td>Read/Write. The state to transition to. Defined in the Timeline Properties dialog.</td>
</tr>
<tr>
<td>TxTime</td>
<td>String</td>
<td>Read/Write. The time that the transition occurs. Value depends on range set in diagram.</td>
</tr>
</tbody>
</table>
16.7.2.7 Connector

public Package

The Connector package details how connectors between elements are accessed and managed.

16.7.2.7.1 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorID</td>
<td>Long</td>
<td>Read/Write. A local ID value (long) - system generated.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The constraint name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Notes about this constraint.</td>
</tr>
</tbody>
</table>
### ConnectorConstraint Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current ConnectorConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.7.2 ConnectorEnd

**public Class**

A ConnectorEnd contains information about a single end of a connector. A ConnectorEnd is accessed from the connector as either the ClientEnd or SupplierEnd.

Associated table in .EAP file: derived from t_connector

### ConnectorEnd Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation</td>
<td>Long</td>
<td>Read/Write. Aggregation as it applies to this end. Valid values are: 0 = None, 1 = Shared, 2 = Composite.</td>
</tr>
<tr>
<td>Alias</td>
<td>String</td>
<td>Read/Write. An optional alias for this connector end.</td>
</tr>
<tr>
<td>AllowDuplicates</td>
<td>Boolean</td>
<td>Read/Write. For multiplicities greater than 1, indicates that duplicate entries are possible.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>String</td>
<td>Read/Write. Cardinality associated with this end.</td>
</tr>
<tr>
<td>Constraint</td>
<td>String</td>
<td>Read/Write. A constraint that can be applied to this connector end.</td>
</tr>
<tr>
<td>Containment</td>
<td>String</td>
<td>Read/Write. Containment type applied to this connector end.</td>
</tr>
<tr>
<td>Derived</td>
<td>Boolean</td>
<td>Read/Write. Indicates that the value of this end is derived.</td>
</tr>
<tr>
<td>DerivedUnion</td>
<td>Boolean</td>
<td>Read/Write. Indicates the value of this role derived from the union of all roles that subset this.</td>
</tr>
<tr>
<td>End</td>
<td>String</td>
<td>Read only. The end this ConnectorEnd object applies to: Client or Supplier.</td>
</tr>
<tr>
<td>IsChangeable</td>
<td>String</td>
<td>Read/Write. Flag indicating whether this end is changeable or not. Values: frozen, addOnly or none.</td>
</tr>
<tr>
<td>IsNavigable</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating this end is navigable from the other end.</td>
</tr>
<tr>
<td>Navigable</td>
<td>String</td>
<td>Read/Write. Indicates whether this role of an association is navigable from the opposite classifier. Three values are valid: Navigable, Non-Navigable and Unspecified.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Ordering</td>
<td>Long</td>
<td>Read/Write. Ordering for this connector end.</td>
</tr>
</tbody>
</table>
**Attribute** | **Type** | **Notes**
--- | --- | ---
OwnedByClassifier | Boolean | Read/Write. Indicates this association end corresponds to an attribute on the opposite end of the association.
Qualifier | String | Read/Write. A qualifier that can apply to connector end.
Role | String | Read/Write. The connector end role.
RoleNote | String | Read/Write. Notes associated with the role of this connector end.
RoleType | String | Read/Write. The role type applied to this end of the connector.
Stereotype | String | Read/Write. The role type applied to this end of the connector.
StereotypeEx | String | Read/Write. All the applied stereotypes of the connector end in a comma-separated list.
TaggedValues | Private | Read only. Collection of RoleTag objects.
Visibility | String | Read/Write. Scope associated with this connector end. Valid types are: Public, Private, Protected and Package.

**ConnectorEnd 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 as an exception is thrown when an error occurs.
Update () | Boolean | Update the current ConnectorEnd object after modification or appending a new item. If false is returned, check the GetLastError function for more information.

**16.7.2.7.3 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 from the following list:

- Aggregation
- Association
- Collaboration
- Dependency
- Generalization
- Instantiation
- Nesting
- NoteLink
- Realisation
- Sequence
- Transition
- UseCase

Associated table in .EAP file: t_connector

**Connector Attributes**

| Attribute | Type | Notes |
--- | --- | ---
Alias | String | Read/Write. An optional alias for this connector.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientEnd</td>
<td>ConnectorEnd</td>
<td>Read only. A pointer to the ConnectorEnd object representing the source end of the relationship.</td>
</tr>
<tr>
<td>ClientID</td>
<td>Long</td>
<td>Read/Write. ElementID of the element at the source end of this connector.</td>
</tr>
<tr>
<td>Color</td>
<td>Long</td>
<td>Read/Write. Sets the color of the connector.</td>
</tr>
<tr>
<td>ConnectorGUID</td>
<td>Variant</td>
<td>Read only. A globally unique ID for the current connector. System generated.</td>
</tr>
<tr>
<td>ConnectorID</td>
<td>Long</td>
<td>Read only. Local identifier for the current connector. System generated.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Collection</td>
<td>Read only. Collection of constraint objects.</td>
</tr>
<tr>
<td>CustomProperties</td>
<td>Collection</td>
<td>Read only. Returns a collection of advanced properties associated with an element in the form of CustomProperty objects.</td>
</tr>
<tr>
<td>DiagramID</td>
<td>Long</td>
<td>Read/Write. The DiagramID of the connector.</td>
</tr>
<tr>
<td>Direction</td>
<td>String</td>
<td>Read/Write. Connector direction. Can be set to one of the following: Unspecified, Bi-Directional, Source -&gt; Destination, Destination -&gt; Source</td>
</tr>
<tr>
<td>EventFlags</td>
<td>String</td>
<td>Read/Write. Structure to hold a variety of flags concerned with event signaling on messages.</td>
</tr>
<tr>
<td>IsLeaf</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating connector is a leaf.</td>
</tr>
<tr>
<td>IsRoot</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating connector is a root.</td>
</tr>
<tr>
<td>IsSpec</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating connector is a specification.</td>
</tr>
<tr>
<td>MetaType</td>
<td>String</td>
<td>Read only: The connector’s domain-specific meta type, as defined by an applied stereotype from an MDG Technology.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The connector name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes about the connector.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Properties</td>
<td>Properties</td>
<td>Returns a list of specialized properties that apply to the connector that might 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.</td>
</tr>
<tr>
<td>RouteStyle</td>
<td>Long</td>
<td>Read/Write. The route style.</td>
</tr>
<tr>
<td>SequenceNo</td>
<td>Long</td>
<td>Read/Write. The SequenceNo of the connector.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>String</td>
<td>Read/Write. Sets or gets the stereotype for this connector end.</td>
</tr>
<tr>
<td>StereotypeEx</td>
<td>String</td>
<td>Read/Write. All the applied stereotypes of the connector in a comma-separated list.</td>
</tr>
<tr>
<td>StyleEx</td>
<td>String</td>
<td>Read/Write. Advanced style settings. Not currently used.</td>
</tr>
<tr>
<td>Subtype</td>
<td>String</td>
<td>Read/Write. A possible subtype to refine the meaning of the connector.</td>
</tr>
<tr>
<td>SupplierEnd</td>
<td>ConnectorEnd</td>
<td>Read only. A pointer to the ConnectorEnd object representing the target end of the relationship.</td>
</tr>
<tr>
<td>SupplierID</td>
<td>Long</td>
<td>Read/Write. ElementID of the element at the target end of this connector.</td>
</tr>
<tr>
<td>TaggedValues</td>
<td>Collection</td>
<td>Read only. Collection of ConnectorTag objects.</td>
</tr>
<tr>
<td>TransitionAction</td>
<td>String</td>
<td>Read/Write. See the Transition topic in the Enterprise Architect User Guide for appropriate values.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TransitionEvent</td>
<td>String</td>
<td>Read/Write. See the Transition topic in the Enterprise Architect User Guide for appropriate values.</td>
</tr>
<tr>
<td>TransitionGuard</td>
<td>String</td>
<td>Read/Write. See the Transition topic in the Enterprise Architect User Guide for appropriate values.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read/Write. Connector type. Valid types are held in the t_connectortypes table in the .EAP file.</td>
</tr>
<tr>
<td>VirtualInheritance</td>
<td>String</td>
<td>Read/Write. For Generalization, indicates if inheritance is virtual.</td>
</tr>
<tr>
<td>Width</td>
<td>Long</td>
<td>Read/Write. Specifies the width of the connector.</td>
</tr>
</tbody>
</table>

**Connector Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current ConnectorObject after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### 16.7.2.7.4 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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorID</td>
<td>Long</td>
<td>Read/Write. The local ID of the associated connector.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The tag or name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Descriptive notes associated with this Tagged Value.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>TagGUID</td>
<td>String</td>
<td>Read/Write. A globally unique ID for this Tagged Value.</td>
</tr>
<tr>
<td>TagID</td>
<td>Long</td>
<td>Read only. A local ID to identify the Tagged Value.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Read/Write. A value associated with the tag.</td>
</tr>
</tbody>
</table>

**ConnectorTag Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Update the current ConnectorTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>
16.7.2.7.5 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 create something that resembles the following (where con is a Connector Object):

Code fragment for accessing a RoleTag in VB.NET:

```vbnet
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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseClass</td>
<td>String</td>
<td>Read/Write. Indicates the role end; set to ASSOCIATION_SOURCE or ASSOCIATION_TARGET.</td>
</tr>
<tr>
<td>ElementGUID</td>
<td>String</td>
<td>Read/Write. GUID of the connector with which this role tag is associated.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>PropertyGUID</td>
<td>String</td>
<td>Read/Write. A system generated GUID to identify the Tagged Value.</td>
</tr>
<tr>
<td>Tag</td>
<td>String</td>
<td>Read/Write. The actual tag name.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Read/Write. The value associated with this tag.</td>
</tr>
</tbody>
</table>

RoleTag Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the RoleTag after changes or on initial creation. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.8 Diagram

public Package

The Diagram package has information on a diagram and on DiagramObjects and DiagramLinks, which are the instances of elements within a diagram.
16.7.2.8.1 Diagram

**public Class**

A Diagram corresponds to a single Enterprise Architect diagram. It is accessed through the Package Diagrams collection and in turn contains a collection of diagram objects and diagram connectors. Adding to the DiagramObjects collection adds 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
Note:

Use the Analysis type for a Collaboration Diagram.

Associated table in .EAP file: t_diagram

Diagram Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>String</td>
<td>Read/Write. The author.</td>
</tr>
<tr>
<td>CreatedDate</td>
<td>Date</td>
<td>Read/Write. The date the diagram was created.</td>
</tr>
<tr>
<td>cx</td>
<td>Long</td>
<td>Read/Write. The X dimension of the diagram (default is 800).</td>
</tr>
<tr>
<td>cy</td>
<td>Long</td>
<td>Read/Write. The Y dimension of the diagram (default is 1100).</td>
</tr>
<tr>
<td>DiagramGUID</td>
<td>Variant</td>
<td>Read/Write. A globally unique ID for this diagram.</td>
</tr>
<tr>
<td>DiagramID</td>
<td>Long</td>
<td>Read only. A local ID for the diagram.</td>
</tr>
<tr>
<td>DiagramLinks</td>
<td>Collection</td>
<td>Read only. A list of DiagramLink objects, each containing information about the display characteristics of a connector in a diagram.</td>
</tr>
<tr>
<td>DiagramObjects</td>
<td>Collection</td>
<td>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.</td>
</tr>
<tr>
<td>ExtendedStyle</td>
<td>String</td>
<td>Read/Write. An extended style attribute.</td>
</tr>
<tr>
<td>HighlightImports</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate elements from other packages should be highlighted.</td>
</tr>
<tr>
<td>IsLocked</td>
<td>Boolean</td>
<td>Read/Write. Flag indicating whether this diagram is locked or not.</td>
</tr>
<tr>
<td>MetaType</td>
<td>String</td>
<td>Read only: The diagram's domain-specific meta type, as defined by an MDG Technology.</td>
</tr>
<tr>
<td>ModifiedDate</td>
<td>Variant</td>
<td>Read/Write. The date the diagram was last modified.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Read/Write. The diagram name.</td>
</tr>
<tr>
<td>Notes</td>
<td>String</td>
<td>Read/Write. Set/retrieve notes for this diagram.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Orientation</td>
<td>String</td>
<td>Read/Write. Page orientation: P for Portrait or L for Landscape.</td>
</tr>
<tr>
<td>PackageID</td>
<td>Long</td>
<td>Read/Write. An ID of the package that this diagram belongs to.</td>
</tr>
<tr>
<td>ParentID</td>
<td>Long</td>
<td>Read/Write. An optional ID of an element that 'owns' this diagram; e.g. a Sequence diagram owned by a Use Case.</td>
</tr>
<tr>
<td>Scale</td>
<td>Long</td>
<td>Read/Write. The zoom scale (default is 100).</td>
</tr>
<tr>
<td>SelectedConnector</td>
<td>Connector</td>
<td>Read/Write. The currently selected connector on this diagram. Null if there is no currently selected diagram.</td>
</tr>
<tr>
<td>SelectedObjects</td>
<td>Collection</td>
<td>Read only. Gets a collection representing the currently selected elements on the diagram. Can remove objects from this collection to deselect them.</td>
</tr>
</tbody>
</table>
### Attribute Types

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowDetails</td>
<td>Long</td>
<td>Read/Write. Flag to indicate Diagram Details text should be shown. 1 = Show, 0 = Hide.</td>
</tr>
<tr>
<td>ShowPackageContents</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate package contents should be shown in the current diagram.</td>
</tr>
<tr>
<td>ShowPrivate</td>
<td>Boolean</td>
<td>Read/Write. Flag to show or hide Private features.</td>
</tr>
<tr>
<td>ShowProtected</td>
<td>Boolean</td>
<td>Read/Write. Flag to show or hide Protected features.</td>
</tr>
<tr>
<td>ShowPublic</td>
<td>Boolean</td>
<td>Read/Write. Flag to show or hide Public features.</td>
</tr>
<tr>
<td>Stereotype</td>
<td>String</td>
<td>Read/Write. Sets or gets the stereotype for this diagram.</td>
</tr>
<tr>
<td>StyleEx</td>
<td>String</td>
<td>Read/Write. Advanced style settings.</td>
</tr>
<tr>
<td>Swimlanes</td>
<td>String</td>
<td>Read/Write. Information on swimlanes contained in the diagram. Please note that this property is superseded by SwimlaneDef.</td>
</tr>
<tr>
<td>SwimlaneDef</td>
<td>SwimlaneDef</td>
<td>Read/Write. Information on swimlanes contained in the diagram.</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>Read only. The diagram type. See the <code>t_diagramtypes</code> table in the .EAP file for more information.</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>Read/Write. The version of the diagram.</td>
</tr>
</tbody>
</table>

### Diagram Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyGroupLock</td>
<td>Boolean</td>
<td>Applies a group lock to this diagram object, for the specified group, on behalf of the current user. Throws an exception if the operation fails. Use GetLastError() to retrieve error information. Parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· aGroupName: String - the name of the user group for which to set the group lock.</td>
</tr>
<tr>
<td>ApplyUserLock ()</td>
<td>Boolean</td>
<td>Applies a user lock to this diagram object, for the current user. Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</td>
</tr>
<tr>
<td>GetLastError ()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>ReleaseLock ()</td>
<td>Boolean</td>
<td>Releases a group lock or user lock on this diagram object. Throws an exception if the operation fails. Use GetLastError() to retrieve error information.</td>
</tr>
<tr>
<td>ReorderMessages ()</td>
<td>Void</td>
<td>Resets the display order of Sequence and Collaboration messages. Typically used after inserting or deleting messages in the diagram.</td>
</tr>
<tr>
<td>ShowAsElementList</td>
<td>Boolean</td>
<td>Toggles the diagram display between diagram format and Element List depending on the value of ShowAsList. If Persist is set, the display format is written to the database so the diagram always opens in that format (diagram or list). Otherwise, the display format falls back to the default (diagram) once the display is closed.</td>
</tr>
<tr>
<td>Update ()</td>
<td>Boolean</td>
<td>Updates this diagram object after modification or appending a new item. If false is returned, use GetLastError() to retrieve error information.</td>
</tr>
</tbody>
</table>
16.7.2.8.2 DiagramLinks

public Class

A DiagramLink is an object that holds display information about a connector between two elements in a specific diagram. It includes, for example, the custom points and display appearance. Accessed from the Diagram DiagramLinks collection.

Associated table in .EAP file: t_diagramlinks

DiagramLinks Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorID</td>
<td>Long</td>
<td>Read/Write. The ID of the associated connector.</td>
</tr>
<tr>
<td>DiagramID</td>
<td>Long</td>
<td>Read/Write. The local ID for the associated diagram.</td>
</tr>
<tr>
<td>Geometry</td>
<td>String</td>
<td>Read/Write. The geometry associated with the current connector in this diagram.</td>
</tr>
<tr>
<td>InstanceID</td>
<td>Long</td>
<td>Read only attribute. Holds the connector identifier for the current model.</td>
</tr>
<tr>
<td>IsHidden</td>
<td>Boolean</td>
<td>Read/Write. Flag to indicate if this item is hidden or not.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Path</td>
<td>String</td>
<td>Read/Write. The path of the connector in this diagram.</td>
</tr>
<tr>
<td>Style</td>
<td>String</td>
<td>Read/Write. Additional style information; e.g. color, thickness.</td>
</tr>
</tbody>
</table>

DiagramLinks Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current DiagramLink object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

16.7.2.8.3 DiagramObjects

public Class

The DiagramObjects collection holds a list of element IDs and presentation information that indicates what is displayed in a diagram and how it is shown.

Associated table in .EAP file: t_diagramobjects

DiagramObjects Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom</td>
<td>Long</td>
<td>Read/Write. The bottom position of the element.</td>
</tr>
<tr>
<td>DiagramID</td>
<td>Long</td>
<td>Read/Write. The ID of the associated diagram (long).</td>
</tr>
<tr>
<td>ElementID</td>
<td>Long</td>
<td>Read/Write. The ElementID of the object instance in this diagram.</td>
</tr>
<tr>
<td>InstanceID</td>
<td>Long</td>
<td>Read/Write. Read only attribute. Holds the connector identifier for the current model.</td>
</tr>
<tr>
<td>Left</td>
<td>Long</td>
<td>Read/Write. The left position of the element.</td>
</tr>
</tbody>
</table>
### Attribute Types and Notes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Right</td>
<td>Long</td>
<td>Read/Write. The right position of the element.</td>
</tr>
<tr>
<td>Sequence</td>
<td>Long</td>
<td>Read/Write. The sequence position when loading into diagram (affects Z order). The Z-order is one-based and the lowest value is in the foreground.</td>
</tr>
<tr>
<td>Style</td>
<td>Variant</td>
<td>Write only (reading this value gives undefined results). Style information for this object.  See Setting the Style below for more information.</td>
</tr>
<tr>
<td>Top</td>
<td>Long</td>
<td>Read/Write. The top position of the element.</td>
</tr>
</tbody>
</table>

### DiagramObjects Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLastError()</td>
<td>String</td>
<td>Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</td>
</tr>
<tr>
<td>Update()</td>
<td>Boolean</td>
<td>Update the current DiagramObject object after modification or appending a new item. If false is returned, check the GetLastError function for more information.</td>
</tr>
</tbody>
</table>

### Setting The Style

The Style attribute is used for setting the appearance of a DiagramObject. 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. For example:

```
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
Next
```

### 16.7.2.8.4 SwimlaneDef

A SwimlaneDef object makes available attributes relating to a single row or column in a list of swimlanes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Boolean</td>
<td>Read/Write. Show the title text in bold.</td>
</tr>
<tr>
<td>FontColor</td>
<td>Long</td>
<td>Read/Write. RGB color used to draw the titles.</td>
</tr>
<tr>
<td>HideClassifier</td>
<td>Boolean</td>
<td>Read/Write. Removes any classifier from title display.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>HideNames</td>
<td>Boolean</td>
<td>Read/Write. Set to true to hide the swimlane titles.</td>
</tr>
<tr>
<td>LineColor</td>
<td>Long</td>
<td>Read/Write. RGB color used to draw swimlane borders.</td>
</tr>
<tr>
<td>LineWidth</td>
<td>Long</td>
<td>Read/Write. Width of line, in pixels, used to draw swimlanes.</td>
</tr>
<tr>
<td>Locked</td>
<td>Boolean</td>
<td>Read/Write. If set to true, disables user modification of the swimlanes via the diagram.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Orientation</td>
<td>String</td>
<td>Read/Write. Indication of whether the swimlanes are vertical or horizontal.</td>
</tr>
<tr>
<td>ShowInTitleBar</td>
<td>Boolean</td>
<td>Read/Write. Enables vertical swimlane titles to be shown in title bar.</td>
</tr>
<tr>
<td>Swimlanes</td>
<td>Swimlanes</td>
<td>Read/Write. A list of individual swimlanes.</td>
</tr>
</tbody>
</table>

### 16.7.2.8.5 Swimlanes

A Swimlanes object is attached to a diagram's SwimlaneDef object and provides a mechanism to access individual swimlanes.

#### Swimlanes Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Read/Write. Gives the number of swimlanes.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>

#### Swimlanes Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add (string Title, long Width)</td>
<td>Swimlane</td>
<td>Adds a new swimlane to the end of the list. Returns a swimlane object representing the newly added entry. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Title: String - The title text that appears at the top of the swimlane. Can be the same as an existing swimlane title.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Width: Long - The width of the swimlane in pixels.</td>
</tr>
<tr>
<td>Delete (object Index)</td>
<td>Void</td>
<td>Deletes a selected swimlane.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 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.</td>
</tr>
<tr>
<td>DeleteAll ()</td>
<td>Void</td>
<td>Removes all swimlanes.</td>
</tr>
<tr>
<td>Insert (long Index, string Title, long Width)</td>
<td>Swimlane</td>
<td>Inserts a swimlane at a specific position. Returns a swimlane object representing the newly added entry. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Index: Long - The zero-based index of the existing Swimlane before which this new entry is inserted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Title: String - The title text which appears at the top of the swimlane. Can be the same as an existing swimlane title.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Width: Long - The width of the swimlane in pixels.</td>
</tr>
<tr>
<td>Items (object collection)</td>
<td>Swimlane</td>
<td>Accesses an individual swimlane.</td>
</tr>
</tbody>
</table>
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td></td>
<td>Parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Index: Object - Either a string representing the title text or an integer representing the zero-based index of the swimlane to get.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the string matches more than one swimlane title, the first matching swimlane is returned.</td>
</tr>
</tbody>
</table>

#### 16.7.2.8.6 Swimlane

A Swimlane object makes available attributes relating to a single row or column in a list of swimlanes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackColor</td>
<td>Long</td>
<td>Read/Write. The swimlane is filled with this RGB color.</td>
</tr>
<tr>
<td>ClassifiedGuid</td>
<td>String</td>
<td>Read/Write. The GUID of the classifier Class. This can be obtained from the corresponding Element object via the <code>ElementGUID</code> property.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
<tr>
<td>Title</td>
<td>String</td>
<td>Read/Write. Text at the head of the swimlane.</td>
</tr>
<tr>
<td>Width</td>
<td>Long</td>
<td>Read/Write. The width of the swimlane in pixels.</td>
</tr>
</tbody>
</table>

#### 16.7.2.9 Project Interface

**public Package**

The `Enterprise.Architect.Project` interface. This is the XML-based interface to Enterprise Architect elements; it also includes some utility functions. You can get a pointer to this interface using the `Repository.GetProjectInterface` method.

#### 16.7.2.9.1 Project

**public Class**

The Project interface can be accessed from the Repository using `GetProjectInterface()`. The returned interface provides access to the XML-based Enterprise Architect Automation Interface. Use this interface to get XML for the various internal elements and to run some utility functions to perform tasks such as load diagrams or run reports.
Note:
These methods all require input GUIDs in XML format; use GUIDtoXML to change the Enterprise Architect GUID to an XML GUID.

Project Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectType</td>
<td>ObjectType</td>
<td>Read only. Distinguishes objects referenced through a Dispatch interface.</td>
</tr>
</tbody>
</table>

Project Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| CreateBaseline (string PackageGUID, string Version, string Notes) | Boolean | Creates a Baseline of a specified package. Parameters:  
· PackageGUID: String - the GUID (in XML format) of the package to Baseline.  
· Version: String - the version of the Baseline.  
· Notes: String - any notes concerning the Baseline. |

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| DoBaselineCompare (string PackageGUID, string Baseline, string ConnectString) | String | Performs a Baseline comparison using the supplied package GUID and Baseline GUID (obtained in the result list from GetBaselines). Optionally you can include the connection string required to find the Baseline if it exists in a different model file. This method returns a log file of the status of all elements found and compared in the difference procedure. You can use this log information as input to DoBaselineMerge - automatically merging information from the Baseline. Parameters:  
· PackageGUID: String - the GUID (in XML format) of the package to run the comparison on.  
· Baseline: String - the GUID (in XML format) of the Baseline to run the comparison on.  
· ConnectString: String - the location of the external EAP file or DBMS to extract the Baseline from. |

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
</table>
| DoBaselineMerge (string PackageGUID, string Baseline, string MergeInstructions, string ConnectString) | String | Performs a batch merge based on instructions contained in an XML file (MergeInstructions). You can supply an optional connection string if the Baseline is located in another model. Parameters:  
· PackageGUID: String - the GUID (in XML format) of the package to merge the Baseline into.  
· Baseline: String - the GUID of the Baseline (in XML format) to merge into the package.  
· MergeInstructions: String - the file containing the GUID of each differenced item from the XML difference log returned by DoBaselineCompare().  
· ConnectString: String - the location of the EAP file or DBMS to get the Baseline from, if not in the same model as the package. In the MergeInstructions file, each MergeItem node supplies the GUID of a differenced item from the XML |
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference log. As the merge is uni-directional and actioned in only one possible way, no additional arguments are required. Enterprise Architect chooses the correct procedure based on the Difference results.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Merge&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;MergeItem guid=&quot;{XXXXXX}&quot; /&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;MergeItem guid=&quot;{XXXXXX}&quot; /&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/Merge&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alternatively, you can supply a single <code>MergeItem</code> with a GUID of <code>RestoreAll</code>. In this case, Enterprise Architect batch-processes ALL differences.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;Merge&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;MergeItem guid=&quot;RestoreAll&quot; /&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/Merge&gt;</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EnumDiagramElements (string DiagramGUID)</th>
<th>protected abstract: String</th>
<th>Gets an XML list of all elements in a diagram. Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagramGUID: String - the GUID (in XML format) of the diagram to get elements for.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EnumDiagrams (string PackageGUID)</th>
<th>protected abstract: String</th>
<th>Gets an XML list of all diagrams in a specified package. Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGUID: String - the GUID (in XML format) of the package to list diagrams for.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EnumElements (string PackageGUID)</th>
<th>protected abstract: String</th>
<th>Gets an XML list of elements in a specified package. Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGUID: String - the GUID (in XML format) of the package to get a list of elements for.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EnumLinks (string ElementGUID)</th>
<th>protected abstract: String</th>
<th>Gets an XML list of connectors for a specified element. Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElementGUID: String - the GUID (in XML format) of the element to get all associated connectors for.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EnumPackages (string PackageGUID)</th>
<th>protected abstract: String</th>
<th>Gets an XML list of child packages inside a parent package. Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageGUID: String - the GUID (in XML format) of the parent package.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| EnumProjects () | protected abstract: String | Gets a list of projects in the current file; corresponds to `Models` in `Repository`. |

| EnumViews () | protected abstract: String | Enumerates the Views for a project. Returned as an XML document. |

<table>
<thead>
<tr>
<th>EnumViewEx (string ProjectGUID)</th>
<th>protected abstract: String</th>
<th>Gets a list of Views in the current project. Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectGUID: String - the GUID (in XML format) of the project to get views for.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exit ()</th>
<th>protected abstract: String</th>
<th>Exits the current instance of Enterprise Architect; this function is maintained for backward compatibility and should never be called.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Architect automatically exits when you are no longer using any of the provided objects.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| ExportPackageXMI (string PackageGUID, enumXMIType XMIType, long DiagramXML, long DiagramImage, long FormatXML, long UseDTD, string FileName) | protected abstract: String | Exports XMI for a specified package. Parameters:  
- PackageGUID: String - the GUID (in XML format) of the package to be exported.  
- XMIType: EnumXMIType - specifies the XMI type and version information; see [XMIType Enum](#) for accepted values.  
- DiagramXML: Long - true if XML for diagrams is required; accepted values:  
  - 0 = Do not export diagrams  
  - 1 = Export diagrams  
  - 2 = Export diagrams along with alternate images.  
- DiagramImage: Long - the 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.  
- FormatXML: Long - true if XML output should be formatted prior to saving.  
- UseDTD: Long - true if a DTD should be used.  
- FileName: String - the filename to output to. |
| GenerateClass (string ElementGUID, string ExtraOptions) | Boolean | Generates the code for a single Class. Parameters:  
- ElementGUID: String - the GUID (in XML format) of the element to generate.  
- ExtraOptions: String - enables extra options to be given to the command; currently unused. |
| GeneratePackage (string PackageGUID, string ExtraOptions) | Boolean | Generates the code for all Classes within a package. Parameters:  
- PackageGUID: String - the GUID (in XML format) of the package to generate.  
- ExtraOptions: String - enables extra options to be given to the command; currently enables:  
  - 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:\ |
| GenerateXSD (string PackageGUID, string FileName, string Encoding, string Options) | Boolean | Creates an XML schema for this GenerateXSD. Returns true on success. Parameters:  
- PackageGUID: String - the GUID (in XML format) of the package.  
- FileName: String - target filepath.  
- Encoding: String - the XML encoding for the code page instruction.  
- Options: String - enables extra options to be given to the command; currently enables:  
  - GenGlobalElement - turn the generation of global elements for all global ComplexTypes |
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBaselines (string PackageGUID, string ConnectString)</td>
<td>String</td>
<td>Returns a list (in XML format) of Baselines associated with the supplied package GUID. Optionally, you can provide a connection string to get Baselines from the same package, but located in a different model file (or DBMS). Parameters:  &lt;ul&gt;  + PackageGUID: String - the GUID (in XML format) of the package to get Baselines for.  + ConnectString: String - the location of the EAP file or DBMS to get the Baselines from, if not in the same model as the package.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetDiagram (string DiagramGUID)</td>
<td>protected abstract: String</td>
<td>Gets diagram details, in XML format. Parameters:  &lt;ul&gt;  + DiagramGUID: String - the GUID (in XML format) of the diagram to get details for.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElement (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets XML for the specified element. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element to retrieve XML for.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementConstraints (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets constraints for an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementEffort (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets efforts for an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementFiles (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets metrics for an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementMetrics (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets files for an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementProblems (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets a list of issues (problems) associated with an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementProperties (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets Tagged values for an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>GetElementRequirements (string ElementGUID)</td>
<td>protected abstract: String</td>
<td>Gets a list of requirements for an element, in XML format. Parameters:  &lt;ul&gt;  + ElementGUID: String - the GUID (in XML format) of the element.  &lt;/ul&gt;</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetElementResources (string ElementGUID)</td>
<td>protected</td>
<td>Gets a list of resources for an element, in XML format.</td>
</tr>
<tr>
<td>GetElementRisks (string ElementGUID)</td>
<td>protected</td>
<td>Gets a list of risks associated with an element, in XML format.</td>
</tr>
<tr>
<td>GetElementScenarios (string ElementGUID)</td>
<td>protected</td>
<td>Gets a list of scenarios for an element, in XML format.</td>
</tr>
<tr>
<td>GetElementTests (string ElementGUID)</td>
<td>protected</td>
<td>Gets a list of tests for an element, in XML format.</td>
</tr>
<tr>
<td>GetLastError ()</td>
<td>protected</td>
<td>Returns a string value describing the most recent error that occurred</td>
</tr>
<tr>
<td>GetLink (string LinkGUID)</td>
<td>protected</td>
<td>Gets connector details, in XML format.</td>
</tr>
<tr>
<td>GUIDtoXML (string GUID)</td>
<td>String</td>
<td>Changes an internal GUID to the form used in XML.</td>
</tr>
<tr>
<td>ImportDirectory (string PackageGUID, string Language, string DirectoryPath, string ExtraOptions)</td>
<td>Boolean</td>
<td>Imports a source code directory into the model.</td>
</tr>
<tr>
<td>ImportFile (string PackageGUID, string Language, string FileName, string ExtraOptions)</td>
<td>Boolean</td>
<td>Imports an individual file or binary module into the model, in a</td>
</tr>
</tbody>
</table>

© 1998-2008 Sparx Systems Pty Ltd
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method</strong></td>
<td><strong>Type</strong></td>
<td><strong>Notes</strong></td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use the value &quot;DNPE&quot; to import a binary module. This imports a .Net assembly or Java .class file, but not a .jar file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Filename: String - specifies the path where the code or module is found on the computer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• ExtraOptions: String - enables extra options to be given to the command; currently unused.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImportPackageXMI (string PackageGUID, string Filename, long ImportDiagrams, long StripGUID)</td>
<td>String</td>
<td>Imports an XMI file at a point in the tree. Parameters:</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If the String is of type filename it is interpreted as a source file, otherwise the String is imported as XML text.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• ImportDiagrams: Long.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• StripGUID: Long - boolean value to indicate whether to replace the element UniqueIDs on import; if stripped, then a package could be imported twice into Enterprise Architect, as two different versions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayoutDiagram (string DiagramGUID, long LayoutStyle)</td>
<td>Boolean</td>
<td>Deprecated. It is recommended that LayoutDiagramEx is used instead. Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams. Parameters:</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DiagramGUID: String - the GUID (in XML format) of the diagram to lay out.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• LayoutStyle: Long - always ignored.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayoutDiagramEx (string DiagramGUID, long LayoutStyle, long Iterations, long LayerSpacing, long ColumnSpacing, boolean SaveToDiagram)</td>
<td>Boolean</td>
<td>Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams. Parameters:</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DiagramGUID: String - the GUID (in XML format) of the diagram to lay out.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• LayoutStyle: Long - the layout style.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Iterations: Long - the number of layout iterations the Layout process should take to perform cross reduction (Default value = 4).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• LayerSpacing: Long - the per-element layer spacing the Layout process shall use (Default value = 20).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• ColumnSpacing: Long - the per-element column spacing the Layout process shall use (Default value = 20).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• SaveToDiagram: Boolean - specifies whether or not Enterprise Architect should save the supplied diagram.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LoadControlledPackage (string PackageGUID)</td>
<td>String</td>
<td>Loads a package that has been marked and configured as controlled. The filename details are stored in the package control data. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PackageGUID: String - the GUID (in XML format) of the package to load.</td>
</tr>
<tr>
<td>LoadDiagram (string DiagramGUID)</td>
<td>protected abstract: Boolean</td>
<td>Loads a diagram by its GUID. Parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DiagramGUID: String - the GUID (in XML format) of the diagram to load; if you retrieve the GUID using the Diagram interface, use the GUIDtoXML function to convert it to XML format.</td>
</tr>
<tr>
<td>LoadProject (string FileName)</td>
<td>protected abstract: Boolean</td>
<td>Loads an Enterprise Architect project file. Do not use this method if you have accessed the Project interface from the Repository, which has already loaded a file. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FileName: String - the name of the project file to load.</td>
</tr>
<tr>
<td>PutDiagramImageOnClipboard (string DiagramGUID, long Type)</td>
<td>protected abstract: Boolean</td>
<td>Copies an image of the specified diagram to the clipboard. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DiagramGUID: String - the GUID (in XML format) of the diagram to copy. Type: Long - the file type. Parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If Type = 0 then it is a metafile. If Type = 1 then it uses the file type from the name extension (i.e., .bmp, .jpg, .gif, .png, .tga)</td>
</tr>
</tbody>
</table>

LayoutStyle accepts the following options (also see ConstLayoutStyles Enum):

- Default Options: lsDiagramDefault lsProgramDefault.
- Cycle Removal Options: lsCycleRemoveGreedy lsCycleRemoveDFS.
- Layering Options: lsLayeringLongestPathSink lsLayeringLongestPathSource lsLayeringOptimalLinkLength.
- Initialize Options: lsInitializeNaive lsInitializeDFSOOut lsInitializeDFSIn.
- Crossing Reduction Option: lsCrossReduceAggressive.
- Layout Options - Direction lsLayoutDirectionUp lsLayoutDirectionDown lsLayoutDirectionLeft lsLayoutDirectionRight.
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PutDiagramImageToFile (string Diagram GUID, string FileName, long Type)</td>
<td>protected abstract: Boolean</td>
<td>Saves an image of the specified diagram to file. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· DiagramGUID: String - the GUID (in XML format) of the diagram to save.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· FileName: String - the name of the file to save the diagram into.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Type: Long - the file type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· If type = 0 then it is a metafile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· If type = 1 then it uses the file type from the name extension (i.e., .bmp, .jpg, .gif, .png, .tga)</td>
</tr>
<tr>
<td>ReloadProject ()</td>
<td>protected abstract: Boolean</td>
<td>Reloads the current project. This is a convenient method to refresh the current loaded project (in case of outside changes to the .EAP file).</td>
</tr>
<tr>
<td>RunReport (string PackageGUID, string TemplateName, string Filename)</td>
<td>protected abstract: Void</td>
<td>Runs a named RTF report. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· PackageGUID: String - the GUID (in XML format) of the package to run the report on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· TemplateName: String - the RTF report template to use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· FileName: String - the file name to store the generated report in.</td>
</tr>
<tr>
<td>RunHTMLReport (string PackageGUID, string ExportPath, string ImageFormat, string Style, string Extension)</td>
<td>String</td>
<td>Runs an HTML report (same as Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· PackageGUID: String - the GUID (in XML format) of the package to run the report on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· ExportPath: String - the file name to store the generated report in.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· ImageFormat: String.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Style: String.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Extension: String.</td>
</tr>
<tr>
<td>SaveControlledPackage (string PackageGUID)</td>
<td>String</td>
<td>Saves a package that has been configured as a controlled package, to XMI. Only the package GUID is required. Enterprise Architect picks the rest up from the package control information. Parameter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· PackageGUID: String - the GUID (in XML format) of the package to save.</td>
</tr>
<tr>
<td>SaveDiagramImageToFile (string Filename)</td>
<td>protected abstract: String</td>
<td>Saves a diagram image of the current diagram to file. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· FileName: String - the filename of the image to save.</td>
</tr>
<tr>
<td>ShowWindow (long Show)</td>
<td>protected abstract: Void</td>
<td>Shows or hides the Enterprise Architect User Interface. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Show: Long.</td>
</tr>
<tr>
<td>SynchronizeClass (string ElementGUID, string ExtraOptions)</td>
<td>Boolean</td>
<td>Synchronizes a Class with the latest source code. Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· ElementGUID: String - the GUID (in XML format) of the element to update from code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· ExtraOptions: String - enables extra options to be</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>SynchronizePackage (string PackageGUID, string ExtraOptions)</td>
<td>Boolean</td>
<td>Synchronizes each Class in a package with the latest source code. Parameters:</td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td>TransformElement (string TransformName, string ElementGUID, string TargetPackageGUID, string ExtraOptions)</td>
<td>Boolean</td>
<td>Transforms an element into a package. Parameters:</td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td>TransformPackage (string TransformName, string SourcePackage, string TargetPackage, string ExtraOptions)</td>
<td>Boolean</td>
<td>Runs a transformation on the contents of a package. Parameters:</td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td>XMLtoGUID (string GUID)</td>
<td>String</td>
<td>Changes a GUID in XML format to the form used inside Enterprise Architect. Parameters:</td>
</tr>
</tbody>
</table>

16.7.2.10 Code Samples

This topic contains various code examples indicating how to use the Automation Interface, written in VB Dot Net:

- Open the Repository
- Iterate Through a .EAP File
- Add and Manage Packages
- Add and Manage Elements
- Add a Connector
- Add and Manage Diagrams
- Add and Delete Features
- Element Extras
- Repository Extras
- Stereotypes
- Work with Attributes
- Work with Methods

© 1998-2008 Sparx Systems Pty Ltd
16.7.2.10.1 Open the Repository

public Object

"An example of how to open an Enterprise Architect repository
"in VB.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 any way required
'DumpModel

"close the repository and tidy up
m_Repository.Exit()

m_Repository = Nothing

....catch e as exception
    Console.WriteLine(e)
End try
End Sub
end Class

16.7.2.10.2 Iterate Through a .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.Elements.Count - 1
        DumpPackage(Indent + " "", Package.Elements.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

16.7.2.10.3 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())
        End If
        package.Element.Stereotype = "system" 
        package.Update

        "for testing purposes just delete the
        "newly created Model and its contents
        m_Repository.Models.Delete(idx)
    End If
Next
End Sub

16.7.2.10.4 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, Enterprise Architect loads 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)
    End If
Next
End Sub
16.7.2.10.5 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 must 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

16.7.2.10.6 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.Diagnostics.AddNew("Logical Diagram","Logical")
If not diagram.Update Then
    Console.WriteLine(diagram.GetLastError)
End if

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

16.7.2.10.7 Add and Delete Features

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 must
'substitute a valid ID from your model
element = m_Repository.GetElementByID(246)

"create a new method
method = element.Methods.AddNew("newMethod", "int")
method.Update

'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)
    End If
Next

'create an attribute
attribute = element.attributes.AddNew("NewAttribute", "int")
attribute.Update

'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)
    End If
Next
16.7.2.10.8 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

'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())
End if

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)
    End if
Next

'efforts
o = element.Efforts.AddNew("Appended","Type")
If not o.Update Then
    Console.WriteLine("Efforts error:" + o.GetLastError())
End if

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)
    End if
Next

'Risks
o = element.Risks.AddNew("Appended","Type")
If not o.Update Then
    Console.WriteLine("Risks error:" + o.GetLastError())
End if

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)
    End if
Next

'Metrics
o = element.Metrics.AddNew("Appended","Change")
If not o.Update Then
    Console.WriteLine("Metrics error:" + o.GetLastError())
End if

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)
    End if
End if
Next

'TaggedValues
o = element.TaggedValues.AddNew("Appended","Change")
If not o.Update Then
    Console.WriteLine("TaggedValues error:" + o.GetLastError())
End if
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)
    End if
End if
Next

'Scenarios
o = element.Scenarios.AddNew("Appended","Change")
If not o.Update Then
    Console.WriteLine("Scenarios error:" + o.GetLastError())
End if
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)
    End if
End if
Next

'Files
o = element.Files.AddNew("MyFile","doc")
If not o.Update Then
    Console.WriteLine("Files error:" + o.GetLastError())
End if
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)
    End if
End if
Next

'Tests
o = element.Tests.AddNew("TestPlan","Load")
If not o.Update Then
    Console.WriteLine("Tests error:" + o.GetLastError())
End if
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)
    End if
End if
Next

'Defect
o = element.Issues.AddNew("Broken","Defect")
If not o.Update Then
    Console.WriteLine("Issues error:" + o.GetLastError())
End if
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)
    End if
End if
Next

'Change
o = element.Issues.AddNew("Change","Change")
If not o.Update Then
Console.WriteLine("Issues error:" + o.GetLastError())
End if

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)
    End if
  Next

Next

catch e as exception
  Console.WriteLine(element.Methods.GetLastError())
  Console.WriteLine(e)
End try

End Sub

16.7.2.10.9 Repository Extras

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())
  End if
  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")
  End if
Next

"tasks
  o = m_Repository.Tasks.AddNew("Task 1","Task type")
  If(o.Update=false) Then
    Console.WriteLine ("error - " + o.GetLastError())
  End if
  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")
  End if
Next

"glossary
  o = m_Repository.Terms.AddNew("Term 1","business")
  If(o.Update=false) Then
    Console.WriteLine ("error - " + o.GetLastError())
  End if
  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 = "Term 1") then
    m_Repository.Terms.DeleteAt(idx,false)
    Console.WriteLine("Delete Terms")
  End if
Next
'authors
    o = m_Repository.Authors.AddNew("Joe B","Writer")
    If(o.Update=false) Then
        Console.WriteLine (o.GetLastError())
    End if
    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 = "Joe B") then
        m_Repository.authors.DeleteAt(idx,false)
    End if
Next

o = m_Repository.Clients.AddNew("Joe Sphere","Client")
If(o.Update=false) Then
    Console.WriteLine (o.GetLastError())
End if
    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)
    End if
Next

o = m_Repository.Resources.AddNew("Joe Worker","Resource")
If(o.Update=false) Then
    Console.WriteLine (o.GetLastError())
End if
    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)
    End if
Next

End Sub

16.7.2.10.10 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 you 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)
        End if
    Next

© 1998-2008 Sparx Systems Pty Ltd
16.7.2.10.11 Work 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.Attribures.GetAt(idx).Name)
    o = element.Attribures.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
  Next
  For idx2 = 0 to o.TaggedValues.Count - 1
    t = o.TaggedValues.GetAt(idx2)
    If(t.Name = "Type2") Then
      "Console.WriteLine("deleteing")
      o.TaggedValues.DeleteAt(idx2, true)
    End If
  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.Attribures.GetAt(idx).Name = "m_Tootle") Then
    Console.WriteLine("delete attribute")
    element.Attribures.DeleteAt(idx, false)
  End If
  Next
  catch e as exception
    Console.WriteLine(element.Attribures.GetLastError())
    Console.WriteLine(e)
  End try
End Sub

16.7.2.10.12 Work with Methods

public Object

"An 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)
        End If
    Next
    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)
    End If
    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
catch e as exception
    Console.WriteLine(element.Methods.GetLastError())
    Console.WriteLine(e)
End try

End Sub
Part XVI
17 Glossary of Terms

This topic provides a detailed glossary for Enterprise Architect.
17.1 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 connector 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 controls 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

active object
An object that owns a thread and can initiate control activity. An instance of active Class.
See also: active class, thread

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 work flow.

activity diagram
A diagram used to model procedural type application development for system design through to modeling business processes in organizational structures and work flow.

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
A binding for a parameter that resolves to a run-time instance.
Synonym: argument
Contrast: (formal) parameter

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**
A diagram 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 requirements.

**analysis time**
Refers to something that occurs during an analysis phase of the software development process.
*See also: modeling time, run time, compile time*
Contrast: design 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: (formal) parameter

**artifact**
A physical piece of information that is used or produced by a business or development process. Examples of Artifacts include models, source files, scripts, and binary executable files. An Artifact can constitute the implementation of a deployable component.
*Synonym: product*
Contrast: component

**assembly**
A connector that 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.
*See also: link*

**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.
*See also: class*

**association end**
The endpoint of an Association, which connects the Association to a classifier.
*See also: classifier, link end*

**attribute**
A feature within a classifier that describes a range of values that instances of the classifier can 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 class*
17.2  B

behavior
The observable effects of an operation or event, including its results.

behavioral diagram
A diagram that depicts 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.
Contrast: 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 Format document that enables you to link inner sections of a document into a master document (using the 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 stereotyped Class that models some system boundary – typically a user interface screen. It is used in the conceptual phase to capture user interaction 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.
17.3 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
A pseudo-state 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 can use a set of interfaces to specify collections of operations it provides to its environment.
See also: interface, object

class diagram
A diagram that shows a collection of declarative (static) model elements, such as Classes, types, and their contents and relationships.
See also: object diagram

classification
The assignment of an object to a classifier.
See also: 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. Now called a Communication diagram.

collaboration occurrence
Uses 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 comment, or note, has no semantics.
*Contrast:* constraint

**communication diagram**
A diagram that 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.
*See also:* collaboration diagram, object diagram

**compile time**
Refers to something that occurs during the compilation of a software module.
*See also:* modeling time, run time, analysis time, design 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 can be implemented by one or more artifacts (e.g. binary, executable, or script files).
*See also:* module
*Contrast:* artifact, product

**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, concurrent substate, disjoint substate

**composite structure diagram**
A diagram that 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 that 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 can be recursive.
*Synonym:* composite aggregation
*See also:* 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. Can be structural, dynamic or possessive.

**constraint**
1. A semantic condition or restriction. Certain constraints are predefined in the UML, others can be user defined. Constraints are one of three extensibility mechanisms in UML.
See also: tagged value, stereotype

Contrast: comment

2. 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 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
A connector linking two nodes in an activity diagram. Control Flow connectors start a node's activity when the preceding node's action is finished.
17.4 D

database schema
The description of a database structure. It defines tables and fields and the relationship between them.

dat astore
An element 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
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 connector that 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.

dele gation
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) affects the other modeling element (the dependent element).

deployment
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 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 is to be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Contrast: analysis

design time
Refers to something that occurs during a design phase of the software development process.
See also: modeling time, run time, compile 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 13 diagram types, and Enterprise Architect extends these with seven more. Add-Ins, technologies and profiles can provide further diagram types.

diagram gate
A simple graphical way to indicate the point at which messages can be transmitted into and out of Interaction Fragments.

diagram view
The Enterprise Architect 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, substate
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 can change its classifier.
See also: multiple classification
Contrast: static classification
17.5 E

element
1. An atomic constituent of a model.
   2. A model object of any type, such as Class, Component, Node or Object.

drop zone
Used in Interaction diagrams to reflect a lost message in sequence.

entity
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
Used to define where external states can enter a Sub Machine.

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
An element that 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
Used in Sub Machine states and State Machines to denote the point where the machine is exited and the transition sourcing this exit point, for Sub Machines, is 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: import

expose interface
A toolbox icon that 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 \((7 + 5 \times 3)\) 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 also: extend, include

extend
A connector used to indicate that 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.
   See also: expression, include
17.6  F

façade
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, that is encapsulated within a classifier, such as an Interface, a Class, or a Datatype.

final
A pseudo-state that 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
An element that 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 class

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
Used in State Machine diagrams as pseudo-states. With respect to State Machine diagrams, a Fork pseudo-state signifies that its incoming transition comes from a single State, and it has multiple outgoing transitions.
Contrast: join

framework
A stereotyped package containing model elements that 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
17.7  G

**generalizable element**
A model element that can 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 can be used where the more general element is allowed.

*See also:* generalizable element, inheritance

**guard condition**
A condition that must be satisfied in order to enable an associated transition to fire.
17.8  H

**history state**

There are two types of History pseudo-states defined in UML: shallow History 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.
17.9 implementation

 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 can 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 can be referenced within a given package (including packages recursively embedded within it).

 See also: visibility

 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 that 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 (i.e. attributes or operations).

 See also: extend, expression

 inheritance
 The mechanism by which more specific elements incorporate the structure and behavior of more general elements related by behavior.

 See also: generalization

 Contrast: delegation

 initial state
 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.

 instance
 An entity that has a unique identity, a set of operations that can be applied to it, and a 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 Timing diagrams, Sequence diagrams, Interaction Overview diagrams and Communication diagrams.

 interaction occurrence
 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
 A diagram that visualizes 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.
See also: class
Contrast: type

**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**
An Enterprise Architect-defined toolbox icon used to define the exception handler and interruptible activity region concepts.
17.10 J

Java
A fully object-oriented, cross platform language based on elements from Smalltalk, C++ and other OO languages.

join
Used in State Machine diagrams and in Activity diagrams to synchronize multiple flows. Contrast: fork

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.
17.11  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**
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, classifier

**local path**
A relative path on a local machine, enabling developers to store shared source code in machine specific directories, but still generate and synchronize code.
17.12 M

**maintenance**
The support of a software system after it is deployed.

**manifest**
A relationship that 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**
An element that defines an endpoint of a Lifeline, such as a State or Value Lifeline in a Timing diagram.

**message label**
Used for messages sent between Lifelines to make the diagram appear less cluttered. Labels with the same name indicate that a message can 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.

**meta-object**
A generic term for all meta-entities in a meta-modeling language. For example, meta-types, meta-classes, meta-attributes, and meta-associations.

**Meta-Object Facility (MOF)**
An Object Management Group (OMG) standard. MOF originated in the UML, when the OMG required a Meta-Modeling architecture to define the UML. MOF is designed as a four-layered architecture.

**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, the meta-metamodel is for brevity frequently referred to simply as 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 enables 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 meta-objects.

**model library**
A stereotyped package containing model elements that are intended to be reused by other packages.
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 the 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, compile 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

**MOF**

Meta-Object Facility, an Object Management Group (OMG) standard. MOF originated in the UML, when the OMG required a Meta-Modeling architecture to define the UML. MOF is designed as a four-layered architecture.

**multiple classification**

A semantic variation of Generalization in which an object can belong directly to more than one classifier.

*See also:* static classification, dynamic classification

**multiple inheritance**

A semantic variation of Generalization in which a type can have more than one supertype.

*Contrast:* single inheritance

**multiplicity**

A specification of the range of enableable cardinalities that a set can assume. Multiplicity specifications can 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, for example, an attribute or parameter, at any point in time.

*Contrast:* single-valued
17.13  N

**name**
A string used to identify a model element.

**namespace**
A part of the model in which the names can 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**
A connector used as an alternative membership notation to indicate nested members within an element; for example, a package that has nested members. The nested members of a package could also be shown inside the package rather than linked by the Nesting connector.

**node**
A classifier that represents a run-time computational resource, which generally has at least a memory and often processing capability. Run-time objects and components can reside on nodes.
17.14  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 can be considered as a special case of a Class diagram or Communication diagram.

*See also:* class diagram, communication diagram

**object flow**
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 (OMG)**
The standards body responsible for the UML specification and management. Their website is [www.omg.org](http://www.omg.org) - follow the links to the UML pages.

**occurrence**
A relationship that 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 could restrict the actual parameters that are possible.
17.15 P

**package**
1. A namespace, as well as an element that can be contained in other packages' namespaces. Packages can own or merge with other packages, and their elements can be imported into a package's namespace.
2. A logical container of model elements. It groups elements and can also contain other packages. The OMG UML specification (UML Superstructure Specification, v2.1.1, p. 109) states:
   
   A package is used to group elements, and provides a namespace for the grouped elements.

   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.

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 exist.

A package is represented by the common folder icon - a large rectangle with a small rectangle (a 'tab') attached to the left side on top.

**package diagram**
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 are imported. Private members of a target package cannot be imported.

**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 can include a name, type, and direction. Parameters are used for operations, messages and events.

*Synonym:* formal parameter
*Contrast:* argument, actual parameter

**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**
A run-time instance of a Class or Interface.

**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 graph 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.
See also: framework

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.
See also: model (MOF)
Contrast: system

port
Defines the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted using the ‘Expose Interface’ toolbox icon.

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.

product
A physical piece of information that is produced by a business or development process. Examples of products include models, source files, scripts, and binary executable files. An product can constitute the implementation of a deployable component.
Synonym: artifact
Contrast: component

profile
A stereotyped package that contains model elements that have been customized for a specific domain or purpose using extension mechanisms, such as stereotypes, tagged definitions and constraints. A profile can also specify model libraries on which it depends and the metamodel subset that it extends.

Project Browser
The workspace window where the model contents are displayed in ‘tree’ format. Displays structures such as packages, diagrams and model elements.

projection
A mapping from a set to its subset.

property
A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are predefined in the UML; others can 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 that becomes available for instantiating repositories and for support
in defining other models. A frozen model's model elements cannot be changed.
17.16 Q

**Qualifier**

An Association attribute or tuple of attributes whose values partition the set of objects related to an object across an Association.
17.17  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, and so on.

receive [a message]
The handling of a stimulus passed from a sender instance.
See also: sender, receiver

receive
An element used to define the acceptance or receipt of a request. Movement on to the next action occurs 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 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.x supports both Expansion Regions and Interruptible Activity Regions. An Expansion Region defines the bounds of a 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 Activity Region contains Activity nodes - when a token leaves an interruptible region, this terminates all of the region's 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
A connector that indicates a Collaboration is used in a classifier. The connector is drawn from the Collaboration to its owning classifier.

requirement
A required feature, property or behavior of a system (external requirement).

responsibility
A contract or obligation of a classifier (internal requirement).

reuse
The use of a pre-existing artifact.

reverse engineering
The process of importing source code into the model as standard UML model objects (such as Classes, attributes and operations).

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/iconix/iconixsw.htm.

**role**

1. The named detail and rules associated with one end of an association. Can indicate name, constraints, multiplicity and collection details.
2. The named specific behavior of an entity participating in a particular context. A role can be static (e.g. an Association end) or dynamic (e.g. a Collaboration role).

**role binding**

The mapping between a Collaboration Occurrence's internal roles and the respective parts required 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.

See also: analysis time, compile time, design time

*Contrast:* modeling time
### Scenario

1. A specific sequence of actions that illustrates behaviors. A scenario can be used to illustrate an interaction or the execution of a Use Case instance.

   **See also:** interaction.

2. A sequence of operations carried out in some order to produce a known result. Can 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, analogous to a package that is a container of model elements. Schema corresponds to a MOF package.

**Contrast:** metamodel, package

### 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 Communication (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 Communication diagrams express similar information, but show it in different ways.

**See also:** communication diagram, object lifeline

### Signal

The specification of an asynchronous stimulus communicated between instances. Signals can have parameters.

### Signature

The name and parameters of a behavioral feature. A signature can include an optional returned parameter.

### Single inheritance

A semantic variation of Generalization in which a type can have only one supertype.

**Contrast:** multiple inheritance

### Single valued [MOF]

A model element with multiplicity defined is single valued when its Multiplicity Type: upper attribute is set to 1. The term single-valued does not pertain to the number of values held by, for example, an attribute or parameter at any point in time, since a single-valued attribute (for instance, with a multiplicity lower bound of zero) could 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 condition applied to a Lifeline that 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 diagram that 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**
A diagram that shows a State Machine.
See also: state machine
Contrast: activity graph

**state continuation**
A symbol that serves two different purposes for Interaction diagrams - as State Invariants and as Continuations. A State Invariant is a condition applied to a Lifeline that must be fulfilled for the Lifeline to exist. A Continuation is used in seq and alt combined fragments to indicate the branches of continuation that 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 can not change classifier.
See also: multiple classification
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 can extend the semantics, but not the structure of pre-existing types and Classes. Certain stereotypes are predefined in the UML, others can 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 can include character sets that support international characters and graphics.

**structural diagram**
A diagram that depicts 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, child, parent

*Contrast:* superclass

**submachine state**
A State in a State Machine that is equivalent to a composite State but its contents are 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:* composite state, concurrent substate, disjoint substate

**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, child, parent

*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 an Activity diagram for organizing the responsibilities for actions. Swimlanes typically correspond to organizational units in a business model.

*See also:* partition

**synch**
A State used for indicating that concurrent paths of a State Machine are synchronized. After bringing the paths to a synch state, the emerging transition indicates 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**
An element used to delineate a particular part of the system.
17.19 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 to as the tag. Certain tags are predefined in the UML; others can be user defined. Tagged Values are one of three extensibility mechanisms in UML.
See also: constraint, property, stereotype

template
The descriptor for a Class with one or more unbound parameters.
Synonym: parameterized element, parameterized class

terminate
A pseudostate indicating that upon entry of its pseudostate, the State Machine's execution ends.

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 a lightweight process.
See also: active object, process, concurrency

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
A diagram that 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 center of Enterprise Architect, from which you can select model elements to insert into diagrams. This is also known as the Enterprise Architect 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, opTopLevels subsystem 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.

temporary 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 performs certain specified actions and enters 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.
See also: fire, object flow

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 can not contain any methods, maintain its own thread of control, or be nested. However, it can have attributes and associations. Although an object can have at most one implementation Class, it can 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.
17.20 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
Diagrams 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 2.1 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 center of Enterprise Architect from which you can select model elements to insert into diagrams. This is also known as the Enterprise Architect UML 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.

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 connector that 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 UML model element that describes how a user of the proposed system interacts 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.

use case diagram
A diagram that captures Use Cases and Actor interactions. It describes the functional requirements of the system, the manner in which 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.

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.
17.21  V

value
An element of a type domain.

value lifeline
A Lifeline that 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
An element that is a textual and/or graphical projection of a collection of model elements.
Contrast: model element (MOF)

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 can be seen outside its enclosing namespace.
See also: export, import

Visual Basic
A rapid application development programming language. Windows’ only scripting language based on COM.
Index

- . -

.EMX
  Import 640

.NET
  Debug Another Process 969
  Debug Assembly 954
  Debug CLR Versions 955
  Debug With COM Interop Process 968
  Debug, System Requirements 964
  Set Up Debug Session 954

.UML2
  Import 640

- A -

Abstract
  Complex Models 1346
  XSD Models 1017

Acceptance Testing 829

Access
  Email 194
  Internet Search Engine 194
  MDG Technologies Remote From Enterprise Architect 518
  Web Site 194

Acknowledgement
  CXImage Library 22
  Of Contributions 22
  Of Trademarks 21
  Print Listview 22

Action
  Element 1280
  Expansion Node 1283
  Local Pre/Post Conditions 1285
  Notation 1281
  Operations 1281
  Pin 1284
  Update Operation 1281

ActionScript
  Import, Reverse Engineering 872
  Modeling Conventions 924
  Options 901
  Versions Supported 901

Activate
  MDG Technologies 517

Activation
  End 1249
  Extend Down 1249
  Extend Up 1249
  Lower 1249
  Raise 1249
  Sequence Element 1249
  Suppress 1249

Activation Layer
  Sequence Diagram Lifelines 1250

Activation Levels
  Sequence Diagram Lifeline Self Messages 1250

Active
  Classes 1338
  State Configuration 1322

Activity
  Element 1286
  Elements And Connectors 142
  Group, Enterprise Architect UML Toolbox 142
  Notation 1287
  Parameter Nodes 1288
  Partition 1289, 1318
  Paste As Action 312
  Paste As Link 312
  Paste From Project Browser 312
  Process Element 1366
  Region Element 1320
  Structured 1326

Activity Diagram
  Create Object From Attribute 384
  Description 1213
  Elements And Connectors 1213
  Example 1213
  Object Flows 1412
  Operations 1281

Activity Edge
  Connector 1393
  Relationship 1393

Activity Final
  Element 1305

Activity Partition
  Element 1318
  Horizontal 1318
  Vertical 1318

Actor
  Element 1290

Adaptive Server Anywhere
  Data Repository, Connect To 593
  ODBC Driver, Set Up 569
  Repository, Create 581
  Upsize To 554

Add
Add
Category To Discussion Forum 252
Code Modules In MDG Technology Wizard 1460
Connectors Between Locked Elements 726
Connectors To UML Model, Quick Start 41
Diagram Properties Note 309
Diagram To Package 39
Diagram To Project 299
Diagram To UML Model, Quick Start 39
Element Changes 839
Element Defects 839
Element Directly To Package 353
Element Issues 839
Element Tasks 839
Element To Diagram 40
Element To Diagram From Project Browser 310
Element To UML Model, Quick Start 40
Elements To Diagram From Project Browser 313
Enumeration Tags To Stereotypes 1438
Expansion Region 1305
Filters To Search 192
Images In MDG Technology Wizard 1462
Instance Variable 1349
Interruptible Activity Region 1314
Key 797
License Key 797
Line Points 446
MDA Transforms In MDG Technology Wizard 1462
Model To Project 545
MS Word Table Of Contents 1186
MS Word Table Of Figures 1187
New Code Sections To Existing Features 921
Note To Connector 443
Note To Link 443
Package In Project Browser 289
Package To UML Model, Quick Start 38
Packages To Model Document 1200
Pattern In MDG Technology Wizard 1458
Pattern To Diagram 512
Port To Element 1352
Post To Discussion Forum 254
Profile Connector To Diagram 492
Profile In MDG Technology Wizard 1457
Project Items Via Toolbar 159
Project Task 847
Property Value To Part 1351
Shape Script To Stereotype In Profile 1439
Tagged Value Types In MDG Technology Wizard 1459
Tagged Values 421
Test Details 825
Topic To Discussion Forum 253
UML Diagram 299
UML Pattern To Diagram 512
UML Profile Connector To Diagram 492
Views 617
Add And Delete Attributes
Automation Interface Code Example 1670
Add And Delete Methods
Automation Interface Code Example 1670
Add And Manage Diagrams
Automation Interface Code Example 1669
Add And Manage Elements
Automation Interface Code Example 1668
Add And Manage Packages
Automation Interface Code Example 1667
Add Connector
Automation Interface Code Example 1669
Add Stereotypes
Automation Interface Code Example 1674
Add Submenu
Element Context Menu, Project Browser 83
Package Context Menu, Project Browser 80
Add to Project Clipboard
Menu Option (Edit Menu) 93
Add-In
And Enterprise ArchitectDeadlocks (.NET) 1534
COM Interoperability 1534
Concurrent Method Calls 1534
Connect To 122
Create 1532
Create, Define Menu Items 1532
Deploy 1533
Disable 1537
Display Help On 122
Enable 1537
Events 1540
Holding State Information 1534
Manage 1537
Manager 1537
MDG 12
Menu 122
Model Driven Generation 12
Pre-2004 1534
Re-entrancy 1534
Register 803
Run Functions From Tasks Pane 1473
Search 189, 1538
Search Data 1538
Submenu 122
Add-In
Tasks 1532
Visual Basic Issues 1534
Add-In Event
EA_Connect 1540
EA_Disconnect 1540
EA_GetMenuItems 1541
EA_GetMenuState 1541
EA_MenuClick 1542
EA_OnOutputItemClicked 1543
EA_OnOutputItemDoubleClicked 1543
EA_ShowHelp 1544
Add-In Model
Add-In Event, EA_Connect 1540
Add-In Event, EA_Disconnect 1540
Add-In Event, EA_GetMenuItems 1541
Add-In Event, EA_GetMenuState 1541
Add-In Event, EA_MenuClick 1542
Add-In Event, EA_OnOutputItemClicked 1543
Add-In Event, EA_OnOutputItemDoubleClicked 1543
Add-In Event, EA_ShowHelp 1544
Add-In Event, Overview 1540
Add-In Tasks 1532
Benefits 1531
Broadcast Event, EA_FileClose 1545
Broadcast Event, EA_FileOpen 1545
Broadcast Event, EA_OnInitialize_Technologies 1570
Broadcast Event, EA_OnPostTransform 1559
Broadcast Event, EA_OnRetrieveModelTemplate 1570
Broadcast Events 1545
Compartment Events 1560
Compartment Events, EA_GetCompartmentData 1560
Compartment Events, EA_QueryAvailableCompartments 1560
Context Item Events 1557
Context Item Events, EA_OnContextListItemChanged 1557, 1558
Context Item Events, EA_OnContextItemDoubleClicked 1558
Create Add-In 1532
Create Add-In, Tricks and Traps 1534
Create Custom View 1572
Custom View 1572
EA_Connect 1540
EA_Disconnect 1540
EA_FileClose 1545
EA_FileOpen 1545
EA_GetCompartmentData 1560
EA_GetMenuItems 1541
EA_GetMenuState 1541
EA_MenuClick 1542
EA_OnContextListItemChanged 1557, 1558
EA_OnContextItemClick 1557
EA_OnContextItemDoubleClicked 1558
EA_OnDeleteTechnology 1556
EA_OnElementValidation 1563
EA_OnImportTechnology 1556
EA_OnInitialize_Technologies 1570
EA_OnInitializeUserRules 1562
EA_OnOutputItemClicked 1543
EA_OnOutputItemDoubleClicked 1543
EA_OnPreDeletePackage 1547
EA_OnPreDeleteTechnology 1555
EA_OnPreNewAttribute 1550
EA_OnPreNewConnector 1549
EA_OnRetrieveModelTemplate 1570
EA_OnRunAttributeRule 1565
EA_OnRunConnectorRule 1564
EA_OnRunDiagramRule 1564
EA_OnRunElementRule 1563
EA_OnRunMethodRule 1565
EA_OnRunPackageRule 1564
EA_OnStartValidation 1562
EA_QueryAvailableCompartments 1560
EA_ShowHelp 1544
Interface 1531
Introduction 1531
MDG Add-Ins 1573
MDG Add-Ins, MDG Events 1573
MDG Events, MDG_BuildProject 1573
MDG Events, MDG_Connect 1574
MDG Events, MDG_Disconnect 1574
MDG Events, MDG_GetConnectedPackages 1575
MDG Events, MDG_GetProperty 1575
MDG Events, MDG_Merge 1576
MDG Events, MDG_NewClass 1577
Add-In Model
  MDG Events, MDG_PostGenerate  1578
  MDG Events, MDG_PostMerge  1579
  MDG Events, MDG_PreGenerate  1579
  MDG Events, MDG_PreMerge  1580
  MDG Events, MDG_PreReverse  1580
  MDG Events, MDG_RunExe  1581
  MDG Events, MDG_View  1581
  Model Validation Broadcasts  1562
  Model Validation Broadcasts,
    EA_OnEndValidation  1563
  Model Validation Broadcasts,
    EA_OnInitializeUserRules  1562
  Model Validation Broadcasts,
    EA_OnRunAttributeRule  1565
  Model Validation Broadcasts,
    EA_OnRunConnectorRule  1564
  Model Validation Broadcasts,
    EA_OnRunDiagramRule  1564
  Model Validation Broadcasts,
    EA_OnRunElementRule  1563
  Model Validation Broadcasts,
    EA_OnRunMethodRule  1565
  Model Validation Broadcasts,
    EA_OnRunPackageRule  1564
  Model Validation Broadcasts,
    EA_OnRunParameterRule  1566
  Model Validation Broadcasts,
    EA_OnStartValidation  1562
  Model Validation Example  1566
Post-New Events  1552
  Post-New Events, EA_OnPostNewAttribute  1554
  Post-New Events, EA_OnPostNewConnector  1553
  Post-New Events, EA_OnPostNewDiagram
    (Object)  1553
  Post-New Events, EA_OnPostNewElement  1552
  Post-New Events, EA_OnPostNewMethod  1554
  Post-New Events, EA_OnPostNewPackage  1555
Pre-Deletion Events  1546
  Pre-Deletion Events, EA_OnPreDeleteConnector  1546
  Pre-Deletion Events, EA_OnPreDeleteDiagram  1547
  Pre-Deletion Events, EA_OnPreDeleteElement  1546
  Pre-Deletion Events, EA_OnPreDeletePackage  1547
Pre-New Events  1548
  Pre-New Events, EA_OnPreNewAttribute  1550
  Pre-New Events, EA_OnPreNewConnector  1549
  Pre-New Events, EA_OnPreNewPackage

Pre-New Events, EA_OnPreNewDiagram
  (Object)  1549
Pre-New Events, EA_OnPreNewElement  1548
Pre-New Events, EA_OnPreNewMethod  1551
Pre-New Events, EA_OnPreNewPackage  1551
Search Data, XML Format  1538
Technology Events  1555
Technology Events, EA_OnDeleteTechnology
  1556
Technology Events, EA_OnImportTechnology  1556
Technology Events, EA_OnPreDeleteTechnology  1555

Administrator
  Security Permissions  709
Advanced
  Settings, Generalizable Elements  411
  Tag Management  420
Advanced (Element)
  Submenu  106
Aggregate
  Connector  1375
  Relationship  1375
Aggregation Connector
  Change Form  1375
Align
  Elements From Toolbar  163
  Multiple Elements  360
Alignment
  Submenu  107
All Permissions
  Dialog, User Security  715
  View, User Security  715
Alternative Image
  For Element  320
  In Diagram  320
  Select  320
  Stereotype  506
Analysis
  Elements and Connectors  147
  Group, Enterprise Architect UML Toolbox  147
  Stereotypes  1357
Analysis Diagram
  Description  1269
  Elements And Connectors  1269
  Example  1269
ANSI C  902, 925
Anti Aliased Text
  In Diagrams  234
Apache Tomcat
  Server Configuration  974
  Server, Debugging  970
Apache Tomcat
  Service Configuration  975
App Object
  Automation Interface  1592
Appearance
  Apply From Clipboard  348
  Connectors Context Menu Section  441
  Copy For Element  348
  Default, Of Element  365
  Element Context Menu  348
Appearance (Element)
  Autosize  107
  Submenu  107
Applets
  Java, In Internet Browsers, Debug  976
Application Workspace  53
Apply
  RTF Report Filter (Legacy)  1175
  Stereotype To Dependency Relationship  1385
  Stereotype To Element  501
  Stereotype To UML Construct  501
  User Lock  728
Argument
  For Operation Parameter  389
Arrange
  Connectors  444
Arrow
  Quick Linker  225
Artifact
  Element  1336
ASA
  Data Repository, Connect To  593
  ODBC Driver, Set Up  569
  Repository, Create  581
  Upsize To  554
ASP .NET
  Debug  965
Assembly
  Connector  1376
  Debug  954
  Relationship  1376
Assign
  Information To Tagged Values  217
  People To Changes  845
  People To Defects  845
  Tagged Values To Item  215
Associate
  Connector  1377
  Relationship  1377
Associated Files
  Elements  418
Association
  Class  1365
  Connector  1377
  Connector, Set Collection Class  899
  Details  457
  N-Ary  1365
  Properties  457
  Relationship  1377
  Specialisation, Set Up  453
Association Class
  Connector  1378
  Link New Class To Association  1379
  Relationship  1378
Attach
  Note To Link  443
Attribute
  Add And Delete, Automation Interface Code
  Example  1670
  Add To Element, In-place Editor  403
  Automation Interface, ElementFeatures Package
  1636
  Collections  378
  Constraints  379
  Copy Between Elements  371
  Create  374
  Create Fast, Option  892
  Create Object From  384
  Definition  374
  Delete  376
  Delete If Not In Code In Reverse Synchronization
  892
  Dialog  374
  Dialog, Constraints Tab  379
  Dialog, Detail Tab  378
  Dialog, General Tab  376
  Edit Keyword  401
  Edit Name, In-Place Editor  399
  Edit Scope  400
  Edit Stereotype, In-place Editor  399
  Fast Create  374
  Imported, Default Name Generated From  892
  Inherited, Display  383
  Inherited, Show  307
  Introduction  374
  Link To Element Via Object  384
  Message Part, WSDL  1035
  Modify  374
  Move Between Elements  372
  Multiplicity  378
  Private, Icon  76
  Properties, Create  381
  Protected, Icon  76
  Show On Diagram  307

© 1998-2008 Sparx Systems Pty Ltd
Attribute
Stereotyped, For Columns 1049
Supported, By XML Element Node 496
Supported, Create Composite Elements 1445
Supported, Define Child Diagram Types 1446
Supported, In UML Profile 496
Supported, Metatype In UML Profiles 1442
Supported, Stereotype In UML Profiles 1442
Tagged Values 379
UML Property, isUnique 378
Work With, Automation Interface Code Example 1675

AttributeConstraint
Automation Interface, ElementFeatures Package 1637

AttributeTag
Automation Interface, ElementFeatures Package 1638

Audit
Scope 734

Audit History Tab
Description 742
How To Display 742
On Output Window 175, 221

Audit Options
All 735
Connectors Audited 735
Core Structural 735
Custom 735
Elements Audited 735
Maintenance 735

Audit View
Audit Changes 738
Controls 738
Custom Time Periods 738
Display Database Changes 738
Filter By Time 738
Mode, Advanced 738
Mode, Raw 738
Mode, Standard 738
Performance Problems 744
Refresh 738
Search 738
Slow Loading 744
Slow Navigation 744
Sort 738

Auditing
Alternative To Differencing 745
And RTF Reporting 732
Audit Tree 737
Display Audit Results 737
Enable 734

Audit To Invoke 733
Include Reverse Engineering 734
Include XMI Export 734
Include XMI Import 734
Introduction 732
Large Deletion Issue 743
Level, Core 735
Level, Extended 735
Level, Standard 735
Performance Issues 743
Quick Start 733
Record Display 737
Reverse Engineering Issue 743
Settings 734
Use Database Timestamp 734
XMI Import Issue 743

Auditing Settings
Clear Logs 735
Load Logs 735
Save Logs 735

Author
Attributes 1607
Define 766
From Windows Active Directory 766
Methods 1607

Author Collection
Automation Interface Repository 1607

Auto Counter
Of Elements 353

Auto Naming
Of Element 353

Autohide
Reveal Autohidden Window 199
Turn Off 199
Turn On 199
Windows 199

Autolayout
Diagram 300
Options 300

Automatically Hidden Windows
Animate 96

Automation Interface
App Object 1592
Attribute, ElementFeatures Package 1636
AttributeConstraint, ElementFeatures Package 1637
AttributeTag, ElementFeatures Package 1638
Available Resources 1588
Call Executables From Enterprise Architect 1583
Call From Enterprise Architect 1587
Automation Interface
  Code Example, Add And Delete Attributes 1670
  Code Example, Add And Delete Methods 1670
  Code Example, Add And Manage Diagrams 1669
  Code Example, Add And Manage Elements 1668
  Code Example, Add And Manage Packages 1667
  Code Example, Add Connector 1669
  Code Example, Add Stereotypes 1674
  Code Example, Iterate Through EAP File 1667
  Code Example, Open The Repository 1667
  Code Example, Use Element Extras 1671
  Code Example, Use Repository Extras 1673
  Code Example, Work With Attributes 1675
  Code Example, Work With Methods 1675
  Code Examples, Introduction 1666
  Connect From Borland Delphi 7.0 1584
  Connect From Java 1584
  Connect From MS C# 1584
  Connect From MS Visual Basic 6.0 1584
  Connect To 1583
  Connector Package Diagram 1645
  Connector, Connector Package 1647
  ConnectorConstraint, Connector Package 1645
  ConnectorEnd, Connector Package 1646
  ConnectorTag, Connector Package 1649
  ConstLayoutStyles Enum 1593
  Constraint, Element Package 1622
  CustomProperties Collection, ElementFeatures Package 1638
  Diagram Package 1650
  Diagram, Diagram Package 1651
  DiagramLinks, Diagram Package 1654
  DiagramObjects, Diagram Package 1654
  Effort, Element Package 1622
  Element Package Diagram 1620
  Element Package, File 1629
  Element, Element Package 1623
  ElementFeatures Package Diagram 1635
  EmbeddedElements Collection, ElementFeatures Package 1639
  Enumerations 1592
  EnumRelationSetType Enum 1593
  Examples 1583
  Examples and Tips 1586
  Introduction 1583
  Issue, Element Package 1629
  MDGMenus Enum 1593
  Method, ElementFeatures Package 1639
  MethodConstraint, ElementFeatures Package 1641
  MethodTag, ElementFeatures Package 1641
  Metric, Element Package 1630
  Model 1589
  ObjectType Enum 1594
  Package 1589
  Parameter, ElementFeatures Package 1642
  Partitions Collection, ElementFeatures Package 1643
  Project Interface 1657
  Project, Project Interface 1657
  Properties, ElementFeatures Package 1643
  Property ElementFeatures Package 1643
  PropType Enum 1594
  Reference 1589
  ReloadType Enum 1595
  Repository 1596
  Repository Package 1595, 1612
  Repository, Author Collection 1607
  Repository, Client Collection 1607
  Repository, Collection Class 1608
  Repository, Datatype 1609
  Repository, EventProperties 1610
  Repository, EventProperty 1611
  Repository, ModelWatcher 1611
  Repository, ProjectIssues 1615
  Repository, ProjectResource 1616
  Repository, PropertyType 1617
  Repository, Reference 1617
  Repository, Stereotype 1618
  Repository, Task 1619
  Repository, Term 1620
  Requirement, Element Package 1631
  Resource, Element Package 1631
  Risk, Element Package 1632
  RoleTag, Connector Package 1650
  Scenario, Element Package 1633
  Set Up Visual Basic 1585
  Swimlane, Diagram Package 1657
  SwimlaneDef, Diagram Package 1655
  Swimlanes, Diagram Package 1656
  TaggedValue, Element Package 1633
  Test, Element Package 1634
  Transitions Collection, ElementFeatures Package 1644
  Using 1584
  VB GetObject Support 1592
  XMIType Enum 1595

Autosize
  Element, Single 310
  Elements, As Group 310

© 1998-2008 Sparx Systems Pty Ltd
Available Resources
  Automation Interface 1588

- B -

Base Project
  Copy 599
  New 599

Baseline
  And Differences, Overview 745
  Considerations 746
  Create 748
  Delete 746
  Export 746
  Import 746
  Load From Alternative Model 746
  Manage 746
  Merge With Current Model, Overview 745
  Model 746
  Overview 746
  Scenarios 746
  Versions 748

BaseModel Script
  InnoDB 562
  MyISAM 562

Batch Generate
  Elements With Code 106
  RTF Resource Documents, Automatically 1166
  RTF Resource Documents, Manually 1166

Batch XML
  Export 650
  Import 650

Behavioral Diagram
  Elements 1279
  Overview 1213

Bend
  Connector At Cursor 446
  Bend Connector 41
  Bezier Lines 446

Binary Module
  Import, Reverse Engineering 875

Binding
  WSDL Diagram 1032
  WSDL Element 1032

Bitmap Image
  In Diagrams 320

Blue Exclamation Mark 729

Bookmark
  Clear 93
  Clear All 93
  For RTF Report 1184

Insert In RTF Template 1157
Multiple Elements 865
Package As 1184
RTF, In Master Document Element 1196
Selected Element 93
Triangle 865

Bookmark Selected
  Menu Option (Edit Menu) 93

Boundary
  Element 1358
  Element Settings 424
  Element, Create 1358
  Object Settings 424
  Properties 424

BPMN
  1.3 529
  1.4 529
  Change Element Appearance 531
  Concepts 529
  Connectors 529
  Core Toolbox Page 529
  Diagram 529
  Disable 529
  Element Appearance, Change 531
  Elements 529
  MDG Technology 529
  Relationships 529
  Types Toolbox Page 529

Branching Macros
  Code Template Syntax 1521

Breakpoint
  Add To Code 982
  Delete All 982
  Delete Single 982
  Disable 982
  Enable 982
  Overview 982
  States 982

Broadcast Event
  Add-In Model 1545
  EA_FileClose 1545
  EA_FileOpen 1545
  EA_OnInitialize_Technologies 1570
  EA_OnPostTransform 1559
  EA_OnRetrieveModelTemplate 1570

Browser
  Element 210

Build and Run 941
  Build Script, Create 947
  Debug With Enterprise Architect 963
  Deploy Script, Create New 958
  Manage Compile Scripts 945
Index 1720

Build and Run 941
  Record A Debug Session 992
  Run Script, Create New 950
  Sequence Diagram 992
  Source Code Configuration 943
  Submenu (Project Menu) 99
  Unit Test Script, Create New 948
  Unit Testing 1003
Build And Run Submenu
  Package Context Menu, Project Browser 81
Build Script
  Create 947
  Dialog 946
  Recursive 956
  Sequence Tab 959
Build Systems Using UML
  Enterprise Architect 6
Built-In
  Diagram Types 1471
  Transformations 1099
Business Analyst
  And Enterprise Architect 271
  Project Role 271
Business Interaction Diagram
  Description 1276
  Elements And Connectors 1276
  Example 1276
Business Modeling
  Business Process Outline 540
  Events 537
  Example 533
  Goals 539
  Information 536
  Inputs 536
  Outputs 538
  Process Element 535
  Process Modeling Notation 535
  Processes 533
  Resources 536
  Traceability 755
Business Modeling Diagram
  Description 1276
  Elements And Connectors 1276
  Example 1276
Business Process
  Analysis 59
  Model, Template 59
  Outline 540
Business Process Modeling 1269
Business Process Modeling Notation (BPMN) 529

- C -

C
  Code Generation 902
  Import, Reverse Engineering 872
  Modeling Conventions 925, 926
  Object Oriented Programming 926
C#
  Import, Reverse Engineering 872
  Modeling Conventions 927
  Options 902
  Transformation 1100
C++
  Code Generation 903
  Implementation Files 903
  Import, Reverse Engineering 872
  Modeling Conventions 929
  Modeling Conventions, CLI Extensions 930
  Modeling Conventions, Managed 930
  Options 903
Calibration
  Of Project Factors 806
Call
  Automation Interface From Enterprise Architect 1587
  Self Message 1399
Camel Case
  Naming Format 1129
Cancel
  Default Diagram, Model 332
  Validation 101
Capture State Transitions
  Debugger Sequence Tab Option 998
Cardinality
  Define 787
CASE Tool
  Enterprise Architect 3
Category
  Add To Discussion Forum 252
  Author 252
Central Buffer Node
  Element 1291
Chaining Transformations 1095
Change
  BPMN Element Appearance 531
  Connector Source Or Target 445
  Connector Type 445
  Diagram Type 305
  Element 841
  Element Type 359
  Form Of Aggregation Connector 1375

© 1998-2008 Sparx Systems Pty Ltd
Change
   Tracking  732
Change Conflicts
   Resolve  634
Change Element
   Hide Stereotype Letter  843
   Show Stereotype Letter  843
Character Set
   Set Up For RTF Report  1171
   Set Up For RTF Report (Legacy)  1181
Check
   Data Integrity  604
   Model Integrity  604
   Project Integrity  604
Check Constraint
   Create  1070
   What Is A?  1070
Check In
   Branch  699
   Explanation  695
   Offline Packages  706
   Packages Online  699
   Project Browser Icon  699
Check Out
   Explanation  695
   Packages Offline  699, 706
   Project Browser Icon  699
Checked In Package
   Icon  76
Checked Out Package
   Icon  76
Child
   Confirm Element As Parent  357
Child Element
   Paste Object As  310
Choice
   Element  1291
Class
   Active Classes  1338
   Collection, Set  899
   Create As Container Class  1062
   Create As Stored Procedure  1065
   Created In Transformation, Connect To  1125
   Element  1337
   Elements And Connectors  135
   Elements, Imported  1088
   Group, Enterprise Architect UML Toolbox  135
   Make Into Association Class  1379
   Members, Show/Hide On Diagram  332
   Model Template  63
   Parameterized Classes (Templates)  1338
   Partial  927
   Partial, Generate  884
   Reset Options  913
   Show Realised Interfaces On Diagram  323
   Source Code Generation  879
   View  617
Class Diagram
   Description  1260
   Edit Elements  105
   Elements And Connectors  1260
   Example  1260
CLASSGUID
   Add-In Hidden Field  1538
Classifier
   Drop As Link  313
   Drop As New Instance  313
   Item Conveyed  1391
   Properties  1264
   Set  423
   Use  423
Classifiers
   Of Objects  422
CLASSTYPE
   Add-In Hidden Field  1538
Clean
   Project  604
Clear All Bookmarks
   Menu Option (Edit Menu)  93
Clear Project Clipboard
   Menu Option (Edit Menu)  93
Clear Selection
   Menu Option (Edit Menu)  93
CLI Extensions
   C++ Modeling Conventions  930
Client
   Define  772
Client Collection
   Automation Interface Repository  1607
Clipboard
   Copy Discussion Forum Path To  261
Clipboard File Format
   Define  231
Close
   Full Screen  124
Close Project
   Menu Option (File Menu)  87
CLR Versions
   Debug .NET  955
Code
   Delete From Features In Model In Fwd Synchronization  892
   Generation, Toolbar  160
   Import, Select Language  160

Enterprise Architect User Guide
Color
  Of Communication Messages 242
Color Coded External Requirements 468
Color Query
  Shape Scripts 1488
Column
  Create In Data Modeling 1049
  Definition 1049
  In UML Data Modeling Profile 1049
  Order, Change 1049
  Stereotyped Attribute 1049
COM Interop
  Debug .NET 968
Combine
  Windows In One Frame 196
Combined Fragment
  Create 1294
  Element 1292
  Interaction Operator 1295
Comma Separated Value
  Export 653, 657
  Import 653, 659
Commands
  Add To Toolbar 111
  Change Icon Appearance 111
  Customize 111
  Remove From Toolbar 111
Common
  Connectors 133
  Elements 133
  Group, Enterprise Architect UML Toolbox 133
  Relationships 133
Communication
  Connector 1382
  Elements and Connectors 138
  Group, Enterprise Architect UML Toolbox 138
  Message 1403
  Message, Create 1403
  Message, Properties 1403, 1404
  Message, Sequence 1404
  Relationship 1382
Communication Diagram
  Description 1253
  Elements And Connectors 1253
  Example 1253
  Labelled Associations 1253
  Message Colors 1254
  Numbering In 1253
Communication Message
  Colors 242
Communication Path
  Connector 1380
  Relationship 1380
Compact
  Project .EAP File 615
Compare
  Data 610
  DDL With Database 1074
  Models 610
  Projects, Instructions 610
  Projects, Why? 609
  Utility 745
Compare Utility 749
  Context Menu 752
  Keyboard Options 752
  Merge Options 752
  Options 750
  Output 751
  Tab 751
  Toolbar 752
Compartment
  Constraint 426
  Element 426
  Maintenance 426
  Responsibility 426
  Tag 426
  Testing 426
Compartment Events
  Add-In Model 1560
  EA_GetCompartmentData 1560
  EA_QueryAvailableCompartments 1560
Compile Scripts
  Manage In Build and Run 945
Compiled 28 October 2008 3
Complex Modeling
  Enterprise Architect 6
Component
  Element 1342
  Elements And Connectors 143
  Group, Enterprise Architect UML Toolbox 143
  Model Template 65
  View 617
Component Diagram
  Description 1265
  Elements And Connectors 1265
  Example 1265
Compose
  Connector 1381
  Relationship 1381
Composite
  Elements 1359
  Elements And Connectors 137
  Foreign Key 1056
  Group, Enterprise Architect UML Toolbox 137
Composite
  State 1321, 1322
  State Regions 1219
Composite Aggregation
  Connector 1381
  Relationship 1381
Composite Element
  Paste From Project Browser 312
Composite Elements
  MetaClass, Create With Supported Attributes 1445
Composite Structure Diagram
  Description 1263
  Elements And Connectors 1263
  Example 1263
Composition
  Hierarchy 474
Compress
  Timeline 1242
  Transition 1242
Concept
  BPMN 529
  Mind Mapping 522
Concurrent Method Calls
  In Add-Ins 1534
Concurrent Substate
  Regions 1219
Conditional Substitution
  Field Substitution Macros, Code Template Syntax 1509
Configuration
  Apache Tomcat Server 974
  JBOSS Server 973
  Tomcat Server 974
  Tomcat Service 975
Configure
  Controlled Packages With XMI 647
  Local Options 230
  Model Validation 622
  Options 230
  Package For Version Control 697
  Packages 647
  User Settings 230
  Version Control, Subversion 689
Configure Timeline Dialog
  States Tab 1236
  Transitions Tab 1237
Confirm
  Parent Element 357
Connect
  Elements 442
  Objects 442
To ASA Data Repository 593
To Automation Interface 1584
To Data Repository 584
To MSDE Server Data Repository 595
To MySQL Data Repository 584
To Oracle 10g Data Repository 589
To Oracle 11g Data Repository 589
To Oracle 9i Data Repository 589
To PostgreSQL Data Repository 591
To Progress OpenEdge Data Repository 595
To SQL Server Data Repository 587
Connections
  In Discussion Forum 263
  In Relationships Window 211
  To Other Forums 263
  Window 211
Connector
  Activity Edge 1393
  Add Between Locked Elements 726
  Add Note 443
  Add To Diagram 41
  Add To UML Model, Quick Start 41
  Add, Automation Interface Code Example 1669
  Advanced, Menu Section 440
  Aggregate 1375
  Appearance 441
  Arrange 444
  Assembly 1376
  Associate 1377
  Association 1377
  Association Class 1378
  At Page Boundaries 446
  Automation Interface, Connector Package 1647
  Bend At Cursor 446
  Bend Connector 41
  BPMN 529
  Change Source Or Target 445
  Change Type 445
  Characteristics, Edit 440
  Communication 1382
  Communication Path 1380
  Compose 1381
  Composite Aggregation 1381
  Connector 1382
  Constraints 458
  Context Menu 439
  Control Flow 1383
  Create Between Elements 448
  Create From Project Browser 448
  Create From UML Toolbox 126
Connector
  Create In MDA-Style Transformation  1125
  Create With Element Using Quick Linker  226
  Create With Quick Linker  228
  Create, Same Type As Previous  103
  Custom Properties  344
  Custom Properties, Set Value  440
  Data Flow  524
  Delegate  1384
  Delete  449
  Dependency  1385
  Dependency, Apply Stereotype  1385
  Deployment  1386
  Destination Role  461
  Details  457
  Display Options  241
  Extend  1387
  Generalization  1388
  Generalize  1388
  Hide  449
  Hide/Show  441
  Implements  1417
  Include  1389
  Information Flow  1390
  Inheritance  1388
  In-place Editor Options  453
  Interrupt Flow  1393
  Labels  323
  Labels, Edit  453
  Line Color  232
  Locked  726
  Manifest  1394
  Message  1395
  Move  44, 444
  Multiplicity  459
  Nesting  1410
  Notelink  1411
  Notes  206
  Object Flow  1412
  Occurrence  1414
  Off-Page  438
  Overview  1373
  Package Import  1415
  Package Merge  1416
  Page Boundary  446
  Pkg Import  1415
  Pkg Merge  1416
  Properties  457
  Properties Menu  439
  Realization, Quick Generation Of  473
  Realize  1417
  Recursion  1418

  Relationship  1382
  Representation  1421
  Represents  1420
  Reverse Direction  453
  Role Binding  1419
  Role, Context Menu  439
  Self-Message  1398
  Shape Script Properties  1489
  Source Role Properties  459
  Styles  441, 446
  Tagged Value, Use  1435
  Target Role Properties  461
  Tasks  442
  To Class Created In Transformation  1125
  Trace  1422
  Transform  1125
  Transition  1423
  Type Specific Menu Section  440
  Type, Change  445
  Usage  1425
  Use  1425
  Visibility  441, 451, 454
  What Is A?  1373
  Working With  438

Connector Package
  Connector, Automation Interface  1647
  ConnectorConstraint, Automation Interface  1645
  ConnectorEnd, Automation Interface  1646
  ConnectorTag, Automation Interface  1649
  RoleTag, Automation Interface  1650

Connector Package Diagram
  Automation Interface  1645

Connector Styles
  Auto Routing  446
  Bezier  446
  Custom  446
  Direct  446
  Line Style  446
  Set  446

ConnectorConstraint
  Automation Interface, Connector Package  1645

ConnectorEnd
  Automation Interface, Connector Package  1646

ConnectorTag
  Automation Interface, Connector Package  1649

ConstLayoutStyles Enum
  Automation Interface  1593

Constraint
  Automation Interface, Element Package  1622
Constraint
  Compartment, Element  426
  Element  415
  In Connector  458
  Inherited, Show  307
  Note, Element  1317
  On States, Debugger Generation Of Sequence Diagrams  1001
  Operators On  1001
  Profile  1436
  Status Type, Define  777
  Stereotype  1436
  Type, Define  776

Constraints
  Delete  379
  In Rules & Scenarios Window  212
  Of Attributes, Create  379

Contents
  Table, MS Word, Add  1186

Contents Submenu
  Package Context Menu, Project Browser  81

Context Diagram  524
Context Element
  Highlight  370
  In Multiple Selection  370

Context Item Events
  Add-In Model  1557
  EA_OnContextItemChanged  1557, 1558
  EA_OnContextItemDoubleClicked  1558

Context Menu
  Connector  439
  Connector Label  323
  Connector Role  439
  Diagram  297
  Diagram, Project Browser  84
  Discussion Forum  260
  Element  341
  Element Label  323
  Element, Add Supporting Diagrams and Elements  344
  Element, Multiple Selection  350
  Element, Project Browser  83
  Linked Document Editor  433
  Method, Project Browser  85
  Model Views  178
  Model, Project Browser  77
  New Element Or Connector  297
  Operation, Project Browser  85
  Package, Project Browser  79

Continuation
  Element  1324

Control
  Create  1360
  Element  1360

Control Flow
  Connector  1383
  Guard  1383
  Relationship  1383
  Weight  1383

Control Macros
  Code Template Syntax  1521

Controlled Package  697
  Batch Export To XMI  650
  Batch Import From XMI  650
  Load  649
  Menu, XMI  646
  Recovery  651
  Version Control  651
  With XMI  646

Convert
  Linked Element To Local Copy  371
  Names In MDA Transformations  1129

CONVERT_NAMES  1129
CONVERT_TYPE  1128

Convey
  Information Item  1391

Copy
  Attributes Between Elements  371
  Base Project  599
  Diagram Image To Clipboard  305
  Diagram Image To Disk File  304
  Diagram, Deep  306
  Diagram, Shallow  306
  Element On Diagram  304
  Model  88, 89, 90
  Operations Between Elements  371
  Packages Between Projects  611
  RTF Bookmark To Clipboard  80, 84
  UML Diagram, Deep  306
  UML Diagram, Shallow  306

Copyright Notice  17

CORBA
  MDG Technology For, Enterprise Architect  12

Co-Region Notation  1395

Corporate Edition  9

Correct Spell Checked Words  267

Corrupt EAP file  616

Create
  A UI Control Element  1370
  Adaptive Server Anywhere Repository  581
  Add-In  1532
  Attribute Properties  381
  Baselines  748
  Boundary Element  1358
Create

Build Script 947
Check Constraint 1070
Columns In Data Modeling 1049
Combined Fragment 1294
Communication Messages 1403
Connector With Quick Linker 228
Control Element 1360
Custom Diagram Background, Image Manager 321
Custom Tagged Values 1505
Custom View, Add-In Model 1572
Data Repository 574
Deploy Script 958
Design Master 633
Diagram 299
Diagram From Linked Document 434
Document Artifact 432
Documents 1132
Element And Connector With Quick Linker 226
Element From Linked Document 434
Element In Diagram 352
Element On Diagram From Project Browser 313
Element Template 369
Elements 352
Entity 1361
Favorites Folder, Model View 178
Favorites Folder, Model View (Context Menu) 178
Foreign Key 1056
Hidden Submenu 1466
HTML Report 1205
Index (Data Modeling) 1070
Link Between Elements 448
Linked Document Template 435
MDG Technologies 1454
Model 551
Model Files Discussion 598
MSDE Server Repository 583
MySQL Repository 574
Notes 363
Oracle 10g Server Repository 578
Oracle 11g Server Repository 578
Oracle 9i Server Repository 578
Pattern 508
PostgreSQL Repository 579
Predefined Reference Data Tagged Value Type 1503
Primary Key 1052
Primary Key Name Template 1052
Profiles 1431
Progress OpenEdge Repository 583

Project 551
Project File 545
Relationship Using Matrix 483
Replicas 633
Requirements 465
Rich Text Format Report (Enhanced Generator) 1135
Root Node, Model View 178
Root Node, Model View (Context Menu) 178
RTF Report (Enhanced Generator) 1135
RTF Style Template 1139
RTF Style Template (Legacy) 1179
Run Script 950
Search Definition 189
Sequence Diagram in Debug Session 996
Sorted Lookup Table 1070
SQL Server Repository 576
State Machine For Sequence Diagram Generation, Debugger 1000
Structured Tags 1499
Table In Data Modeling 1042
Tasks Pane Profiles 1472
Text 363
Timing Diagram 1230
Timing Message 1407
Toolbox Profile In MDG Technology 1465
Trigger Operation 1070
UML Diagram 299
UML Pattern 508
UML Profiles 1431
UML Project In Enterprise Architect 35
Unit Test Script 948
View 1067
Views Folder, Model View 178
Views Folder, Model View (Context Menu) 178
Views, Model View 178
Views, Model View (Context Menu) 178

Create Diagram
Automatically 289
Create Property Implementation Dialog 381
Cross Reference
Between EMX Files 642
Delete 356
Set For Element 356
Trace With Hierarchy Window 213
Use In Element 356

CSV
Export 653, 657
Export From Relationship Matrix 484
Export State Machine Table To 1228
Import 653, 659
Preserve Hierarchy 654
CSV Specifications 654
CTF In SDK 1507 Overview 915
Current Connector Toolbar 165
Current Element Toolbar 164
Custom Diagram 1272, 1273, 1274 Diagram Types 1470 Elements and Connectors 148
Group, Enterprise Architect UML Toolbox 148
Layout 245
Stereotypes 1429

Custom Background Create For Diagram, Image Manager 321

Custom Diagram
Description 1270 Elements And Connectors 1270 Example 1270 Model 1270

Custom Language Create Templates For In Code Template Editor 1529
Settings, RTF Report Generation 1171

Custom Properties
Dialog 344 For Connectors 344 For Elements 344

Custom Reference
Delete 356 Set For Element 356 Trace With Hierarchy Window 213 Use In Element 356

Custom Tagged Values Create 1505

Custom Template Create From Project 551

Custom Tools
External Applications 116

Custom View Add-In Model 1572

Customize
Commands 110, 111 Diagram Appearance 325 Dialog 110 External Tools, Pass Parameters To 117 Keyboard 110 Keyboard Shortcuts 118 Menu Appearance 120

Menus 110 Options 110 RTF Language 1171 RTF Language (Legacy) 1181 Toolbar Option Appearance 121 Toolbars 110, 111 Tools 110, 114 Visibility Of Elements 363 Window 110

CustomProperties Collection
Automation Interface, ElementFeatures Package 1638

 CVS
Remote Repository 679 Version Control Options 679, 683 Version Control With Local Repositories 683

- D -

Dashed Border On Element 357

Data
Compare 610 Export 790 Import 791 Integrity 603 Integrity Check 604 Integrity, Run SQL Patches 606 Model Template 64

Data Distribution Service
MDG Technology For, Enterprise Architect 12

Data Flow
Concepts 524 Connector 524 Context Diagram 524 Diagram 524 MDG Technology 524 Relationship 524 Toolbox Page 524

Data Management
Project Compare 109 Project Integrity 109 Project Transfer 109 Submenu (Tools Menu) 109

Data Modeling
Check Constraint 1070 Compare DDL With Database 1074 Create Columns 1049 Create Table 1042 Data Model Diagram 1041 Data Type Conversion Procedures 1078 DBMS Conversion Procedure Package 1079

© 1998-2008 Sparx Systems Pty Ltd
<table>
<thead>
<tr>
<th>Data Modeling</th>
<th>DBMS Data Types</th>
<th>1081</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DDL, Generate</td>
<td>1072</td>
</tr>
<tr>
<td></td>
<td>Elements And Connectors</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>Foreign Keys</td>
<td>1055</td>
</tr>
<tr>
<td></td>
<td>Generate DDL</td>
<td>1072</td>
</tr>
<tr>
<td></td>
<td>Generate DDL For A Package</td>
<td>1074</td>
</tr>
<tr>
<td>Group, Enterprise Architect UML Toolbox</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td>1070</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>1039</td>
<td></td>
</tr>
<tr>
<td>Primary Key Extended Properties</td>
<td>1054</td>
<td></td>
</tr>
<tr>
<td>Primary Key, Create</td>
<td>1052</td>
<td></td>
</tr>
<tr>
<td>Profile (UML)</td>
<td>1039</td>
<td></td>
</tr>
<tr>
<td>Set MySQL Table Type</td>
<td>1046</td>
<td></td>
</tr>
<tr>
<td>Set Oracle Table Properties</td>
<td>1047</td>
<td></td>
</tr>
<tr>
<td>Set Table Owner</td>
<td>1045</td>
<td></td>
</tr>
<tr>
<td>Set Table Properties</td>
<td>1043</td>
<td></td>
</tr>
<tr>
<td>Sorted Lookup Table</td>
<td>1070</td>
<td></td>
</tr>
<tr>
<td>Trigger Operation</td>
<td>1070</td>
<td></td>
</tr>
<tr>
<td>Typical Tasks</td>
<td>1039</td>
<td></td>
</tr>
<tr>
<td>Data Repository</td>
<td>Adaptive Server Anywhere, Connect To</td>
<td>593</td>
</tr>
<tr>
<td></td>
<td>Connect To</td>
<td>545, 584</td>
</tr>
<tr>
<td></td>
<td>Create</td>
<td>574</td>
</tr>
<tr>
<td></td>
<td>MSDE Server, Connect To</td>
<td>595</td>
</tr>
<tr>
<td></td>
<td>MySQL, Connect To</td>
<td>584</td>
</tr>
<tr>
<td></td>
<td>Oracle 10g, Connect To</td>
<td>589</td>
</tr>
<tr>
<td></td>
<td>Oracle 11g, Connect To</td>
<td>589</td>
</tr>
<tr>
<td></td>
<td>Oracle 9i, Connect To</td>
<td>589</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL, Connect To</td>
<td>591</td>
</tr>
<tr>
<td></td>
<td>Progress OpenEdge, Connect To</td>
<td>595</td>
</tr>
<tr>
<td></td>
<td>SQL Server, Connect To</td>
<td>587</td>
</tr>
<tr>
<td>Data Source</td>
<td>Select</td>
<td>1086</td>
</tr>
<tr>
<td>Data Store</td>
<td>Element (Data Flow Diagram)</td>
<td>524</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>Between Repositories</td>
<td>607</td>
</tr>
<tr>
<td></td>
<td>Compare Projects, Instructions</td>
<td>610</td>
</tr>
<tr>
<td></td>
<td>Compare Projects, Why?</td>
<td>609</td>
</tr>
<tr>
<td></td>
<td>Copy Packages Between Projects</td>
<td>611</td>
</tr>
<tr>
<td></td>
<td>Transfer Project Data</td>
<td>608</td>
</tr>
<tr>
<td></td>
<td>Upsize to MySQL</td>
<td>562</td>
</tr>
<tr>
<td>Data Type</td>
<td>Add</td>
<td>789</td>
</tr>
<tr>
<td></td>
<td>Code</td>
<td>789</td>
</tr>
<tr>
<td></td>
<td>Conversion Procedures</td>
<td>1078</td>
</tr>
<tr>
<td></td>
<td>DBMS</td>
<td>1081</td>
</tr>
<tr>
<td></td>
<td>Definition</td>
<td>645</td>
</tr>
<tr>
<td></td>
<td>Delete</td>
<td>789</td>
</tr>
<tr>
<td></td>
<td>Extend</td>
<td>789</td>
</tr>
<tr>
<td>Modify</td>
<td>789</td>
<td></td>
</tr>
<tr>
<td>Programming Language</td>
<td>789</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>Compare Package DDL With</td>
<td>1074</td>
</tr>
<tr>
<td></td>
<td>Design</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>Keys</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>Model Template</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Modeling</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>Schema, Import Of</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>Set Default</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>Supported Types</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>View</td>
<td>1067</td>
</tr>
<tr>
<td>Database Administrator</td>
<td>And Enterprise Architect</td>
<td>283</td>
</tr>
<tr>
<td></td>
<td>Project Role</td>
<td>283</td>
</tr>
<tr>
<td>Database Engineering</td>
<td>Submenu (Project Menu)</td>
<td>100</td>
</tr>
<tr>
<td>Database Modeling</td>
<td>Enterprise Architect</td>
<td>6</td>
</tr>
<tr>
<td>Database Repository</td>
<td>Connect To</td>
<td>553</td>
</tr>
<tr>
<td></td>
<td>Create</td>
<td>553</td>
</tr>
<tr>
<td></td>
<td>Set Up</td>
<td>553</td>
</tr>
<tr>
<td>Database Table</td>
<td>Select From ODBC Data Source</td>
<td>1087</td>
</tr>
<tr>
<td>Datastore</td>
<td>Element</td>
<td>1298</td>
</tr>
<tr>
<td>Datatype</td>
<td>Automation Interface Repository</td>
<td>1609</td>
</tr>
<tr>
<td>DBMS</td>
<td>Conversion Procedures</td>
<td>1078</td>
</tr>
<tr>
<td></td>
<td>Data Type Conversion Procedures</td>
<td>1078</td>
</tr>
<tr>
<td>DBMS Conversion</td>
<td>Mapper</td>
<td>1079</td>
</tr>
<tr>
<td></td>
<td>Procedure</td>
<td>1079</td>
</tr>
<tr>
<td></td>
<td>Table Conversion Between DBMS Types</td>
<td>1079</td>
</tr>
<tr>
<td>DDL</td>
<td>Compare With Database</td>
<td>1074</td>
</tr>
<tr>
<td></td>
<td>Data Modeling</td>
<td>1072</td>
</tr>
<tr>
<td></td>
<td>Default Script Editor</td>
<td>890</td>
</tr>
<tr>
<td></td>
<td>Display In Source Code Viewer</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>Generate For Package</td>
<td>1074</td>
</tr>
<tr>
<td></td>
<td>Generate For Table</td>
<td>1072</td>
</tr>
<tr>
<td></td>
<td>Scripts And Generated Tables</td>
<td>1039</td>
</tr>
<tr>
<td></td>
<td>Transformation</td>
<td>1102</td>
</tr>
<tr>
<td>DDS</td>
<td>MDG Technology For, Enterprise Architect</td>
<td>12</td>
</tr>
<tr>
<td>Debug</td>
<td>.NET</td>
<td>964</td>
</tr>
<tr>
<td></td>
<td>.NET Assembly</td>
<td>954</td>
</tr>
<tr>
<td></td>
<td>.NET CLR Versions</td>
<td>955</td>
</tr>
<tr>
<td></td>
<td>.NET With COM Interop Process</td>
<td>968</td>
</tr>
</tbody>
</table>
Debug
Another .NET Process 969
ASP .NET 965
Enable Capture Of State Transitions In Sequence Diagram 999
Enterprise Architect 6
Invoke Methods 989
Java 964
Java Applets In Internet Browsers 976
Java Web Servers 970, 975
Platforms 964
Set Up Session 951
System Requirements 964
Toolbar 979
Under Windows Vista 964
Variables 986
With Enterprise Architect 951, 963
Workbench Window 981

Debug Session
Generate Sequence Diagram From 991
Java Setup 952
Microsoft Native Setup 956
Record Automatically For Thread 995
Record For Method 992
Record Manually For Thread 994
Recording Markers 995
Set Up For Java 952
Set Up For Microsoft Native 956
Trace Marking 995

Debug Workbench Window
Breakpoints Tab 982
Local Variables 984
Output 985
Recording History 985
Stack Tab 984
Workbench 986

Debugger
Capture State Transitions, Sequence Tab Option 998
Create State Machine For Sequence Diagram Generation 1000
Enable Filter, Sequence Tab Option 959
Filters 959
Inspect Variables at Run-Time 981
Overview 941
Record Arguments To Function Calls, Sequence Tab Option 961
Record Calls To External Modules, Sequence Tab Option 961
Run-Time Inspection 981
Sequence Tab Options 959
System Requirements 964
To Sequence Diagram 992

Using 978
Wildcard in Filter 959

Decision
Element 1298

Deep Copy
Of Diagram 306

Default
Code Language, Set 160
Database, Set 160
Hours 813
Model Diagram, Cancel 103
Model Diagram, Set 103
Project Browser Behavior 73
Templates 1528
UML To XSD Mappings 1019
User Diagram, Cancel 103
User Diagram, Set 103

Default Appearance
Background Color 365
Border Color 365
Border Thickness 365
Font Color 365
Of An Element 348
Of Element 365
Set For Profile Stereotype Objects 1441

Default Diagram
Model, Cancel 103
Model, Cancel For 332
Model, Set 103
Model, Set For 332
User, Cancel 103
User, Set 103

Default Fonts
Model 239
Set 239
User 239

Default Hours
Estimation 813
Per Adjusted Use Case Point 813
Project Management 813
Rate 813
Settings 813

Default Settings
Options Dialog 229

Default Templates
Override in Code Template Editor 1528

Default Tools Toolbar 158

Defect
Add 842
Element 841, 842
Hide Stereotype Letter 842
Show Stereotype Letter 842
Define
    Author 766
    Browser Behavior 231
    Clients 772
    Clipboard Image File Format 231
    Clear Directory 231
    Clear Foreign Key Name Template 1060
    Clear Home Web Site 231
    Clear Internet Search Engine 231
    Clear Resources 771
    Clear Roles 769
    Clear Run-Time Variable 1349
    Clear Stereotype As Metatype 1443
    Clear Stereotype Constraints 1436
    Clear Validation Configuration For MDG Technology 1477

Define Menu Items
    Create Add-In 1532

Defined Environment Types 809

Delegate
    Connector 1384
    Relationship 1384

Delete
    Category In Discussion Forum 259
    Connectors 449
    Connectors, Quick Start 46
    Diagram 303
    Diagrams, Quick Start 46
    Element Changes 839
    Element Defects 839
    Element From Diagram 362
    Element from Model 362
    Element From Project Browser 362
    Element Issues 839
    Element Tasks 839
    Elements, Impact of Auditing 743
    Elements, Quick Start 46
    Instance Variable 1350
    Item In Discussion Forum 259
    Line Points 446
    Linked Document 434
    Linked Document Template 435
    Locks 720
    Package Attributes From Model Document 1200
    Package In Project Browser 293
    Packages From Model Document 1200
    Packages, Quick Start 46
    Post In Discussion Forum 259
    Project Task 847
    Relationship 417

Relationship Using Matrix 483
    RTF Style Template 1139
    Topic In Discussion Forum 259
    Views 619
    Workbench Variables 986

Delete Selected Element(s)
    Menu Option (Edit Menu) 93

Deletion
    And Auditing 743

Delphi
    Import, Reverse Engineering 872
    Modeling Conventions 932
    Options 904
    Properties 905

Demonstration
    Of Enterprise Architect 53

Dependency
    Connector 1385
    Relationship 1385
    Relationship, Apply Stereotype 1385
    Report 1192

Dependency Report
    In Traceability 763

Deploy
    Add-In 1533

Deploy Script
    Create 958

Deployment
    Connector 1386
    Elements and Connectors 144
    Group, Enterprise Architect UML Toolbox 144
    Model Template 66
    Relationship 1386
    View 617

Deployment and Rollout
    And Enterprise Architect 280
    Project Role 280

Deployment Diagram
    Description 1267
    Elements And Connectors 1267
    Example 1267

Deployment Spec
    Element 1343

Design
    Patterns 507
    Design Master 602, 632
    Create 633

Design Systems Using UML
    Enterprise Architect 6

Desktop Edition 9
    Upsize From 553

Desktop Tools
Desktop Tools
   Add 114
   Configure 114
   Customize 114
Destination Role 461
Developer
   And Enterprise Architect 276
   Forward Engineering 276
   Project Role 276
   Reverse Engineering 276
   Round-Trip Engineering 276
   Visualise Package Arrangement 276
Device
   Element 1344
Diagram
   Activity, Description 1213
   Add And Manage, Automation Interface Code Example 1669
   Add Elements Via Context Menu 297
   Add Link To Forum Post 258
   Add Profile Connector 492
   Add To Project 299
   Add To UML Model, Quick Start 39
   Alias 328
   Alternative Image For Element 320
   Analysis 1269
   Anti Aliased Text 234
   Appearance Options, Connectors 331
   Appearance Options, Diagram Tab 328
   Appearance Options, Element 329
   Appearance Options, Features 330
   Appearance Options, General 326
   Appearance Options, Set 325
   Appearance Options, Visible Class Members 332
   Attribute Details, Show 330
   Automatic Layout 300
   Automation Interface, Diagram Package 1651
   Background Color 232
   Behavior Options 235
   Behavioral, Overview 1213
   BPMN 529
   Build 173
   Business Interaction 1276
   Business Modeling 1276
   Business Process 59
   Cancel Model Default 332
   Change Type 305
   Class 63, 1260
   Class Features, Visibility 234
   Collaboration 1253
   Communication 1253
   Component 65, 1265
   Composite Structure 1263
   Connector Appearance Options 331
   Connector Notation, Show 331
   Context 524
   Context Menu 297
   Context Menu, Project Browser 84
   Copy Image 103
   Copy, Deep 306
   Copy, Shallow 306
   Create 299
   Create Automatically 289
   Create Custom Background, Image Manager 321
   Create From Linked Document 434
   Create Using Image Library 321
   Creator 326
   Custom 1270, 1272, 1273, 1274
   Customize Appearance 325
   Data Flow 524
   Data Model, Example 1041
   Database Schema 64, 1275
   Define Child Type, Supported Attributes 1446
   Delete 303
   Delete Element From 362
   Delete Multiple Elements From 362
   Deployment 1267
   Description 1275
   Details Note 309, 328
   Diagram To Package 39
   Display Options 234
   Divide Between Pages In RTF Rept 328
   Drag & Drop Elements From Project Browser 313
   Drag Existing Package Onto 291
   Duplicate 306
   Element Appearance Options 329
   Element Compartments, Show/Hide 329
   Element Stereotypes, Show 329
   Element, Copy 304
   Element, Paste 304
   Elements And Connectors 1275
   Example 1275
   Extended 1212
   Extended UML 1269
   Feature Return Types, Show 330
   Feature Stereotypes, Show 330
   Features Appearance Options 330
   Find In Project Browser 103, 105
   Find Related Elements 314
   Frame 234, 1300
   Frame (Border) 1300
Diagram
Gradient Fill Effect 232
Hyperlink 1300
Increase Display Size 238, 294
Interaction 1228, 1245, 1253, 1255
Interaction Overview 1255
Layout Options 300
Legend 332
Lock 103
Lock, Security Off 324
Logical 1260
Maintenance 1273
Make All Elements Selectable 297
Make Model Default 332
Manage Display 173
Managing 299
MDG Technology 1212
Menu 103
Mind Mapping 522
Modeling With 294
Move Elements 357
Move, Impact On Element 1298, 1305, 1306, 1312
Move, Impact On Elements 44
Navigation And Selection Hotkeys 304
Note 309
Notes 326
Notes, Show/Hide 234
Object 1261
Open From Shortcut 88, 90
Open From Shortcut (Direct Definition) 89
Open Package In 307
Overview 1212
Package 1258
Page Setup 234
Pan And Zoom 224
Paste 306
Place Related Elements On Current 314
Print Page Footer 328
Print Page Header 328
Profiles 1470
Properties (Diagram Menu) 103
Properties Dialog - Connectors Tab 331
Properties Dialog - Diagram Tab 328
Properties Dialog - Elements Tab 329
Properties Dialog - Features Tab 330
Properties Dialog - General Tab 326
Properties Dialog, Visible Class Members, 332
Properties Note, Add 309
Properties, Set 325
Property Strings, Show 331
Qualifiers, Show 330
Redo Last Action 325
Reference 1300
Relationships, Show 331
Rename 304
Requirements 60, 1272
Robustness 1245, 1253, 1269, 1357, 1358, 1360, 1361
Rotate Image In RTF Report 328
RTF Document Options 328, 1136
RTF Report On Elements Linked From Other Packages 1136
Save 103
Save As Pattern 508
Save Changes 297
Save Image Of 103
Save Profile 1442
Schema Diagram 1275
Scroll Through From Diagram Toolbar 163
Sequence 1245
Set Page Size 336
Show As Element List 297
Show Element List As 175
Show Realised Interfaces For Class 323
Show/Hide Package Contents 292
State 1216
State Machine 1216
Stereotype 326
Structural, Overview 1258
Swimlanes 315
Swimlanes Matrix 317
Table Owner, Show 329
Tabs 171
Tabs Menu 171
Tasks, General 299
Timing 1228
Toolbar 163
Traceability 762
Types 1212
Types, Built In 1471
Types, Custom 1470
UML 1212
Undo Last Action 325
Use Case 61, 1215
User Interface 1274
User Interface Design 1270
Version 326
View 173
View Next 325
View Previous 325
Visibility Indicators 330
What Is A? 1212
Working With 294
Distributed Development
   Replication 631
   XML Import/Export 631

Dock Windows
   Navigation Compass 196

Dockable Windows 201

Document
   Enterprise Architect Content 1132
   Exclude Packages 1132
   HTML 1132
   Linking 430
   Projects 1132
   Resource, RTF Generator (Enhanced) 1166
   Rich Text Format 1132
   RTF 1132
   Single Element, RTF (Legacy) 1174
   WSDL Element 1029

Document Artifact
   Element 430, 1344
   Link Into RTF Report 430, 1142

Document Generation
   Enterprise Architect 6

Document Options
   RTF Generator, From Diagram (Enhanced) 1136
   RTF Generator, From Diagram (Legacy) 1136

Documentation
   Elements 1196
   Generate, Project Browser Option 75
   Group, Enterprise Architect UML Toolbox 1196
   HTML 1204
   Rich Text Format 1133
   RTF 1133

Documentation (Reports)
   Submenu (Project Menu) 98

Documentation Submenu
   Package Context Menu, Project Browser 80
   Reports 80

DoD Af-MODAF
   MDG Technology For, Enterprise Architect 12

Domain
   Model Template 62
   Organizational Relationships 62
   Physical Units 62
   Structure 62

DOORS, Telelogic
   MDG Link For, Enterprise Architect 12

Dotted Line Above Menu 200

Download
   GoF Pattern 511

Drag

Elements From Project Browser Onto Diagram 313
Objects From Project Browser Onto Diagram 313

Drawing Methods
   Shape Scripts 1485

Drop
   Classifiers As Links 313
   Classifiers As New Instances 313
   Elements From Project Browser Onto Diagram 313
   Objects From Project Browser Onto Diagram 313

DTD 645

Duplicate
   Diagram 306
   UML Diagram 306

Duration
   Constraint 1399
   Constraint Between Messages 1399
   Observation 1399

Dynamic View 617

- E -

EA_Connect
   Add-In Event 1540

EA_Disconnect
   Add-In Event 1540

EA_FileClose
   Broadcast Events, Add-In Model 1545

EA_FileOpen
   Broadcast Events, Add-In Model 1545

EA_GetCompartmentData
   Compartment Events, Add-In Model 1560

EA_GetMenuItems
   Add-In Event 1541

EA_GetMenuState
   Add-In Event 1541

EA_MenuClick
   Add-In Event 1542

EA_OnContextItemChanged
   Context Item Events, Add-In Model 1557

EA_OnContextItemDoubleClicked
   Context Item Events, Add-In Model 1558

EA_OnDeleteTechnology
   Technology Events, Add-In Model 1556

EA_OnEndValidation
   Model Validation Broadcasts, Add-In Model 1563

EA_OnImportTechnology
   Technology Events, Add-In Model 1556
Index

- EA_OnInitialize_Technologies  
  Broadcast Events, Add-In Model 1570
- EA_OnInitializeUserRules  
  Model Validation Broadcasts, Add-In Model 1562
- EA_OnNotifyContextItemModified  
  Context Item Events, Add-In Model 1558
- EA_OnOutputItemClicked  
  Add-In Event 1543
- EA_OnOutputItemDoubleClicked  
  Add-In Event 1543
- EA_OnPostNewAttribute  
  Post-New Events, Add-In Model 1554
- EA_OnPostNewConnector  
  Post-New Events, Add-In Model 1553
- EA_OnPostNewDiagram (Object)  
  Post-New Events, Add-In Model 1553
- EA_OnPostNewElement  
  Post-New Events, Add-In Model 1552
- EA_OnPostNewMethod  
  Post-New Events, Add-In Model 1554
- EA_OnPostNewPackage  
  Post-New Events, Add-In Model 1555
- EA_OnPostTransform  
  Broadcast Events, Add-In Model 1559
- EA_OnPreDeleteConnector  
  Pre-Deletion Events, Add-In Model 1546
- EA_OnPreDeleteDiagram  
  Pre-Deletion Events, Add-In Model 1547
- EA_OnPreDeleteElement  
  Pre-Deletion Events, Add-In Model 1546
- EA_OnPreDeletePackage  
  Pre-Deletion Events, Add-In Model 1547
- EA_OnPreDeleteTechnology  
  Technology Events, Add-In Model 1555
- EA_OnPreNewAttribute  
  Pre-New Events, Add-In Model 1550
- EA_OnPreNewConnector  
  Pre-New Events, Add-In Model 1549
- EA_OnPreNewDiagram (Object)  
  Pre-New Events, Add-In Model 1549
- EA_OnPreNewElement  
  Pre-New Events, Add-In Model 1548
- EA_OnPreNewMethod  
  Pre-New Events, Add-In Model 1551
- EA_OnPreNewPackage  
  Pre-New Events, Add-In Model 1551
- EA_OnRetrieveModelTemplate  
  Broadcast Events, Add-In Model 1570
- EA_OnRunAttributeRule  
  Model Validation Broadcasts, Add-In Model 1565
- EA_OnRunConnectorRule  
  Model Validation Broadcasts, Add-In Model 1564
- EA_OnRunDiagramRule  
  Model Validation Broadcasts, Add-In Model 1564
- EA_OnRunElementRule  
  Model Validation Broadcasts, Add-In Model 1563
- EA_OnRunMethodRule  
  Model Validation Broadcasts, Add-In Model 1565
- EA_OnRunPackageRule  
  Model Validation Broadcasts, Add-In Model 1564
- EA_OnRunParameterRule  
  Model Validation Broadcasts, Add-In Model 1566
- EA_OnStartValidation  
  Model Validation Broadcasts, Add-In Model 1562
- EA_QueryAvailableCompartments  
  Compartment Events, Add-In Model 1560
- EA_ShowHelp  
  Add-In Event 1544
- EAB File  
  Export 702
  Import 703
  Model Branch File 702
- EABase  
  As Source 599
  Project 52
  Project File 551
- EAExample File 549
- EAP File  
  As Project Database 545
  Corrupt 616
  Iterate Through, Automation Interface Code Example 1667
- ECF  
  Value 809
  Weighting 809
- Eclipse  
  MDG Integration For, Enterprise Architect 12
  MDG Link For, Enterprise Architect 12
- Edit  
  Attribute Name, In-Place Editor 399
  Element Name, In-Place Editor 399
  Linked Document Template 436
  Linked Documents 433
  Menu 93
  Operation Name, In-Place Editor 399
  Pattern Default 512
Edit
Post In Discussion Forum 256
RTF Style Template 1139
Test Details 825
Editions
Corporate 9
Desktop 9
Floating Licence 9
Of Enterprise Architect 9
Professional 9
Standalone 9
Effort
Attributes 1622
Automation Interface, Element Package 1622
Methods 1622
Effort Management 816
Effort Types
Define 820
Global 820
Non-Global 816
EJB
Entity Bean Transformations 1105
Session Bean Transformations 1105
Element
Abort Edit Changes 105
Accept Edit Changes 105
Action 1280
Active 411
Activity 1286
Activity Final 1305
Activity Partition 1318
Activity Region 1320
Actor 1290
Add And Manage, Automation Interface Code Example 1668
Add Attribute, In-place Editor 403
Add Directly To Package 353
Add Link To Forum Post 258
Add New Item (Inline Features Menu Option) 105
Add Operation, In-place Editor 403
Add Supporting Diagrams and Elements 344
Add To Diagram 40
Add To Diagram From Project Browser 310, 313
Add To Diagram Via Context Menu 297
Add To Favorites 345
Add To Profile 1431
Add To UML Model, Quick Start 40
Align 350
Align From Diagram Toolbar 163
Align Multiple 360
Alternative Image 320
Appearance, Context Menu Option 348
Appearance, Format From Toolbar 166
Apply Image From Clipboard 107
Artifact 1336
Associated Files 418
Attributes Option 348
Author 410
Auto Counters 353
Auto Element Naming 353
Automation Interface, Element Package 1623
Autosize Group 310
Autosize Single 310
Background Color 232
Behavioral Diagram 1279
Boundary 1358
Boundary, Settings 424
BPMN 529
BPMN, Change Appearance 531
Browser Window 210
Cardinality 412
Central Buffer Node 1291
Change 841, 843
Change Appearance, BPMN 531
Change Type 359
Changes 838
Changes, Add/Modify/Delete 839
Child Validation 623
Choice 1291
Class 1337
Code Engineering Menu 348
Collaboration 1340
Collaboration Occurrence 1341
Combined Fragment 1292
Compartment 426
Complexity 410
Component 1342
Composite 1359
Concurrency 412
Confirm As Parent 357
Connectors In Relationships Window 211
Constraint Note 1317
Constraint, Attach 105
Constraints 415
Context Menu 341
Context Menu, Add Submenu 344
Context Menu, Advanced 343
Context Menu, Features 348
Context Menu, Find Submenu 345
Context Menu, Project Browser 83
Continuation 1324
Continuatio...
Element

Control 1360
Create Child Diagram 299
Create From Linked Document 434
Create From UML Toolbox 126
Create In Diagram 352
Create Link From Project Browser 448
Create With Quick Linker 226
Create, Same Type As Previous 103
Custom Properties 344
Customize Visibility 363
Dashed Border On 357
Data Store 524
Datastore 1298
Decision 1298
Default Appearance 107
Default Element Template 369
Defect 841
Defects 838
Defects, Add/Modify/Delete 839
Delete 362
Delete Item From 105
Deployment Spec 1343
Details 412
Device 1344
Diagram Frame 1300
Diagram Gate 1301
Display Depth 306
Display Options 238
Document Artifact 1344
Documentation 1196
Drag & Drop Onto Diagram From Project Browser 313
Edit Attribute Keyword 401
Edit Attribute Scope 400
Edit Attribute Stereotype, In-place Editor 399
Edit Item, Tasks 398
Edit Name, In-Place Editor 399
Edit Operation Parameter Keyword 402
Edit Operation Parameter Kind 403
Edit Operation Scope 400
Edit Operation Stereotype, In-place Editor 399
Embedded Submenu 346
Embedded, Add 347
Endpoint 1302
Entity 1361
Entry Point 1303
Enumeration 1344
Event 1362
Exception 1303
Execution Environment 1345
Exit Point 1305

Expansion Region 1303
Export Data In CSV Format 657
Expose Interface 1345
Extended By Stereotype 1357
External 524
External Requirements 414
Feature 1362
Fill Color 232
Final (Leaf) 411
Find In Diagram Menu Options 83
Find in Diagrams 345
Find In Diagrams, Element List 175
Find In Project 185
Find In Project Browser 105, 345
Find Related 314
Flow Final 1306
Font, Set 107
Fork 1307, 1309
Fragment 1292
General Settings 410
Generalizable, Advanced Settings 411
Get Project Custom Colors 367
Gradient Fill Effect 232
Highlight Context 370
History 1311
Hyperlink 1363
Import Data In CSV Format 659
Include Linked In RTF Report 1136
Information Item 1346
Initial 1312
In-place Editor Options 398
Insert Maintenance Feature 405
Insert New Feature (Inline Features Menu Option) 105
Insert Operation Parameter 404
Insert Related Elements 345
Insert Testing Features 407
Instance 1348
Interaction Occurrence 1313
Interface 1347
Internal Requirements 413
Interruptible Activity Region 1313
Issue 841
Issues 838
Issues, Add/Modify/Delete 839
Join 1307, 1310
Junction 1314
Keywords 410
Labels 323
Layout 350
Legend 332
Lifeline 1315
Element
   Link To Attribute Via Object 384
   Linked, Convert To Local Copy 371
   Lock 725
   Lock Indicators 729
   Locked, Add Connector To 726
   Maintenance 838
   Make All Selectable/Unselectable 297
   Make Non-Selectable, Multiple Elements 350
   Make Non-Selectable, Single Element 348
   Make Selectable, Multiple Elements 350
   Make Selectable, Single Element 348
   Managing 352
   Master Document 1196
   Match Size 350
   Menu 105
   Merge 1298
   Merge Node 1316
   Mind Mapping 522
   Model Document 1196
   Modeling With 338
   Move Between Packages 358
   Move In Diagrams 357
   Move, Impact Of Diagram 44
   Multiple Selection 350
   N-Ary Association 1365
   New 352
   Node 1348
   Non-Selectable, Multiple Elements 350
   Non-Selectable, Single Element 348
   Note (Constraint, Comment) 1317
   Note, Attach 105
   Notes 206, 363
   Object 1348
   Occurrence 1313
   Operations Option 348
   Package 1350
   Part 1351
   Partition 1318
   Paste As Link 93
   Paste As New 93
   Paste From Clipboard As Metafile 93
   Paste From Project Browser 310
   Phase 410
   Place Related On Current Diagram 314
   Port 1352
   Postconditions 415
   Preconditions 415
   Primitive 1353
   Process 524, 1366
   Properties Dialog 409
   Properties Dialog, Changes and Defects 844

Properties Menu 342
Properties Window 202
Properties, As Attributes 381
Properties, Edit From Package 1258
Properties, Links 417
Pseudo-State 1220
Receive 1319
Receive Event 1362
Region 1320
Region, Expansion 1303
Region, Interruptible Activity 1313
Relationship Hierarchy For 213
Relationship, Delete 417
Relationship, Hide 417
Relationship, Show 417
Requirement 466, 1366
Requirements 413
Resize 360
Responsibilities 413
Root 411
Rules & Scenarios 212
Scenario 418
Screen 1368
Selectable, Multiple Elements 350
Selectable, Single Element 348
Send 1320
Send Event 1362
Sequence Diagram 1248
Sequence, Lifecycle 1247
Set Alternative Image 107
Set Cross References 356
Set Custom References 356
Set Default Appearance 365
Set Element Template Package 369
Set Feature Visibility 348
Set Font 349
Set Parent 354
Set Project Custom Colors 367
Shadow Color 232
Shape Script Properties 1489
Show Usage 355
Signal 1356
Size 360
Specification 411
State 1321
State Invariant 1324, 1326
State Lifeline 1323
State Machine 1328
State/Continuation 1324
Status 410
Stereotype 410
Element
Structured Activity 1326
Sub-Activity 1286, 1326
Submachine State 1321, 1328
Superimposition 306
Synch 1329
System Boundary 1329
Table 1369
Tagged Values 419
Tasks 352, 838
Tasks, Add/Modify/Delete 839
Template Package 369
Template Parameters 412
Template, Default Element 369
Templates And Profiles 369, 488
Terminate 1330
Test Case 1369
Test Scripts Compartment 835
Text 363
Text Color 232
Toolbar 162
Transformation 1093
Trigger 1330
Type 343
UI Control 1369
UML 1278
Use Case 1331
Use Cross References 356
Use Custom References 356
Use Extras, Automation Interface Code Example 1671
User Interface 1369
Value Lifeline 1333
Version 410
View Properties (Inline Features Menu Option) 105
Visibility 412
Visibility Options 240
Work On From Element List 174
Work On From Toolbar 164
Working With 338
WSDL Binding 1032
WSDL Message 1032
WSDL Namespace 1027
WSDL Service 1030
WSDL, Document 1029
WSDL, Port Type 1031
Z Order 306
Element Context Menu
Add Submenu, Project Browser 83
Element Features
Operations 385
Element List
Context Menu Options 175
Description 174
Generate RTF Report 1135
Options 175
Show As Diagram 175
Show Diagram As 297
Was Report View 174
Work On Elements 175
Element Package, Automation Interface
Constraint 1622
Diagram 1620
Effort 1622
Element 1623
File 1629
Issue 1629
Metric 1630
Requirement 1631
Resource 1631
Risk 1632
Scenario 1633
TaggedValue 1633
Test 1634
Element Templates
And Profiles 1428
Element Features Package, Automation Interface
Attribute 1636
AttributeConstraint 1637
AttributeTag 1638
CustomProperties Collection 1638
Diagram 1635
EmbeddedElements Collection 1639
Method 1639
MethodConstraint 1641
MethodTag 1641
Parameter 1642
Partitions Collection 1643
Properties 1643
Property 1643
Transitions Collection 1644
Elements
Maintenance 837
Email
Access Within Enterprise Architect 194
Exchange Server, Define Default 231
Embedded Elements
Dialog 347
Element Context Submenu 346
Incorporate Inherited Properties 347
Window 347
EmbeddedElements Collection
Embedded Elements Collection
  Automation Interface, ElementFeatures Package 1639

EMX
  File Cross References 642
  File Import 642

Enable
  Add-Ins 1537
  MDG Technologies 517
  Security 709

Enable Filter
  Debugger Sequence Tab Option 959

Encrypt Password
  Prior To Release 7.1 Of Enterprise Architect 721

Encrypt Password (Repository)
  At Release 7.1 Of Enterprise Architect 88, 92

End User License Agreement 18

Endpoint
  Element 1302

Enterprise Architect
  Add-In Model 1531
  Alignment With UML 1210
  And Deployment 280
  And IDEs 6
  Arrange Windows and Menus 195
  Autohide Windows 199
  Build Systems 6
  CASE Tool 3
  Code Engineering With 34
  Connectors 1373
  Create UML Project, Tutorial 35
  Database Modeling 6
  Debug 6
  Demonstration 53
  Design Systems 6
  Dockable Windows 196, 201
  Editions, Differences Between 9
  Editor 890
  End User License Agreement 18
  Example Project File 549
  Export Data In CSV Format 657
  Feedback Pages 3
  For Business Analysts 271
  For Database Administrators 283
  For Developers 276
  For Implementation Managers 280
  For Project Managers 278
  For Software Architects 273
  For Software Engineers 275
  For Technology Developers 281
  For Testers 279

Formal Statements 16
Fundamental Processes 30
Generate Documentation 6
Generate Source Code 6
Glossary 1678
Help 13
How To Use 3
Import Data In CSV Format 659
In Action 53
Install 25
Interfaces For UML Modeling 51
Introduction 3
Key Features 8
Keyboard Shortcuts 247
License Agreement 18
Main Menu 86
Manage Model Structure 6
Manage Requirements 6
MDA Transformation 6
Model Complexity 6
Model Requirements 6
Navigation Compass 196
Object Model, Introduction 1583
Online Resources 55
Online User Guide 3
Order 24
Predefined Search Definitions 191
Pricing And Purchasing 24
Professional Roles 270
Project Discussion Forum 251
Project Files, Open A Project 549
Project Roles 270
Project, What Is A? 547
Quick Start 29
Register Full License 26
Replication Merge Rules 632
Reverse Engineer 6
SDK, Introduction 1427
Share Model Development 6
Software Product License Agreement 18
Spell Checking 264
Start 52
Start Page 55
Support 13, 15
Tear Off Menus 200
Tools And Features For UML Modeling 51
Trial Version 23
Tutorial, UML Modeling 29
UML 2.1 Support 6
UML Model Management 542
UML Modeling Tool 3
UML Modeling With 285
<table>
<thead>
<tr>
<th>Enterprise Architect</th>
<th>53</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>53</td>
</tr>
<tr>
<td>Uses Of</td>
<td>6</td>
</tr>
<tr>
<td>Visualize Systems</td>
<td>6</td>
</tr>
<tr>
<td>Web Services Access</td>
<td>194</td>
</tr>
<tr>
<td>What Can I Do With It?</td>
<td>6</td>
</tr>
<tr>
<td>What is Enterprise Architect?</td>
<td>5</td>
</tr>
<tr>
<td>Working With</td>
<td>29, 30</td>
</tr>
<tr>
<td>Workspace Status Bar</td>
<td>169</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enterprise Architect UML Toolbox</th>
<th>126</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Group</td>
<td>142</td>
</tr>
<tr>
<td>Analysis Group</td>
<td>147</td>
</tr>
<tr>
<td>Class Group</td>
<td>135</td>
</tr>
<tr>
<td>Common Group</td>
<td>133</td>
</tr>
<tr>
<td>Communication Group</td>
<td>138</td>
</tr>
<tr>
<td>Component Group</td>
<td>143</td>
</tr>
<tr>
<td>Composite Group</td>
<td>137</td>
</tr>
<tr>
<td>Custom Group</td>
<td>148</td>
</tr>
<tr>
<td>Data Modeling Group</td>
<td>155</td>
</tr>
<tr>
<td>Deployment Group</td>
<td>144</td>
</tr>
<tr>
<td>Documentation Group</td>
<td>1196</td>
</tr>
<tr>
<td>Interaction Group</td>
<td>139</td>
</tr>
<tr>
<td>Maintenance Group</td>
<td>150</td>
</tr>
<tr>
<td>MDG Technology Groups</td>
<td>520</td>
</tr>
<tr>
<td>Metamodel Group</td>
<td>146</td>
</tr>
<tr>
<td>Object Group</td>
<td>136</td>
</tr>
<tr>
<td>Profile Group</td>
<td>145</td>
</tr>
<tr>
<td>Requirement Group</td>
<td>149</td>
</tr>
<tr>
<td>Shortcut Menu</td>
<td>131</td>
</tr>
<tr>
<td>State (Machine) Group</td>
<td>141</td>
</tr>
<tr>
<td>Timing Group</td>
<td>140</td>
</tr>
<tr>
<td>Use Case Group</td>
<td>134</td>
</tr>
<tr>
<td>User Interface Group</td>
<td>151</td>
</tr>
<tr>
<td>WSDL Group</td>
<td>153</td>
</tr>
<tr>
<td>XML Schema Group</td>
<td>154</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enterprise Java Beans</th>
<th>1105</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDG Technology For, Enterprise Architect</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity</th>
<th>1361</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>1361</td>
</tr>
<tr>
<td>Element</td>
<td>1303</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>1303</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>1303</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>1595</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Interface</td>
<td>1592</td>
</tr>
<tr>
<td>ConstLayoutStyles</td>
<td>1593</td>
</tr>
<tr>
<td>Element</td>
<td>1344</td>
</tr>
<tr>
<td>EnumRelationSetType</td>
<td>1593</td>
</tr>
<tr>
<td>Literal</td>
<td>1344</td>
</tr>
<tr>
<td>MDGMenus</td>
<td>1593</td>
</tr>
<tr>
<td>ObjectType</td>
<td>1594</td>
</tr>
<tr>
<td>PropType</td>
<td>1594</td>
</tr>
<tr>
<td>ReloadType</td>
<td>1595</td>
</tr>
<tr>
<td>XMIType</td>
<td>1595</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enumeration Elements</th>
<th>1438</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add To Profiles</td>
<td>1438</td>
</tr>
<tr>
<td>EnumRelationSetType</td>
<td>1593</td>
</tr>
<tr>
<td>EnumValue</td>
<td>1344</td>
</tr>
<tr>
<td>Environment Complexity Factor</td>
<td>1593</td>
</tr>
<tr>
<td>Definition</td>
<td>809</td>
</tr>
<tr>
<td>Estimate Project Size</td>
<td>811</td>
</tr>
<tr>
<td>Estimation</td>
<td>809</td>
</tr>
<tr>
<td>Value</td>
<td>809</td>
</tr>
<tr>
<td>Weighting</td>
<td>809</td>
</tr>
<tr>
<td>Eriksson-Penker Business Extensions</td>
<td>1269</td>
</tr>
<tr>
<td>Estimation</td>
<td>784</td>
</tr>
<tr>
<td>Default Hours</td>
<td>813</td>
</tr>
<tr>
<td>Environment Complexity Factors</td>
<td>809</td>
</tr>
<tr>
<td>Of Project Factors</td>
<td>806</td>
</tr>
<tr>
<td>Of Project Size</td>
<td>811</td>
</tr>
<tr>
<td>Of Project Timescale</td>
<td>806</td>
</tr>
<tr>
<td>Technical Complexity Factors</td>
<td>807</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event</th>
<th>1362</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>1362</td>
</tr>
<tr>
<td>Receive</td>
<td>1362</td>
</tr>
<tr>
<td>Send</td>
<td>1362</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EventProperties</th>
<th>1610</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Interface Repository</td>
<td>1610</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EventProperty</th>
<th>1611</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Interface Repository</td>
<td>1611</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples And Tips</th>
<th>1586</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>1303</td>
</tr>
<tr>
<td>Exclamation Mark</td>
<td>729</td>
</tr>
<tr>
<td>Blue</td>
<td>729</td>
</tr>
<tr>
<td>Red</td>
<td>729</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exclude</th>
<th>1137</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package In RTF Report</td>
<td>1137</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution Environment</th>
<th>1345</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>1345</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exit</th>
<th>87</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu Option (File Menu)</td>
<td>87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exit Point</th>
<th>1305</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>1305</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expansion Region</th>
<th>1305</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>1305</td>
</tr>
<tr>
<td>Element</td>
<td>1305</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Explorer</th>
<th>109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>109</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Export</th>
<th>702</th>
</tr>
</thead>
<tbody>
<tr>
<td>.EAB File</td>
<td>702</td>
</tr>
<tr>
<td>Code Templates</td>
<td>1527</td>
</tr>
<tr>
<td>Data</td>
<td>790</td>
</tr>
<tr>
<td>MOF To XMI</td>
<td>666</td>
</tr>
</tbody>
</table>

© 1998-2008 Sparx Systems Pty Ltd
Export
Profile 1441
Reference Data 790
Reference Data, Introduction 790
RTF Style Template 1139
State Machine Table To CSV 1228
To Rational Rose 636, 638, 644
To XMI 638
UML Profile 1441
Version Controlled Model Branch 702
XMI, Batch 650
Export Diagrams
To RTF Document 1193
Expose Interface
Element 1345
Extend
Connector 1387
Relationship 1387
UML Toolbox Connectors 1468
UML Toolbox Elements 1467
Extended Elements 1357
Extended UML Diagrams 1269
Extension Points
Use Case 1332
Extension Stereotypes 1357
External
Element (Data Flow Diagram) 524
Requirements 414, 466
Requirements, Color Coded 468
External Tools
Open 116
Pass Parameters To 117

- F -
Favorites
Add To Resources Window 205
Delete From Resources Window 205
Drag Objects Into 177, 179
Elements, Resources Window 205
Folder, Resources Window 205
Model Views Folder 177
View Properties In Resources Window 205
FDD Methodology 1362
Feature
As Attribute 374
As Operation 385
Element 1362
Feature Driven Design Methodology 1362
Feature Visibility
Attributes 307
Customize 307
Inherited 307
Operations 307
Set 307
Supress 307
Features
Of Enterprise Architect 8
Feedback Pages 3
Field Substitution Macros
Access Data from Attributes 1509
Access Data from Classes 1509
Access Data from Operations 1509
Access Data from Packages 1509
Access Data from Parameters 1509
Conditional Substitution 1509
Direct Substitution 1509
Fields and Conditions
In Search 193
File
.EAB 702
Element Package, Automation Interface 1629
Hyperlink To 1363
Menu 87
Filters
Add To Search 192
AND 185
OR 185
Search 185
Find
Diagram In Project Browser 103, 105
Element In All Diagrams 345
Element In Diagrams, Element List 175
Element In Project Browser 105, 345
Submenu 345
Find In Diagram
Element Context Menu, Project Browser 83
Find in Project
Dialog, Simple 185
Menu Option (Edit Menu) 93
Results 181
Find In Project Browser
Search Option 79
Floating Licence
Version of Corporate Edition 9
Floating Windows 196
Flow Final
Element 1306
Font
Set For Element Text 349
Fonts
Set Model Default 239
Set User Default 239
Garden Of Eden Style 1021
General
   Types, Dialog 774
   General Options
      Set For Project 231
   General Ordering
   - G -
   Sequence Diagram Messages 1401
   Generalizable Elements 411
   Generalization
      Sets 450
   Generalization Link
      Implement Parent Operations 395
      Override Parent Operations 395
   Generalize
      Connector 1388
      Relationship 1388
   Generate
      Global Element For Global ComplexTypes 1020
      Global Element In XSD 1021
      Report On Elements 174
      Report On Project 174
      Rich Text Format Report (Legacy) 1179
      RTF Document from Resources (Enhanced) 1166
      RTF Document From Resources (Legacy) 1181
      RTF Report (Legacy) 1179
      RTF Report From Element List 174
      RTF Report From Virtual Document 1202
      Sequence Diagram From Method 992
      Sequence Diagram In Debug Session 991
      WSDL 1036
      XML Schema For Referenced Packages 1020
      XML Schema, For Child Packages 1020
      XSD 1020
   Generate HTML Report Dialog 1206
   Generate RTF Documentation
      From Model Search 181
   Generate RTF Documentation Dialog
      Options 1137
   Generate RTF Report
      Update Links In Word 1190
   Generate Source Code
      Enterprise Architect 6
      Overview 879
   Get Function
      Create As Attribute Property 381
   Get Project Custom Colors
      For Element 367
   Getting Started
      Add License Key 797
      Installing Enterprise Architect 25
      License Information 796
      License Management 794
      Quick Start Guide To Enterprise Architect 29
      Register Enterprise Architect 26
      Registration Key 796
HTML
Quick Start, Generate Report 1205
Report 1132, 1204
Report, Create 1205

Hyperlink
As Sub Activities 1363
Diagrams 1363
Element 1363
In Linked Document 434
Insert In RTF Template 1157
Insert New From Toolbar 162
To Discussion Forum 1363
To Element From Linked Document 434
To External Files 1363
To Help Topics 1363
To Internet Facilities 1363
To Matrix Profiles 1363
To Model Search 1363

Icon
Attribute Private 76
Attribute Protected 76
Checked In Package 76
Checked Out Package 76
For MDG Technologies 1476
Namespace Root Package 76
Operation Private 76
Operation Protected 76
Version Controlled Package 76

ICONIX
Disable 526
Elements 526
Layout 526
MDG Technology 526
Process 526
Relationships 526
Roadmap 526
UML Toolbox Pages 526

Icons
For Toolbox Items, Assign 1466

IDE
And Enterprise Architect 6

Image
Change Linked To Embedded 1183
Handling, RTF 1183
Print Scaled 335
Refresh 320
Scale To Page Size 335
Select Alternative 320

Store In Enterprise Architect 320
Use From Image Library 321

Image Library
Import 321
Use 321

Image Manager
Create Custom Diagram Background 321
Select Alternative Image For Element 320
Use 320

Implementation
Of Requirements 473

Implementation Manager
And Enterprise Architect 280
Project Role 280

Implementation Report 1193
In Traceability 763
Target Types 1195

Implemented Interfaces
Generate/Disable Methods For 892

Implements
Connector 1417
Relationship 1417

Import
.EAB File 703
ActionScript, Reverse Engineering 872
Binary Module, Reverse Engineering 875
C#, Reverse Engineering 872
C, Reverse Engineering 872
C++, Reverse Engineering 872
Code Templates 1527
Code, Select Language 160
Component Types 890
Data 791
Database Schema from ODBC 1083
DDL Schema from ODBC 1083
Delphi, Reverse Engineering 872
Directory Structure, Reverse Engineering 874
EMX files 642
From XMI 640
Global Elements 1023
Handle Classes Not Found 877
Image Library 321
Java, Reverse Engineering 872
MDA-Style Transformations 1096
MDG Technologies 515
MOF From XMI 666
Pattern 511
PHP, Reverse Engineering 872
Python, Reverse Engineering 872
Reference Data 791
Reference Data, Introduction 790
Referenced XML Schema 1023
Import
- RTF Style Template 1139
- RTF Template 1165
- Scenario as Test 831
- Scenarios From Package 831
- Source Code, Reverse Engineering 870, 877
- Technology Files 109
- Test From Other Element 833
- UML Pattern 511
- UML Profiles 490
- UML2 files 642
- Version Controlled Model Branch 703
- Visual Basic, Reverse Engineering 872
- Visual Basic.Net, Reverse Engineering 872
- WSDL 1037
- XMI 650
- XSD 1023

Import/Export
- Submenu (Project Menu) 102
- Import/Export Submenu
  - Package Context Menu, Project Browser 81

Imported Class Elements 1088

Inbuilt Stereotypes 1357

Include
- Connector 1389
- Package In RTF Report 1137
- Relationship 1389

Increase Text Display Size 201

Index
- Create in Data Modeling 1070
- What Is An? 1070

Indicator
- Locked Element 729
- Visibility 330

Information Flow
- And Patterns 1390
- Connector 1390
- In Combination 1390
- Realized 1392
- Relationship 1390

Information Item
- Conveyed 1391
- Element 1346

Inheritance
- Connector 1388
- Relationship 1388

Inherited Attributes
- Display 383
- Inherited Feature
  - Show 307
- Inherited Operation
  - Display 396

Inherited Port
- Manage 1353
- Initial
  - Element 1312

Initial Code
- Operations Properties 393

Inline Features
- Submenu (Element Menu) 105

Inline Sequence Elements
- Part And Port 1252

InnoDB
- BaseModel Script 562

In-place Editor
- Connector Options 453
- Edit Attribute Keyword 401
- Edit Attribute Name 399
- Edit Attribute Scope 400
- Edit Attribute Stereotype 399
- Edit Connector Labels 453
- Edit Element Name 399
- Edit Operation Name 399
- Edit Operation Parameter Keyword 402
- Edit Operation Parameter Kind 403
- Edit Operation Scope 400
- Edit Operation Stereotype 399
- Element Item, Tasks 398
- Element Options 398
- Insert Attribute To Element 403
- Insert Maintenance Feature 405
- Insert Operation Parameter 404
- Insert Operation To Element 403
- Insert Testing Feature 407

Insert
- Attribute To Element, In-place Editor 403
- Bookmark RTF Into Word 1184
- Boundary Element 424
- Diagram Properties Note 309
- Linked Elements 345
- Maintenance Feature In Element 405
- Operation To Element, In-place Editor 403
- Related Elements 345
- Testing Features In Element 407

Install
- Enterprise Architect 25

Instance
- Define Behavior On Creation, Supported Attributes 1445
- Paste Object As 310

Instance Classifier
- Set 424

Integration Testing
- Display Details 827
Java
  Web Servers, Debugging 970
JBOSS
  Server Configuration 973
  Server, Debugging 970
Join
  Element 1307, 1310
  Pseudo-State 1307, 1310
Junction
  Element 1314
JUnit Transformation
  MDA-Style Transform 1109

- K -

Key
  Combinations 247
Key Features
  Of Enterprise Architect 8
Keyboard
  Accelerator Map 247
  Shortcuts 247
  Shortcuts, Customize 118
  Shortcuts, Reset 118
Keys
  Foreign, Definition 1039
  Primary, Definition 1039
Keystore
  Troubleshooting 799
Keywords
  For RTF Report (Legacy) 1181

- L -

Label
  Alignment 323
  Bold 323
  Connector 323
  Connector, Hide/Show 441
  Direction 323
  Element 323
  Hide 323, 452
  Menu 323
  Set Color 323
  Show 452
  Show And Hide In Toolbox 128
  Visibility 452
Label Visibility
  On Sequence Messages 1251
Labels
  Display Under Toolbar Icons 111
Language
  Adjust in RTF (Legacy) 1181
  Adjust in RTF Report 1171
  Create Properties As Attributes 381
  Custom, Create Templates For In Code
  Template Editor 1529
  For Spell Check 268
  Macros 897
  Options 901
  Options, C 902
  Options, C++ 903
Layout
  Custom, Save 245
  Default Desktop 245
  Diagram, Automatically 300
  ICONIX 526
  Sequence Diagram 1247
Legend
  Add To Diagram 332
  Add To State Machine Table 1227
  Edit 332
  Element 332
  Insert New From Toolbox 162
  Remove From State Machine Table 1227
  Style 332
Level Numbering
  Show In Project Browser 79
License
  Agreement 18
  Information 796
  Management 794
Lifecycle
  Of A Sequence Element 1247
Lifeline
  Element 1315
  Sequence Element, Termination 1247
Limitations
  Of XMI 644
Line
  Angles, Tidy 446
  Bend At Cursor 446
  Bezier 446
  Straighten At Cursor 446
  Suppress Segments 446
Line Points
  Add 446
  Delete 446
  Toggle 446
Link
  Add Note 443
  Create Between Elements 448
  Create From Project Browser 448

Enterprise Architect User Guide
MDG Add-In For
Teamcenter Systems Engineering, Siemens PLM 12
Testing 12
The Open Group Architecture Framework 12
TOGAF 12
Visio 12
Visual Studio 2005/2008 12
Visual Studio.NET 12
Zachman Framework 12

MDG Add-Ins
Add-In Model 1573
MDG Events 1573
MDG_BeBuildProject 1573
MDG_Connect 1574
MDG_Disconnect 1574
MDG_GetConnectedPackages 1575
MDG_GetProperty 1575
MDG_Merge 1576
MDG_NewClass 1577
MDG_PostGenerate 1578
MDG_PostMerge 1579
MDG_PreGenerate 1579
MDG_PreMerge 1580
MDG_PreReverse 1580
MDG_Run_Exe 1581
MDG_View 1581

MDG Events
Add-In Model 1573
MDG_BeBuildProject 1573
MDG_Connect 1574
MDG_Disconnect 1574
MDG_GetConnectedPackages 1575
MDG_GetProperty 1575
MDG_Merge 1576
MDG_NewClass 1577
MDG_PostGenerate 1578
MDG_PostMerge 1579
MDG_PreGenerate 1579
MDG_PreMerge 1580
MDG_PreReverse 1580
MDG_Run_Exe 1581
MDG_View 1581

MDG Integration For
Eclipse 12
TcSE 12
Teamcenter Systems Engineering, Siemens PLM 12
Visual Studio 2005/2008 12

MDG Link For
DOORS, Telelogic 12
Eclipse 12

Visio 12
Visual Studio.NET 12

MDG Technologies
Access, Remote From Enterprise Architect 518
Activate 517
And Resources Window 520
BPMN 514
Create 1454
Data Flow Diagrams 514
Define Validation Configuration 1477
Deploy From Add-In 1479
Deploy From File 1479
Disable 517
Enable 517
Iconix 514
Icons 1476
Import 515
In SDK 1453
Incorporate Model Template 1478
Introduction 514
Link for Downloads 514
Logos 1476
Manage 517
Mind Mapping 514
Toolbox Groups 520

MDG Technology
Active, Automatically Pinned Pages 128

MDG Technology (Integrated)
BPMN 529
Data Flow Diagram 524
ICONIX Process 526
Mind Mapping 522, 529

MDG Technology For
CORBA 12
Data Distribution Service 12
DDS 12
Department Of Defense Architecture Framework 12
- Ministry Of Defence Architecture Framework 12
DoDAF-MODAF 12
Enterprise Java Beans 12
Python 12
SysML 12
Systems Modeling Languages 12
Testing 12
The Open Group Architecture Framework 12
TOGAF 12
Zachman Framework 12

MDG Technology Wizard
Add Code Modules 1460
Add Images 1462
MDG Technology Wizard
  Add MDA Transforms 1462
  Add Pattern 1458
  Add Profile 1457
  Add Tagged Value Types 1459
  Create Technologies 1454

MDG_BuildProject
  Add-In Model 1573

MDG_Connect
  Add-In Model 1574

MDG_Disconnect
  Add-In Model 1574

MDG_GetConnectedPackages
  Add-In Model 1575

MDG_GetProperty
  Add-In Model 1575

MDG_Merge
  Add-In Model 1576

MDG_NewClass
  Add-In Model 1577

MDG_PostGenerate
  Add-In Model 1578

MDG_PostMerge
  Add-In Model 1579

MDG_PreGenerate
  Add-In Model 1579

MDG_PreMerge
  Add-In Model 1580

MDG_PreReverse
  Add-In Model 1580

MDG_Run_Exe
  Add-In Model 1581

MDG_View
  Add-In Model 1581

MDGMenus Enum
  Automation Interface 1593

Member Variables
  Inspect During Debugging 981

Menu
  Add-Ins 122
  Context 53
  Customize Appearance 120
  Diagram 103
  Diagram Context 297
  Edit 93
  Element 105
  Element Context 341
  File 87
  Help 125
  Items, Define In Add-In 1532
  Label 323
  Main 86
  More Windows (View Menu Option) 95
  New Element Or Connector Context 297
  Project 98
  Rearrange 195
  Set Animation 120
  Set Shadowing 120
  Settings (Project) 123
  Show Grid (View Menu Option) 95
  Snap to Grid (View Menu Option) 95
  Tear Off 200
  Toolbars (View Menu Option) 95
  Tools 109
  View 95
  Visual Layouts (View Menu Option) 95
  Visual Styles (View Menu Option) 95
  Window 124

Menu Option (File Menu)
  Close Project 87
  Exit 87
  New Project 87
  Open Project 87
  Open Source File 87
  Page Setup 87
  Print 87
  Print Preview 87
  Print Setup 87
  Reload Current Project 87
  Save Project As 87

Merge
  Baseline With Current Model, Overview 745
  Element 1298, 1316
  Node 1316
  Options, Compare Utility 752

Merge Packages
  Relationship 1416

Message
  Collaboration 1404
  Colors In Communication Diagrams 1254
  Communication 1403
  Communication, Coloring 242
  Communication, Create 1403
  Connector 1395
  Create On Timing Diagram 1407
  Endpoint 1316
  Group, Start New 1404
  Label 1317
  Move 1395
  Recursion 1418
  Relationship 1395
  Scope 463
  Self Message 1398
  Self Message Call 1399
Index 1754

Message
- Sequence Communication 1404
- Sequence Diagram, Examples 1402
- Sequence Diagram, General Ordering 1401
- Sequence Diagram, Self Message 1398
- Sequence, Create 1395
- Sequence, Label Visibility 1251
- Sequencing 1404
- Source and Target 463
- Timing Diagram 1406
- WSDL Diagram 1032
- WSDL Element 1032

Message Angle
- Adjust With Duration Constraint 1399

Message Dialog
- Discussion Forum 257

Message Part
- WSDL Attribute 1035

Meta Object Facility
- Introduction 661

Metaclass
- Add To Profile 1431

META-INF Package 1105

Metamodel
- Elements and Connectors 146
- Group, Enterprise Architect UML Toolbox 146

Method
- Add And Delete, Automation Interface Code Example 1670
- Automation Interface, Element Features Package 1639
- Context Menu, Project Browser 85
- Delete If Not In Code In Reverse Synchronization 892
- Generate Sequence Diagram 992
- Implemented Interfaces 892
- Include Bodies In Model When Reverse Engineering 892
- Record Debug Session For 992
- Show Parameters On Diagram 332
- Work With, Automation Interface Code Example 1675

Method Constraint
- Automation Interface, Element Features Package 1641

Methodology
- FDD 1362
- Feature-Driven Design 1362

Method Tag
- Automation Interface, Element Features Package 1641

Metric
- Automation Interface, Element Package 1630

Metric Types
- Define 821
- Global 821
- Non-Global 818

Metrics 784
- Project Management Window 220

Metrics And Estimation
- Default Hour Rate 813
- ECF 809
- Effort Types 820
- Environment Complexity Factors 809
- For An Element 818
- Metric Types 821
- Risk Types 822
- TCF 807
- Technical Complexity Factors 807

Microsoft Native
- Set Up Debug Session 956

Microsoft Word
- Special Considerations 1186
- Use In RTF Documentation 1183

Migration
- From UML 1.3 604
- To UML 2.0 604

Mind Mapping
- Concept 522
- Diagram 522
- Disable 522
- Elements 522
- MDG Technology 522
- Relationship 522
- Toolbox Page 522

Misc Data 1623

Model
- Add To Project 545
- Automation Interface 1589
- Context Menu, Project Browser 77
- Data Integrity 603
- Databases 1039
- Default Diagram, Cancel 332
- Default Diagram, Set 332
- Default Fonts 239
- Delete Element From 362
- Delete Multiple Elements From 362
- Glossary 857
- Integrity Check 604
- Integrity, Run SQL Patches 606
- Issues 849
- Pattern 552
- Remove Recent 57
- Root Node 77
- Sharing 628
Model
Shortcut 88
Shortcut (Capture Current Environment) 90
Shortcut (Direct Definition) 89
Structure 285
Templates, Incorporate In Technology 1478
Transformation 1093
UML, Manage In Enterprise Architect 33
WSDL 1027
WSDL, Binding 1032
WSDL, Document 1029
WSDL, Message 1032
WSDL, Message Part 1035
WSDL, Namespace 1027
WSDL, Port Type 1031
WSDL, Port Type Operation 1034
WSDL, Service 1030
XSD 1010

Model Branch
.eab File 702
Apply Version Control 701
Check In 699
Export 702
File 702
Import 703

Model Changes
Auditing 732
Record 732

Model Context Menu
Package Control Submenu, Project Browser 78

Model Document
Add Packages As Attributes 1200
Change Package Sequence 1201
Delete Package Attributes 1200
Delete Packages 1200
Document Order 1201
Element 1196
Element, Create 1199
Generate Documentation 1202
Identify Search For 1199
Model Search Sequence 1201
Move Package Attributes Between Elements 1201
Overview, Virtual Document 1196
Select Template For 1199
Sequence of Model Document Elements 1201
Sequence of Package Attributes 1201
Tagged Values 1199

Model Driven Architecture
Overview 1090

Add-Ins 12

Model File
Connect To ASA Data Repository 593
Connect To Data Repository 584
Connect To MSDE Server Data Repository 595
Connect To MySQL Data Repository 584
Connect To Oracle 10g Data Repository 589
Connect To Oracle 11g Data Repository 589
Connect To Oracle 9i Data Repository 589
Connect To PostgreSQL Data Repository 591
Connect To Progress OpenEdge Data Repository 595
Connect To SQL Server Data Repository 587
Create Adaptive Server Anywhere Repository 581
Create Data Repository 574
Create Model 551
Create Model Files Discussion 598
Create MSDE Server Repository 583
Create MySQL Repository 574
Create Oracle 10g Server Repository 578
Create Oracle 11g Server Repository 578
Create Oracle 9i Server Repository 578
Create PostgreSQL Repository 579
Create Progress OpenEdge Repository 583
Create SQL Server Repository 576
Enterprise Architect Project Files 545
Open Model Files Discussion 598
Set Up Adaptive Server Anywhere ODBC Driver 569
Set Up Database Model Files 553
Set Up MySQL ODBC Driver 564
Set Up ODBC Driver 563
Set Up PostgreSQL ODBC Driver 566
Set Up Progress OpenEdge ODBC Driver 573

Model Glossary
Add Item, Glossary Detail Dialog 860
Add Item, Glossary Dialog 858
Delete Item, Glossary Detail Dialog 860
Delete Item, Glossary Dialog 858
Glossary Dialog 858
Glossary Report 862
Modify Item, Glossary Detail Dialog 860
Modify Item, Glossary Dialog 858
Project Glossary Tab 860

Model Maintenance (.EAP)
Compact Project 615
Introduction 613
Rename Project 614
Repair Project 616

Enterprise Architect User Guide
<table>
<thead>
<tr>
<th>Model Node</th>
<th>Package Control Submenu, Project Browser</th>
<th>78</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Pattern</td>
<td>Introduction 58</td>
<td></td>
</tr>
<tr>
<td>Model Profile</td>
<td>Apply 317</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Save 317</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Swimlanes Matrix 317</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zachman 317</td>
<td></td>
</tr>
<tr>
<td>Model Search</td>
<td>Access From Add-In 184, 1538</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Access From MTS File 184</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Access From Shortcut 88, 90, 184</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Access From Shortcut (Direct Definition) 89</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add Filters 192</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create Search Definition 189</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define In MTS File 1464</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Export Search Definitions 185</td>
<td></td>
</tr>
<tr>
<td></td>
<td>External Access 184</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generate RTF Documentation 181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generate RTF Report 1135</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hyperlink To 184, 1363</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Introduction 181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Options 181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Process Results Of Search 181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To Populate Model Views 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use 184</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View 181</td>
<td></td>
</tr>
<tr>
<td>Model Template</td>
<td>Business Process 59</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Class 63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Component 65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Database 64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deployment 66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Domain 62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Introduction 58</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>Logical 63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintenance 68</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Physical 66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project 69</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relational Database 64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requirements 60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Testing 67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use Case 61</td>
<td></td>
</tr>
<tr>
<td>Model Transformations</td>
<td>Submenu (Project Menu) 100</td>
<td>100</td>
</tr>
<tr>
<td>Model Validation</td>
<td>Configure 622</td>
<td>622</td>
</tr>
<tr>
<td></td>
<td>Define Configuration For MDG Technology 1477</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Element Composition Rule 623</td>
<td>623</td>
</tr>
<tr>
<td></td>
<td>Object Constraint Language 620</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OCL 620</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OCL Conformance Rule 624</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Property Validity Rule 624</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rule, Element Composition 623</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rule, OCL Conformance 624</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rule, Property Validity 624</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rule, Well Formedness 623</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rules 623</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Submenu (Project Menu) 101</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>Well Formedness Rule 623</td>
<td></td>
</tr>
<tr>
<td>Model Validation Broadcasts</td>
<td>Add-In Model 1562</td>
<td>1562</td>
</tr>
<tr>
<td></td>
<td>EA_OnEndValidation 1563</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnInitializeUserRules 1562</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunAttributeRule 1565</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunConnectorRule 1564</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunDiagramRule 1564</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunElementRule 1563</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunMethodRule 1565</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunPackageRule 1564</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnRunParameterRule 1566</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EA_OnStartValidation 1562</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model Validation Example 1566</td>
<td></td>
</tr>
<tr>
<td>Model Views</td>
<td>Context Menu Options 178</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>Define In MTS File 1464</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define Search 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete (Context Menu) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete (Toolbar) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Export As XML 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Favorites 177</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Favorites Folder, Create (Context Menu) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Favorites Folder, Create (Toolbar) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Import As XML 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Move Objects Between Views 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Move Objects Into Favorites 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>My Views 177</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>Operations 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Properties 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Refresh (Toolbar) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Root Node, Create (Context Menu) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Root Node, Create (Toolbar) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Technology-Defined 177</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Toolbar Options 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use Objects In Views And Favorites 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View, Create (Context Menu) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View, Create (Toolbar) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Views 177</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>Views Folder, Create (Context Menu) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Views Folder, Create (Toolbar) 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Window 177</td>
<td>177</td>
</tr>
<tr>
<td>Model Violations</td>
<td>Examples 620</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>Model Wizard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Model</td>
<td>552</td>
<td></td>
</tr>
<tr>
<td>Quick Start</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Use To Create Model</td>
<td>551</td>
<td></td>
</tr>
<tr>
<td>Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Business Process</td>
<td>533</td>
<td></td>
</tr>
<tr>
<td>With UML, Enterprise Architect Overview</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Modeling Conventions</td>
<td>923</td>
<td></td>
</tr>
<tr>
<td>ActionScript 2 and 3</td>
<td>924</td>
<td></td>
</tr>
<tr>
<td>ANSI C</td>
<td>925</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>925</td>
<td></td>
</tr>
<tr>
<td>C, Object Oriented Programming</td>
<td>926</td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>929</td>
<td></td>
</tr>
<tr>
<td>C++, Managed</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>C++/CLI Extensions</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>Delphi</td>
<td>932</td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>933</td>
<td></td>
</tr>
<tr>
<td>Java AspectJ Extensions</td>
<td>933</td>
<td></td>
</tr>
<tr>
<td>Object Oriented Programming in C</td>
<td>926</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>935</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>936</td>
<td></td>
</tr>
<tr>
<td>VB.Net</td>
<td>937</td>
<td></td>
</tr>
<tr>
<td>Visual Basic</td>
<td>939</td>
<td></td>
</tr>
<tr>
<td>Models Collection</td>
<td>1596</td>
<td></td>
</tr>
<tr>
<td>Model-View-Controller Pattern</td>
<td>1360</td>
<td></td>
</tr>
<tr>
<td>ModelWatcher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automation Interface Repository</td>
<td>1611</td>
<td></td>
</tr>
<tr>
<td>Modify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element Changes</td>
<td>839</td>
<td></td>
</tr>
<tr>
<td>Element Defects</td>
<td>839</td>
<td></td>
</tr>
<tr>
<td>Element Issues</td>
<td>839</td>
<td></td>
</tr>
<tr>
<td>Element Tasks</td>
<td>839</td>
<td></td>
</tr>
<tr>
<td>Linked Document Template</td>
<td>435</td>
<td></td>
</tr>
<tr>
<td>Project Task</td>
<td>847</td>
<td></td>
</tr>
<tr>
<td>Relationship Using Matrix</td>
<td>483</td>
<td></td>
</tr>
<tr>
<td>RTF Style Template (Legacy)</td>
<td>1179</td>
<td></td>
</tr>
<tr>
<td>Tagged Values</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>MOF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export To XMI</td>
<td>666</td>
<td></td>
</tr>
<tr>
<td>Getting Started</td>
<td>664</td>
<td></td>
</tr>
<tr>
<td>Import From XMI</td>
<td>666</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>661</td>
<td></td>
</tr>
<tr>
<td>Primitive</td>
<td>1353</td>
<td></td>
</tr>
<tr>
<td>More Windows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submenu</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attributes Between Elements</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>Connectors</td>
<td>444</td>
<td></td>
</tr>
<tr>
<td>Connectors, Quick Start</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Diagrams, Quick Start</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Elements Between Packages</td>
<td>358</td>
<td></td>
</tr>
<tr>
<td>Elements In Diagrams</td>
<td>357</td>
<td></td>
</tr>
<tr>
<td>Elements, Quick Start</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Internal Responsibility To External Requirement</td>
<td>470</td>
<td></td>
</tr>
<tr>
<td>Objects Between Packages</td>
<td>358</td>
<td></td>
</tr>
<tr>
<td>Operations Between Elements</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>Package Contents Up Or Down</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>Packages, Quick Start</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>Submenu</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>MS Jet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database As Model Repository</td>
<td>553</td>
<td></td>
</tr>
<tr>
<td>MS Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update RTF Report Links</td>
<td>1190</td>
<td></td>
</tr>
<tr>
<td>Use In RTF Documentation</td>
<td>1183</td>
<td></td>
</tr>
<tr>
<td>MS Word Tables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply Styles</td>
<td>1189</td>
<td></td>
</tr>
<tr>
<td>Manipulate</td>
<td>1189</td>
<td></td>
</tr>
<tr>
<td>Resize</td>
<td>1189</td>
<td></td>
</tr>
<tr>
<td>MSDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server Data Repository, Connect To</td>
<td>595</td>
<td></td>
</tr>
<tr>
<td>Server Repository, Create New</td>
<td>583</td>
<td></td>
</tr>
<tr>
<td>Upsize To</td>
<td>557</td>
<td></td>
</tr>
<tr>
<td>MTS File</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Options</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Incorporate Diagram Profile</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Incorporate Model Search</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Incorporate Model View</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Incorporate Task Pane Profile</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Incorporate Toolbox Profile</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Working With</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>Multi-page Diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print</td>
<td>335</td>
<td></td>
</tr>
<tr>
<td>Multiple Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Under One User ID</td>
<td>716</td>
<td></td>
</tr>
<tr>
<td>Multiple Select</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Items From Project Browser</td>
<td>311</td>
<td></td>
</tr>
<tr>
<td>Multiple Stereotype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restrict Application Of</td>
<td>1444</td>
<td></td>
</tr>
<tr>
<td>Multiplicity</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>Explanation</td>
<td>459</td>
<td></td>
</tr>
<tr>
<td>MVC Pattern</td>
<td>1360</td>
<td></td>
</tr>
<tr>
<td>MyISAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BaseModel Script</td>
<td>562</td>
<td></td>
</tr>
<tr>
<td>MySQL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Repository</td>
<td>574</td>
<td></td>
</tr>
<tr>
<td>Data Repository, Connect To</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>ODBC Driver, Set Up</td>
<td>564</td>
<td></td>
</tr>
<tr>
<td>Table Type, Set</td>
<td>1046</td>
<td></td>
</tr>
<tr>
<td>Upsize To</td>
<td>562</td>
<td></td>
</tr>
</tbody>
</table>
- N -

Name Template
- Foreign Key 1060
- Primary Key 1052

Namespace
- Clear 887
- Dialog 887
- Explanation 887
- List 887
- Locate In Project Browser 887
- Root 887
- Root Package Icon 76
- Set 887
- WSDL Element 1027

Naming Format
- Camel Case 1129
- Pascal Case 1129
- Spaced 1129
- Underscored 1129

N-Ary
- Association 1365
- Element 1365

Navigate
- Diagram 224

Navigation And Selection
- Hotkeys, For Diagram 304
- Navigation Compass 196

Nested Version Control Packages 670, 675

Nest
- Connector 1410
- Relationship 1410

New Code Sections
- Add To Existing Features 921

New Project
- Menu Option (File Menu) 87

New Search Query 189

New Structured Activity Dialog 1326

Node
- Element 1348

Non-Selectable
- Element 348

Notation
- Co-Region 1395
- Process Modeling 535

Note
- Add To Link/Connector 443
- Element 1317
- Element, Create 363
- For Attribute 206
- For Connector 206

For Diagram 206
For Element 206
For Operation 206
Insert New From Toolbar 162
Link To Element Feature 428
Link To Internal Documentation 428
Tab 206
Window 206

NoteLink
- Connector 1411
- Insert New From Toolbar 162
- Relationship 1411

Numbered Sections
- In RTF Reports 1161

Numbering Levels
- Apply 1161
- RTF Reports 1161
- Use 1161

Numbering List
- RTF Reports 1161

Numeric Range Generator
- Timeline Element States 1236

NUnit Transformation
- MDA-Style Transform 1111

- O -

Object
- Appearance, Options 238
- Attributes 1120
- Base Type, Set 424
- Classes 1120
- Classifiers 422
- Columns 1120
- Connector 417
- Create From Attribute 384
- Definition 1120
- Element 1348
- Elements And Connectors 136
- Group, Enterprise Architect UML Toolbox 136
- Instance 1348
- Links 417
- Move Between Packages 358
- Multiplicity 459
- Operations 1120
- Packages 1120
- Parameters 1120
- Properties 409, 1120
- Relationships 417
- Scenario 418
- State, Set 1350
Operation
- Protected, Icon 76
- Show On Diagram 307
- Tagged Values, Add 394
- WSDL Port Type Operation 1034

Operators
- On Constraints 1001

Options
- Compare Utility 750
- Element Visibility 240
- Project Views 172
- Reset For A Class 913
- Views Of Project 172
- Visual Styles 246

Options Dialog
- ActionScript 901
- Attribute/Operation Specifications 892
- C# 902
- C++ 903
- Communication Message Coloring 242
- Connector Settings 241
- Delphi 904
- Diagram Behavior 235
- Diagram Settings 234
- General Settings 231
- Introduction 230
- Java 908
- Links Settings 241
- Object Appearance 238
- Object Appearance, Default Fonts 239
- Object Appearance, Element Visibility 240
- PHP 908
- Python 909
- Sequence Diagram Options 237
- Standard Colors 232
- VB.Net 911
- Visual Basic 910
- XML Specifications 243

Oracle
- Package, Create 1051
- Sequence Options, DDL For Packages 1074
- Sequence Options, DDL For Table 1072
- Tables, Set Properties 1047
- Tables, Tagged Values 1047
- Temporary Table 1047
- Upsize To 559

Oracle 10g
- Data Repository, Connect To 589
- Server Repository, Create 578

Oracle 11g
- Data Repository, Connect To 589
- Server Repository, Create 578

Oracle 9i
- Data Repository, Connect To 589
- Server Repository, Create 578

Order
- Enterprise Architect 24
- Package Contents 72

Other Views
- Show/Hide From Toolbar 168
- Toolbar 168

Output Tab
- Debug Workbench Window 985

Output Window
- Context Menu 221

Override
- Default Toolbox 1466
- Implement Parent Operations 395
- Parent Operations 395
- Override Attribute Initializer 383

Package
- Add And Manage, Automation Interface Code Example 1667
- Add Element Directly 353
- Add To Diagram From Toolbox 289
- Add To Project Browser 289
- Add To UML Model, Quick Start 38
- Apply Version Control To Model Branch 701
- As RTF Bookmark 1184
- Automation Interface 1589
- Automation Interface Repository 1612
- Batch Export To XMI 650
- Batch Import From XMI 650
- Body, For Oracle 1051
- Check In 699
- Check Out 699
- Comparison, Example 751
- Configuration 647
- Configure For Version Control 697
- Context Menu, Project Browser 79
- Control, Menu 646
- Control, Remove 648
- Controlled 697
- Copy Between Projects 611
- Create Diagram 299
- Create Oracle Packages 1051
- CSV Import/Export Specification 654
- Delete In Project Browser 293
- Element 1350
- Existing, Drag Onto Diagram 291
- Export 646
Package
- Export Version Controlled Model Branch 702
- Hide Contents On Diagram 292
- Icon Overlays 76
- Import 646
- Import Version Controlled Model Branch 703
- Load 649
- Lock 727
- META-INF 1105
- Modeling With 287
- Move 44
- Move Element Between 358
- Move Objects Between 358
- Nested in Version Control 675
- Open From Diagram 307
- Open From Project Browser 288
- Profile 1431
- Rename 290
- Review Version Control History 704
- Save 649
- Save Profile 1442
- Show Contents On Diagram 292
- Specification, For Oracle 1051
- Synchronize Contents 886
- Tasks 287
- Update Contents 886
- Version Control 667
- Working With 287
- XML 646

Package Context Menu
- Add Submenu, Project Browser 80
- Build And Run Submenu, Project Browser 81
- Code Engineering Submenu, Project Browser 80
- Contents Submenu, Project Browser 81
- Documentation Submenu, Project Browser 80
- Import/Export Submenu, Project Browser 81

Package Control Submenu
- Model Node Context Menu, Project Browser 78

Package Diagram
- Description 1258
- Elements And Connectors 1258
- Example 1258

Package Import
- Connector 1415
- Relationship 1415

Package Merge
- Connector 1416
- Relationship 1416

Package Scenarios
- Import As Test Scenarios 831

Package Status
- Update 864

Page
- Footer, Print On Diagram 328
- Header, Print On Diagram 328
- Setup, Menu Option (File Menu) 87

Page Size
- Set For Diagram 336

Pan
- Diagram View 337

Pan And Zoom
- Diagram 224
- Window 224

Parameter
- Automation Interface, ElementFeatures Package 1642
- Kind Inot, By Reference 391
- Operation, By Reference 391
- Reference 391
- Show Details On Diagram 332

Parameter Kind
- Edit for Element Operation 403

Parameterized Classes (Templates) 1338

Parameters Dialog
- Operations 389

Parent
- Confirm Element As Parent 357
- Set For Element 354

Parse
- Source Code Files In Viewer 208

Part
- Add Property Value 1351
- Element 1351
- Represent On Sequence Diagram 1252

Partial Class
- Generate 884

Partition
- Activity 1318
- Element 1318
- Horizontal 1318
- Vertical 1318

Partitions Collection
- Automation Interface, ElementFeatures Package 1643

Pascal Case
- Naming Format 1129

Password
- Administrator Change 723
- Administrator Set 723
- Security, Change 711
- User Change 723
- Password Encryption
Password Encryption
Prior To Release 7.1 Of Enterprise Architect 721

Password Encryption (Repository)
At Release 7.1 Of Enterprise Architect 88, 92

Paste
Activity As Action 312
Activity As Link 312
Composite Element 312
Diagram Into Package 306
Element From Project Browser 310
Element On Diagram 304
Multiple Items As Children 311
Multiple Items As Instances 311
Multiple Items As Links 311
Multiple Items From Project Browser 311
Object As Child 310
Object As Link 310
Object As New Instance 310

Paste Elements
As Link 93
As New 93
From Clipboard As Metatile 93
Menu Option (Edit Menu) 93
Submenu (Edit Menu) 93

Patches
SQL, Run 606

Pattern
Action, Modify 512
Actions 508
Create From Diagram 508
Default, Change 512
Design 507
GoF 507
GoF, Download 511
Group of Four 507
Import Into Model 511
In Resources View 508
Model-View-Controller 1360
MVC 1360
Save 508
Save From Diagram 508

PDATA 1623

People
As Project Resources 771
Assign To Changes 845
Assign To Defects 845
Dialog 766
Settings 766

Permission List
User Security 718

PHP

Import, Reverse Engineering 872
Modeling Conventions 935
Options 908
Physical Model 66
PI Macro
Code Template Syntax 1521
PIM
Internal Bindings 1090
Pin
Connector End Point 165
Connector Start Point 165
Toolbox Pages 128
Pkg Import
Connection 1415
Relationship 1415
Pkg Merge
Connector 1416
Relationship 1416
Platform Naming Conventions 1129
Platform Specific Model 1090
Platform-Independent Model 1090
Port
Add To Element 1352
Element 1352
Inherited 1353
Redefined 1353
Represent On Sequence Diagram 1252
Port Type
WSDL Diagram 1031
WSDL Element 1031
Port Type Operation
WSDL 1034
Position Elements 107
Post
Add To Discussion Forum 254
Author 254
Edit In Discussion Forum 256
Reply To In Discussion Forum 255
PostgreSQL
Data Repository, Connect To 591
ODBC Driver, Set Up 566
Repository, Create 579
Upsize To 557
Post-New Events
Add-In Model 1552
EA_OnPostNewAttribute 1554
EA_OnPostNewConnector 1553
EA_OnPostNewDiagram (Object) 1553
EA_OnPostNewElement 1552
EA_OnPostNewMethod 1554
EA_OnPostNewPackage 1555
Pre/Post Conditions 1285
Predefined Reference Data
   Tagged Value  1502
   Tagged Value Type, Create  1503

Predefined Tag Type
   Assign To Stereotype  1433
   Define  1433

Predefined Tagged Value Type
   Arguments  1500
   Filters  1500

Pre-Deletion Events
   Add-In Model  1546
   EA_OnPreDeleteConnector  1546
   EA_OnPreDeleteDiagram  1547
   EA_OnPreDeleteElement  1546
   EA_OnPreDeletePackage  1547

Pre-New Events
   Add-In Model  1548
   EA_OnPreNewAttribute  1550
   EA_OnPreNewConnector  1549
   EA_OnPreNewDiagram (Object)  1549
   EA_OnPreNewElement  1548
   EA_OnPreNewMethod  1551
   EA_OnPreNewPackage  1551

Preprocessor Macros  897

Preserve Hierarchy
   CSV Specification  654

Primary Key
   Complex  1052
   Create  1052
   Description  1052
   Extended Properties  1054
   Name Template, Define  1052
   Simple  1052
   SQL Server, Non-Clustered  1054

Primitive
   Element  1353
   MOF  1353

Print
   Menu Option (File Menu)  87
   Multi-page Diagrams  335
   Page Footer On Diagram  328
   Page Header On Diagram  328
   Preview, Menu Option (File Menu)  87
   Project Issues  851
   Scaled Image  335
   Setup, Menu Option (File Menu)  87
   Task List  846

Print Preview
   Display  87
   Multiple Pages  87

Private Key
   Add  797

Problem Type
   Define  781

Procedure Class
   Create  1065

Process
   Element  535, 1366
   Element (Data Flow Diagram)  524
   Model Template  59
   Modeling  59

Process Modeling Notation  535

Professional Edition  9
   Upsize From  553

Profile
   Add Elements  1431
   Add Enumeration Elements  1438
   Add Metaclasses  1431
   Add Shape Script  1439
   Add Stereotypes  1431
   And Element Templates  369, 488, 1428
   Constraints  1436
   Create  1431
   Elements And Connectors  145
   Example File  496
   Export  1441
   Group, Enterprise Architect UML Toolbox  145
   Import  490
   Import From XML  488, 1428
   Package  1431
   References  494
   Save From Diagram  1442
   Save From Package  1442
   Set Default Appearance Of Stereotype Objects  1441
   Stereotype  1431, 1447
   Stereotypes  488, 1428
   Swimlanes Matrix  317
   Tags  1433
   Toolbox  1465
   Use  490
   Work With  1431
   Zachman  317

Profile Connector
   Add To Diagram  492

Progress OpenEdge
   Data Repository, Connect To  595
   ODBC Driver, Set Up  573
   Server Repository, Create  583
   Upsize To  555

Project
   Administration, Security Permissions  709
   Author  766
   Browser  70
Project

Clean 604
Clients 772
Compact .EAP File 615
Comparison, Why? 609
Configure 551
Data Integrity 603
Data, Transfer 608
EABase 52
Estimation 806
Explorer 70
File, EABase 551
Glossary 207, 857
Integrity Check 604
Integrity, Run SQL Patches 606
Issues 207, 849
Items, Add Via Toolbar 159
Menu 98
Metrics 806
Model Template 69
Open Existing 545
Recover 604
Remove Recent 57
Rename .EAP File 614
Repair Project .EAP File 616
Resources 771
Roles 769
Share On Network Drive 630
Share, DBMS Repository 629
Share, Network Drive 629
Share, Replication 629
Spell Checking 264
Statistics, View 98
Structure 285
Tasks 48, 207
Timescale Estimation 806
Toolbar 159
Transfer 562
Use The Model Search 184
View 70
What Is A? 547

Project Branch

Check In 699

Project Browser

Collapse Contents 81
Default Behaviour 73
Diagram Context Menu 84
Element Context Menu 83
Element Context Menu, Add Submenu 83
Exclamation Marks 729
Expand Contents 81
Free Sorting 73

Hide And Show 70
Icon Overlays 76
Introduction 70
Method Context Menu 85
Model Context Menu 77
Model Node, Package Control Submenu 78
Move Items Within 70
Open Package 288
Operation Context Menu 85
Order Package Contents 70
Package Context Menu 79
Package Context Menu, Add Submenu 80
Package Context, Build And Run Submenu 81
Package Context, Code Engineeriing Submenu 80
Package Context, Contents Submenu 81
Package Context, Documentation Submenu 80
Package Context, Import/Export Submenu 81
Reload Current Package 81
Reset Sort Order 81
Selective Collapse of Packages 70
Show Level Numbering 79
Show Stereotypes 73
Toolbar 75
Version Control Indicators 667
Views 70

Project Custom Colors
Get For Element 367
Set For Element 367

Project Discussion Forum
Icons 251
Introduction 251

Project Factor Calibration 806

Project File
Create 545
Example 549
Open 549

Project Glossary
Add Item, Glossary Detail Dialog 860
Add Item, Glossary Dialog 858
Delete Item, Glossary Detail Dialog 860
Delete Item, Glossary Dialog 858
Glossary Dialog 858
Glossary Report 862
Modify Item, Glossary Detail Dialog 860
Modify Item, Glossary Dialog 858
Project Glossary Tab 860

Project Indicators
Risk Types 822

Project Interface
Automation Interface 1657
Index

PropType Enum
    Automation Interface 1594
Proxy
    (Shortcut) File 88
Pseudo-State
    Elements 1220
    Fork 1307, 1309
    In State Machine Diagram 1220
    Join 1307, 1310
PSM 1090
Purchase
    Enterprise Architect 24
Python
    Import, Reverse Engineering 872
    MDG Technology For Enterprise Architect 12
    Modeling Conventions 936
    Options 909

- Q -

Qualified Association 1354
Qualifier
    Association Property 1354
Query Builder
    Search Definition 189
Query Methods
    In Shape Scripts 1489
Quick Add
    Tagged Values 421
Quick Linker 442
    Arrow 225
    Connector Names 1451
    Create Connector 228
    Create Element And Connector 226
    Default Settings, Hide 1451
    Definition Format 1448
    Element Names 1451
    Example 1450
    Introduction 225, 1448
    Options 226
Quick Start
    Add Connectors To UML Model 41
    Add Diagram To Package 39
    Add Diagram To UML Model 39
    Add Element To Diagram 40
    Add Element To UML Model 40
    Add Package to UML Model 38
    Auditing 733
    Create A UML Project In Enterprise Architect 35
    Define Connector Properties 42
    Define Element Properties 42
    Delete UML Model Components 46
    Generate HTML Report 1205
    Generate RTF Report 1135
    Guide To Enterprise Architect 29
    Move Project Components 44
    Project Tasks 48
    Save Project Changes 47

- R -

Rational Rose
    And XMI 636
    Export To 644
Rational Software Architect
    Models 642
Rational Software Modeler
    Import *.emx Files 640
    Import *.uml2 Files 640
Realization
    Connector, Quick Generation Of 473
    Implementation Link
    Implement Parent Operations 395
    Override Parent Operations 395
Realize
    An Information Flow 1392
    Connector 1417
    Relationship 1417
Realized Interfaces
    For Class, Show On Diagram 323
Receive
    Element 1319
    Event 1362
Record
    Automatic Debug Session For Thread 995
    Debug Session 992
    Manual Debug Session For Thread 994
    Thread, Automatic 992
    Thread, Manual 992
Record Arguments To Function Calls
    Debugger Sequence Tab Option 961
Record Calls To External Modules
    Debugger Sequence Tab Option 961
Recording History
    Tab In Debug Workbench Window 985
Recording Markers
    Set In Debug Session 995
Recover
    Controlled Package 651
    Project 604
Rectangle Notation
    Element Menu Option 343

© 1998-2008 Sparx Systems Pty Ltd
Relationship
   Mind Mapping 522
   Modify Using Relationship Matrix 483
   Nesting 1410
   Notelink 1411
   Object Flow 1412
   Occurrence 1414
   Package Import 1415
   Package Merge 1416
   Pkg Import 1415
   Pkg Merge 1416
   Realize 1417
   Recursion 1418
   Representation 1421
   Represents 1420
   Role Binding 1419
   Self Message 1398
   Show 417
   Trace 1422
   Transition 1423
   Usage 1425
   Use 1425
   Visibility 454
   Window, Context Menu 211

Relationship Matrix
   Access 478
   Access From Shortcut 88, 90
   Access From Shortcut (Direct Definition) 89
   Create Relationship 483
   Delete Relationship 483
   Export To CSV 484
   Introduction 476
   Link Direction 480
   Link Type 480
   Locate Elements 487
   Modify Relationship 483
   Open 478
   Options 482
   Print Matrix 485
   Print Preview 485
   Profiles, Save 486
   Review Elements 487
   Set Element Type 479
   Set Link Direction 480
   Set Link Type 480
   Set Source Package 481
   Set Target Package 481

Relationships Window
   Context Menu 314

Reload
   Diagram 705
   Model (Shared) 705
   Project 705
   View 705
   Reload Current Project
      Menu Option (File Menu) 87
   ReloadType Enumeration
      Automation Interface 1595
   Remove
      Package Control 648
      Recent Project 57
      Replication 634
   REMOVE_PREFIX 1129
   Rename
      Diagram 304
      Package 290
      Project .EAP File 614
      Views 618
   Re-Order
      Messages 1404
   Repair
      Project .EAP File 616
   Replace
      Linked Document 434

Replica
   Create 633
   Synchronize 633
   Upgrade 634
   Replication
      Change Collisions 632
      Disable 632
      Introduction 632
      Merge Rules 632
      Remove 634
      Using 632

Reply
   To Post In Discussion Forum 255

Report
   Dependency 1192
   HTML 1132, 1204
   Implementation 1193
   Open In Microsoft Word 1183
   Project Issues, Via Project Issues Dialog 854
   Project Issues, Via Project Issues Tab 855
   Rich Text Format 1132, 1133
   RTF 1132, 1133
   Testing Details 834

Report View
   Now Element List 174

Repository
   Adaptive Server Anywhere, Create 581
   Attributes 1596
   Author Collection 1607
   Automation Interface 1596
Repository
Client Collection 1607
Collection Class 1608
Connect To 553
Create 553
Datatype 1609
Encrypt Password 88, 92
EventProperties 1610
EventProperty 1611
Methods 1596
ModelWatcher 1611
MSDE Server, Create 583
Open, Automation Interface Code Example 1667
Package 1612
Package, Automation Interface 1595
Progress OpenEdge, Create 583
ProjectIssues 1615
ProjectResource 1616
PropertyType 1617
Reference 1617
Set Up Database 553
Stereotype 1618
Task 1619
Term 1620
Transfer Data Between 607
Use Extras, Automation Interface Code Example 1673
Representation
Connector 1421
Relationship 1421
Represents
Connector 1420
Relationship 1420
Requirement
Automation Interface, Element Package 1631
Convert From Responsibility 470
Element 466, 1366
Elements And Connectors 149
External 466
Group, Enterprise Architect UML Toolbox 149
Hide Stereotype Letter 466, 1366
Hierarchy 475
Inherited, Show 307
Internal 469
Properties 472
Show Stereotype Letter 466, 1366
Requirement Type
Define 778
Requirements
Analysis 60
Create 465
Element, Hide/Show Connectors 451
External 414
Functional 413
Generate Realization Connector 473
Implementation 473
In Rules & Scenarios Window 212
Internal 413
Linked 212
Management 60
Management, Overview 464
Model Template 60
Modeling 60
Non-Functional 413
Template 60
Requirements Diagram
Description 1272
Elements And Connectors 1272
Example 1272
Requirements Management
Enterprise Architect 6
Requirements Modeling
Enterprise Architect 6
Reserved Names
In Shape Scripts, Connectors 1492
In Shape Scripts, Elements 1492
Reset Options
For A Class 913
For All Classes 913
Source Code Language 75, 913
Reset Sort Order
In Project Browser 44
Resize
Element 360
Multiple Elements 360
Resolve Change Conflicts
Between Replicas 634
Resource
Allocation 815
And Tasking Details Dialog 819
Automation Interface, Element Package 1631
Report 819
Resource Document
RTF Generator (Enhanced) 1166
RTF, Batch Generate Reports 1166
Resource Management 814
Effort Types 820
Metric Types 821
Risk Types 822
Resources
Define 771
Documents 204
Favorites Folder 204, 205
Index 1770

Resources
- Matrix Profiles 204
- MDG Technologies 204
- Project Management Window 220
- Scripts 204
- Templates 204
- UML Patterns 204
- UML Profiles 204
- Window 204
- Window, And MDG Technologies 520
- XSL Stylesheets 204

Resources View
- Of Patterns 508

Responsibilities
- Internal 413

Responsibility
- Compartment, Element 426
- Move To External Requirement 470

Reverse Connector 453

Reverse Engineer
- Enterprise Architect 6
- Source Code 868

Reverse Engineering
- And Auditing 743
- And MDG Link 876
- Directory Structure 874
- Eclipse 876
- Handling Classes Not Found During Import 877
- Import ActionScript 872
- Import Binary Module 875
- Import C 872
- Import C# 872
- Import C++ 872
- Import Delphi 872
- Import Java 872
- Import PHP 872
- Import Python 872
- Import Source Code 870, 877
- Import Visual Basic 872
- Import Visual Basic.Net 872
- Initial Code In Operations 393
- Introduction 867
- ODBC Data Sources 1083
- Source Code, Import Directory Structure 874
- Synchronize Model And Code 878
- Visual Studio.NET 876

Reverse Synchronization
- Delete Attribute If Not In Code 892
- Delete Method If Not In Code 892
- Delete Model Aggregations For Attributes Not In Code 892
- Delete Model Associations For Attributes Not In Code 892
- Include Method Bodies In Model 892

Review
- Package Version Control History 704

Rich Text Format
- Copy Bookmark To Clipboard 80
- Document 1133
- Report 1133

Rich Text Format Generator
- Enhanced 1137
- Legacy 1173

Rich Text Format Report
- Apply Filter (Legacy) 1175
- Diagram Format (Legacy) 1176
- Diagram Only 1193
- Dialog (Legacy) 1173, 1174
- Element-Level 1135
- Exclude Elements (Legacy) 1176
- Exclude Objects (Legacy) 1176
- Exclude Package 1137
- Generate (Enhanced) 1135
- Generate (Legacy) 1173, 1179
- Generate From Element List 174
- Generate, Quick Start 1135
- Include Glossary (Legacy) 1177
- Include Issues (Legacy) 1177
- Include Package 1137
- Include Tasks (Legacy) 1177
- Object Selections (Legacy) 1178
- Options (Legacy) 1177
- Save As RTF Document (Enhanced) 1166
- Save As RTF Document (Legacy) 1181
- Set Main Properties (Legacy) 1175
- Single Element (Legacy) 1174
- Templates, Load (Legacy) 1179
- Through Element List 1135
- Through Model Search 1135
- Wizard (Legacy) 1174

Rich Text Notes
- Formatting 170
- In Legacy RTF Generator Reports 1173
- Keyboard Shortcuts 170
- Toolbar Options 170

Risk
- Automation Interface, Element Package 1632
- Management 817

Risk Types
- Define 822
- Global 822
- Non-Global 817

Risks
RTF Style Template Editor
Columns 1160
Commands 1148
Copy Text 1150
Cut And Paste 1150
Delete Text 1150
Description (Enhanced) 1141
Drawing Objects 1164
Edit Picture 1152
Export As RTF Document 1149
File Options 1149
Fonts 1153
Footers 1156
Frames 1164
Headers 1156
Highlight Text 1150
Hyperlinks 1157
Import RTF Document 1149
Move Text 1150
Page Breaks 1155
Page Columns 1160
Page Mode 1151
Paragraph Formatting 1154
Paragraph Styles 1154
Paste External Objects 1150
Picture Embed 1152
Picture Frame 1164
Picture Link 1152
Print Options 1149
Redo Edit 1148
Repagination 1155
Revert To Previous Copy 1149
Save File 1149
Search and Replace Options 1165
Sections 1160
Select Model Elements 1142
Select Text 1150
Show Headers & Footers 1151
Special Text 1160
Style Sheets 1160
Tab Support (Enhanced) 1155
Table Commands 1158
Table of Contents 1160
Tabular Sections 1145
Text Frame 1164
Text Scrolling 1148
Undo Edit 1148
View Options 1151
Zoom 1151

RTF Template
Import 1165
RTF Templates Tab (Enhanced) 1139

Rules & Scenarios
Window 212
Run
SQL Patches 606
Run Script
Create 950
Run State 1349
Add Instance Variable 1349
Define, Element Context Menu 343
Run-Time
Inspection 981
Variable, Define 1349
Run-Time State
Add Instance Variable 1349
Delete Instance Variable 1350
Introduction 1349

- S -

Save
Changes 47
Controlled Package 649
Diagram As UML Pattern 508
Diagram Image To Disk File 304
Diagram, Context Menu Option 297
Diagram, Quick Start 47
Package with XMI 649
Profile From Diagram Context 1442
Profile From Package Context 1442
RTF Report As RTF Document (Enhanced) 1166
RTF Report As RTF Document (Legacy) 1181
UML Pattern 508
Save As
Copy 88
Shortcut 88
Save Project As
Menu Option (File Menu) 87
Save Project As (File Menu Option)
Copy 89, 90
Shortcut (Capture Current Environment) 90
Shortcut (Direct Definition) 89
Scale
Image To Page Size 335
SCC
Providers Dialog 675
Version Control Options 675
Version Control, Upgrade For Enterprise Architect 4.5 678
Scenario
Automation Interface, Element Package 1633
Element 418
Index

Scenario
- Object 418
- Testing 830
- Type, Define 779
- Use Case 1248

Scenarios
- In Rules & Scenarios Window 212

Schema
- Database, Import From ODBC 1083
- DDL, Import From ODBC 1083
- Diagram 1275

Scope
- Values 923

Screen
- Element 1368

SDK
- Enterprise Architect 1427

Search
- A Package 184
- Add-In 1538
- Conditions 193
- Definitions 185
- Definitions, New 189
- Definitions, Predefined 191
- Element Features 185
- Fields 193
- Filters 185
- List, Predefined Searches 191
- Model 185
- Project 184, 185
- Project, Create Search Definition 189
- Query, New 189
- Results, Manipulate 181
- Simple 185
- The Model Search 184
- Within Project Browser 75

Search Data Parameter
- Add-In Search 1538

Search Filters
- Add To Search 192

Search Model
- Dialog 185

Search Project
- Add Filters 192

Section Numbering
- In RTF Reports 1161

Security
- Basics 708
- Change Password 711, 723
- Disable 709
- Enable 709
- Maintain Groups 716
- Maintain Users 711
- Policy 710
- Re-enable 709
- Reset Password 723
- Set Password 723
- Submenu (Project Menu) 101
- Tasks 708
- User Permission List 718
- What Is User Security? 708

Security Group Permissions 717

Select
- Alternative Image 320
- ODBC Data Source 1086
- Stereotypes 502
- Tables From ODBC Source 1087

Select All
- Menu Option (Edit Menu) 93

Select Attribute Type
- Dialog 376

Select By Type
- Menu Option (Edit Menu) 93

SELECT Statement
- SQL Search 189

Selectable
- Element 348

Self Message
- Calls 1399

Self-Message
- Connector 1398
- Hierarchy, Sequence Diagram 1250
- Relationship 1398
- Return 1398

Send
- Element 1320
- Event 1362

Sequence
- Communication Messages 1404
- Elements and Connectors 139
- Message, Change Timing Details 1399
- Message, Create 1395
- Message, Timing Details 1399
- Oracle, DDL Options 1072
- Oracle, DDL Options For Packages 1074

Sequence Diagram
- Activation Levels 1250
- Constraints On States, Debugger Generation Of Diagrams 1001
- Create After Recording Debug Session 996
- Create In Debug Session 992
- Description 1245
- Display Options 237
- Element Activation 1249
Index

Sequence Diagram
Elements 1248
Elements And Connectors 1245
Enable Capture Of State Transitions in Debug Session 999
Example 1245
Generate From Automatic Debug Record 995
Generate From Manual Debug Record 994
Generate From Method 992
Generate In Debug Session 991
Layout 1247
Lifeline Activation Level 1250
Messages, Self Message 1398
Self-Message Hierarchy 1250
Show Transitions In State, Overview 998
State Transition Options, Generate From Debug 996
Top Margin, Change 1252

Sequence Diagram Message
General Ordering 1401

Sequence Diagram Messages
Examples 1402
External To Sequence 1402

Sequence Element
Inline, Part And Port 1252

Sequence Message
Label Visibility 1251
Modify Height 1247

Sequence Tab
Build Script Options 959
Debugger Options 959

Server
Apache Tomcat, Debugging 970
JBoss, Debugging 970
Tomcat, Debugging 970

Server Configuration
JBoss 973
Tomcat 974

Server Repository
Create for Oracle 10g 578
Create for Oracle 9i 578

Service
WSDL Diagram 1030
WSDL Element 1030

Service Configuration
Tomcat 975

Service Oriented Architecture 1008
Development 34
Implementing XML-Based, In Enterprise Architect 34

Set
Association Specialization 453

Collection Classes 899
Connector Visibility 454
Default Diagram, Model 332
Default Tree Behavior 73
Diagram Appearance Options 325
Diagram Page Size 336
Diagram Properties 325
Element Classifier 512
Element Cross-References 356
Element Custom References 356
Element Parent 354
Feature Visibility 307
Font For Element Text 349
Group Permissions 717
Instance Classifier 424
Interface For Element 354
Main RTF Report Properties (Legacy) 1175
Message Source and Target 463
Object Base Type 424
Object State 1350
Parent For Element 354
Relationship Visibility 454

Set Attribute Dialog 1283
Set Feature Dialog 1283
Set Feature Visibility
Element Context Menu Option 348

Set Function
Create As Attribute Property 381

Set Operation Dialog 1283
Set Project Custom Colors
For Element 367

Set Up
Adaptive Server Anywhere ODBC Driver 569
Database Model Files 553
Debug Session 951, 954
For .NET 954
MySQL ODBC Driver 564
ODBC Driver 563
PostgreSQL ODBC Driver 566
Progress OpenEdge ODBC Driver 573
Single Permissions 714
User Groups 713

Settings
Author 766
Cardinality 787
Clients 772
Constraint Status Types 777
Constraint Types 776
Default Hours 813
Effort Types 820
Environment Complexity Factors 809
Estimation 784

© 1998-2008 Sparx Systems Pty Ltd
Index

Settings
  General Types  774
  Maintenance  781
  Menu  765
  Menu, Configure  123
  Metric Types  821
  Metrics  784
  People  766, 771
  Problem Types  781
  Project Author  766
  Project Resources  771
  Requirement Types  778
  Risk Types  822
  Roles  769
  Scenario Types  779
  Status Types  774
  Stereotypes  785
  Tagged Value Types  786
  Technical Complexity Factors  807
  Template Package  369
  Testing Types  782
  UML Types  785

Shallow Copy
  Of Diagram  306

Shape Attributes
  Of Shape Scripts  1484

Shape Editor  786, 1497

Shape Scripts
  Add To Profile  1439
  Arithmetical Operations  1492
  Assign To Stereotype  1481
  Basic Shapes  1494
  Change Font Of Text  1492
  Cloud Path  1494
  Color Queries  1488
  Comments  1492
  Conditional Branching  1489
  Connector  1494
  Create  1481
  Custom Shapes  1480
  Display Element Properties  1489
  Double Line  1494
  Drawing Methods  1485
  Editable Field  1494
  Example Shape Scripts  1494
  Filled Arrow  1494
  Fonts  1492
  Getting Started  1481
  Introduction  1480
  Looping  1492
  Miscellaneous  1492
  Multiple Condition  1494
  Override Element Appearance  1481
  Properties, Connector  1489
  Properties, Element  1489
  Query Methods  1489
  Reserved Names, Connectors  1492
  Reserved Names, Elements  1492
  Return Statement  1494
  Shape Attributes  1484
  Shape Editor  786, 1497
  Single Condition  1494
  Stereotypes  1480
  String Manipulation  1492
  Subshape  1494
  Subshape Layout  1491
  Syntax Grammar  1483
  Terminate Execution  1492
  Variable Declarations  1492
  Without Stereotypes  1492
  Writing Scripts  1483

Share
  An Enterprise Architect Project  629
  Project On Network Drive  630

Shared Key
  Add  797
  Issues  799

Shared Model Development
  Enterprise Architect  6

Shortcut
  Clear  90
  Diagram (Capture Current Environment)  90
  Discussion Forum (Capture Current Environment)  90
  Menu, Enterprise Architect UML Toolbox  131
  Model (Capture Current Environment)  90
  Model Search (Capture Current Environment)  90
  Relationship Matrix (Capture Current Environment)  90

Shortcut To
  Diagram  88
  Diagram (Direct Definition)  89
  Discussion Forum  88
  Discussion Forum (Direct Definition)  89
  Model  88
  Model (Direct Definition)  89
  Model Search  88
  Model Search (Direct Definition)  89
  Relationship Matrix  88
  Relationship Matrix (Direct Definition)  89

Show
  Code Execution (Project Menu)  99
  Connectors, All Diagrams  451
Spell Check
Single Package 265
Spell Checking
Introduction 264
Spelling
Language (Tools Menu) 109
Spelling Language
Select 268
SQL
Custom Searches 189
Editor 189
Patches, Run 606
Search, SELECT Statements 189
Server Data Repository, Connect To 587
Server Repository, Create 576
SQL Server
Desktop Engine, Upsize To 557
Non-Clustered Primary Key 1054
Upsize To 560
Stack
Debug Workbench Window 984
Standard Colors
Attribute 232
Connector Line 232
Diagram Backgound 232
Element Fill 232
Element Line 232
Method 232
Note Text 232
Operation 232
Options 232
Screen 232
Shadow 232
Standard Element Stereotypes 504
Start
Application 52
Enterprise Architect 52
Start Page 55
Quick Start 35
State
Add To State Lifeline Element 1231
Chart 1216
Composite 1321, 1322
Constraints, Debugger Generation Of Sequence Diagrams 1001
Delete On State Lifeline Element 1231
Diagram 1216
Edit On State Lifeline Element 1231
Element 1321
Entry And Exit Actions 386, 1321
In Timeline Element 1236
Locate In State Machine Diagram 1227
Locate In State Machine Table 1227
Reposition In State Machine Table 1227
Simple 1321
State Machine Table Conventions 1227
State (Machine)
Elements and Connectors 141
Group, Enterprise Architect UML Toolbox 141
State Invariant
Element 1324, 1326
State Lifeline
Element 1323
State Lifeline Element
Add State 1231
Add To Timing Diagram 1231
Add Transition 1232
Change Transition Time 1232
Define Name 1231
Delete State 1231
Delete Transition 1232
Edit State 1231
Edit Transition 1232
Merge Transitions 1232
Move Transition 1232
Set Timeline Start Position 1231
Sizing and Scale 1231
Synchronize Transition 1232
State Machine
Create For Sequence Diagram Generation In Debugger 1000
Element 1328
Entry And Exit Actions 386
Regions 1219
State Machine Diagram
Description 1216
Display Format 1216
Elements And Connectors 1216
Example 1216
Locate State In State Machine Table 1227
Locate Transition In State Machine Table 1227
Locate Trigger In State Machine Table 1227
State Machine Table
Add States 1225
Add Substates 1225
Add Triggers 1226
Cell Color 1222
Cell Enumeration 1222
Cell Highlights 1222
Cell Size 1222
Change Position In Diagram View 1225
Change Size 1225
ChangeTransitions 1226
Conventions 1227
Index

State Machine Table
Description 1220
Export To CSV 1228
Format 1220
Insert Transitions 1226
Legend, Add 1227
Legend, Remove 1227
Locate State In State Machine Diagram 1227
Locate Transition In State Machine Diagram 1227
Locate Trigger In State Machine Diagram 1227
Operations, Overview 1224
Options 1222
Remove Substate Parent Relation 1225
Reposition States 1227
Reposition Substates 1227
Reposition Triggers 1227
State-Next State 1220
State-Trigger 1220
Table Format 1222
Trigger-State 1220
State Region
Composite 1219
State Transitions
Enable Capture In Sequence Diagram, Debug Session 999
Recorded In Debugger, Overview 998
State/Continuation Element 1324
Status Bar
Enterprise Architect Workspace 169
Hide 169
Show 169
Status Type
Color 774
Define 774
For Different Elements 774
Stereotype
Add Shape Script In Profile 1439
Add To Profile 1431
Add, Automation Interface Code Example 1674
Analysis 1357
And Metatiles 499
Apply To Dependency Relationship 1385
Automation Interface Repository 1618
Custom 1429
Define As Metatype 1443
Definition 499
Dialog 1429
Extension 1357
Inbuilt 1357
Multiple, Restrict Application Of 1444
Predefined Tag Types 1433
Profile 1447
Selector 502
Set Default Appearance Of Objects In Profile 1441
Settings 785
Show On Project Browser 73
Standard Element 504
Tagged Values In Profile 1433
Tags For Supported Attributes 1434
Tags, Define 1433
UML Description 499
View 1067
Visibility 503
With Alternative Images 506
XSD In UML Profile 1011
Stereotyped Element Table 1369
Store
Image In Enterprise Architect 320
Stored Procedure
As Individual Class 1065
As Operation Of Container Class 1062
Create Class As 1062
Define Procedure Stereotype 1065
Define Stereotype 1062
Definition 1061
Operation Parameters 1062
Supported Databases 1061
Straighten
Line At Cursor 446
Structural Diagrams
Overview 1258
Structured Activity
Conditional Node 1326
Element 1326
Loop Node 1326
Nested 1326
Node 1326
Sequential Node 1326
Simple Composite 1326
Structured Tags
Create 1499
Style
For Connectors 441
Sub-Activity Element 1286, 1326
Submachine State Element 1321, 1328
Submenu
Add-In 122
Index

Synch
   Element 1329
Synchronization 632
   Initial Code In Operations 393
   Introduction 867
   Macros, Code Template Syntax 1521
   Of Source Code And Model 868
Synchronize
   Batch With Code 106
   Class With Code 106
   Code 921
   Existing Code Sections 921
   Package With Source, (Project Menu Option) 99
   Replicas 633
   UML Profile Tags and Constraints 492
Syntax Check
   For UML, Option 234
Syntax Grammar
   Of Shape Scripts 1483
SysML
   MDG Technology For, Enterprise Architect 12
System
   Project Glossary 207
   Project Issues 207
   Project Tasks 207
   Tabs 207
   Testing 828
   Users 711
   Window 207
System Boundary
   Element 1329
   Insert New From Toolbar 162
System Requirements
   For Debug 964
System Window
   Add Item, Glossary Detail Dialog 860
   Delete Item, Glossary Detail Dialog 860
   Modify Item, Glossary Detail Dialog 860
   Project Glossary Tab 860
   Project Issues Tab 851
   Project Tasks Tab 846
Systems Modeling Languages
   MDG Technology For, Enterprise Architect 12
   RTF Style Template Editor (Enhanced) 1155
Tabbed Frame
   Combine Windows In 196
   Remove Window From 196
Table
   DDL Script For 1039
   Detail 1369
   Element 1369
   Owner Tagged Value 1045
   Properties 1369
   Set Owner 1045
   Set Properties 1043
   State Machine 1220
Table Of Contents
   MS Word, Add 1186
Table Of Figures
   MS Word, Add 1187
Tag
   Compartment, Element 426
   Filters 1500
   Management, Advanced 420
   Profile 1433
Tag Type
   Masked 1500
   Predefined, Assign To Stereotype 1433
Tagged Value
   Add 421
   Add To Operations 394
   Assign Information To 217
   Assign To Item 215
   Connector, Use 1435
   Duplicate Values 214
   Element 419
   Filters 1500
   For Oracle Table Properties 1047
   For Table Owner 1045
   Fully Qualified, Show 307
   Fully-Qualified Value 214
   In UML Profiles 491
   Inherited, Show 307
   Macros, Code Template Syntax 1518
   Model Components And 214
   Modify 214
   Modify Value 215
   Of Attributes 379
   Operation Parameters 390
   Predefined Reference Data 1502
   Quick Add 421
   Show Duplicates 218
   Supported For UML Profile Stereotypes 494
   Toolbar 214
   Types 786

© 1998-2008 Sparx Systems Pty Ltd
Tagged Value
  Types Of Value Field 215
  View 214
  Window 214
Tagged Value Type
  Introduction 1498
  Predefined 1500
  Predefined Reference Data, Create 1503
Tagged Value Type
  Automation Interface, Element Package 1633
Target
  Set For Message 463
  Types 1195
Target Role 461
Task
  Automation Interface Repository 1619
  Completion 815
  Details 847
Task Pane Profile
  Define In From MTS File 1464
Tasks Pane 13
  Allocate Toolbox To Contexts 1475
  Commands, Built In 1473
  Contexts, Define 1474
  Getting Started 222
  Incorporate In MDG Technology 1475
  Named Contexts 1474
  Profiles, Create 1472
  Run Add-In Functions From 1473
  Save Profile 1475
  Toolboxes, Define 1472
  Tools 222
  Window 222
TCF
  Value 807
  Weighting 807
TcSE
  MDG Integration For, Enterprise Architect 12
Team Deployment
  And Version Control 671
  Version Control Branching 671
Team Development 628
Team Foundation Server
  Version Control Options 691
Teamcenter Systems Engineering, Siemens PLM
  MDG Integration For, Enterprise Architect 12
Tear Off Menus 200
Technical Complexity Factor
  Estimate Project Size 811
  Value 807
  Weighting 807
Technology Developer
  And Enterprise Architect 281
  Project Role 281
Technology Events
  Add-In Model 1555
  EA_OnDeleteTechnology 1556
  EA_OnImportTechnology 1556
  EA_OnPreDeleteTechnology 1555
Template
  Editor In SDK 1527
  Editor, Code Templates 919
  Element 488
  Model, Incorporate In Technology 1478
  Package, Settings 369
  Parameterized Classes 1338
  RTF, Import 1165
Templates
  Transformation, Default 1118
Term
  Automation Interface Repository 1620
Terminate
  Element 1330
Test
  Automation Interface, Element Package 1634
  Cases 826
  Documentation 836
  Import From Other Element 833
  Model Template 67
  Report 836
  Result Output 836
  Script Output 836
Test Case
  Element 1369
Test Details Dialog 825
Test Scripts
  Compartment 835
Tester
  And Enterprise Architect 279
  Project Role 279
Testing
  Acceptance 829
  Compartment, Element 426
  Import Element Scenarios 831
  Import Package Scenarios 831
  Integration 827
  MDG Technology For, Enterprise Architect 12
  Model Template 67
  Overview 823
  Report, Create 1195
  Scenario 830
  Support 823
  System 828
  Type, Define 782
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>826</td>
</tr>
<tr>
<td>Testing Details Report</td>
<td>834</td>
</tr>
<tr>
<td>Testing Feature</td>
<td>407</td>
</tr>
<tr>
<td>Text</td>
<td>363</td>
</tr>
<tr>
<td>Text Element</td>
<td>162</td>
</tr>
<tr>
<td>TFS</td>
<td>691</td>
</tr>
<tr>
<td>The Open Group Architecture Framework</td>
<td>12</td>
</tr>
<tr>
<td>Tidy Line Angles</td>
<td>446</td>
</tr>
<tr>
<td>Time Event</td>
<td>1319</td>
</tr>
<tr>
<td>Time Interval</td>
<td></td>
</tr>
<tr>
<td>Compress</td>
<td>1239</td>
</tr>
<tr>
<td>Context Menu</td>
<td>1239</td>
</tr>
<tr>
<td>Copy and Paste</td>
<td>1242</td>
</tr>
<tr>
<td>Create</td>
<td>1239</td>
</tr>
<tr>
<td>Delete</td>
<td>1239</td>
</tr>
<tr>
<td>Description</td>
<td>1239</td>
</tr>
<tr>
<td>Move</td>
<td>1239</td>
</tr>
<tr>
<td>Operations</td>
<td>1242</td>
</tr>
<tr>
<td>Resize</td>
<td>1239</td>
</tr>
<tr>
<td>Select</td>
<td>1239</td>
</tr>
<tr>
<td>Shift Left Or Right</td>
<td>1242</td>
</tr>
<tr>
<td>Transitions</td>
<td>1242</td>
</tr>
<tr>
<td>Time Range</td>
<td></td>
</tr>
<tr>
<td>Set For Timing Diagram</td>
<td>1230</td>
</tr>
<tr>
<td>Timeline Element States</td>
<td></td>
</tr>
<tr>
<td>Add Via Configure Timeline Dialog</td>
<td>1236</td>
</tr>
<tr>
<td>Delete via Configure Timeline Dialog</td>
<td>1236</td>
</tr>
<tr>
<td>Edit Via Configure Timeline Dialog</td>
<td>1236</td>
</tr>
<tr>
<td>Maintain</td>
<td>1236</td>
</tr>
<tr>
<td>Numeric Range Generator</td>
<td>1236</td>
</tr>
<tr>
<td>Timeline Range</td>
<td></td>
</tr>
<tr>
<td>Set For Timing Diagram</td>
<td>1230</td>
</tr>
<tr>
<td>Timeline Start Position</td>
<td></td>
</tr>
<tr>
<td>Set For State Lifeline Element</td>
<td>1231</td>
</tr>
<tr>
<td>Timing</td>
<td></td>
</tr>
<tr>
<td>Constraint</td>
<td>1399</td>
</tr>
<tr>
<td>Details, Change</td>
<td>1399</td>
</tr>
<tr>
<td>Elements and Connectors</td>
<td>140</td>
</tr>
<tr>
<td>Group, Enterprise Architect UML Toolbox</td>
<td>140</td>
</tr>
<tr>
<td>Message, Create</td>
<td>1407</td>
</tr>
<tr>
<td>Observation</td>
<td>1399</td>
</tr>
<tr>
<td>Timing Diagram</td>
<td></td>
</tr>
<tr>
<td>Add Value Lifeline Element</td>
<td>1234</td>
</tr>
<tr>
<td>Create</td>
<td>1230</td>
</tr>
<tr>
<td>Description</td>
<td>1228</td>
</tr>
<tr>
<td>Edit Options</td>
<td>1230</td>
</tr>
<tr>
<td>Edit Value Lifeline Element</td>
<td>1234</td>
</tr>
<tr>
<td>Elements And Connectors</td>
<td>1228</td>
</tr>
<tr>
<td>Example</td>
<td>1228</td>
</tr>
<tr>
<td>Set Time Range</td>
<td>1230</td>
</tr>
<tr>
<td>TOGAF</td>
<td></td>
</tr>
<tr>
<td>MDG Technology For, Enterprise Architect</td>
<td>12</td>
</tr>
<tr>
<td>Tomcat</td>
<td></td>
</tr>
<tr>
<td>Server, Configuration</td>
<td>974</td>
</tr>
<tr>
<td>Server, Debugging</td>
<td>970</td>
</tr>
<tr>
<td>Service Configuration</td>
<td>975</td>
</tr>
<tr>
<td>Toolbar</td>
<td></td>
</tr>
<tr>
<td>Add Commands</td>
<td>111</td>
</tr>
<tr>
<td>Change Command Icon Appearance</td>
<td>111</td>
</tr>
<tr>
<td>Code Generation</td>
<td>160</td>
</tr>
<tr>
<td>Create</td>
<td>111</td>
</tr>
<tr>
<td>Current Connector</td>
<td>165</td>
</tr>
<tr>
<td>Current Element</td>
<td>164</td>
</tr>
<tr>
<td>Customize</td>
<td>111</td>
</tr>
<tr>
<td>Debug</td>
<td>979</td>
</tr>
<tr>
<td>Default Tools</td>
<td>158</td>
</tr>
<tr>
<td>Diagram</td>
<td>163</td>
</tr>
<tr>
<td>Display Labels</td>
<td>111</td>
</tr>
<tr>
<td>Docked</td>
<td>156</td>
</tr>
<tr>
<td>Element</td>
<td>162</td>
</tr>
<tr>
<td>Format (Element Appearance)</td>
<td>166</td>
</tr>
<tr>
<td>Hide</td>
<td>111</td>
</tr>
<tr>
<td>Large Icons</td>
<td>121</td>
</tr>
<tr>
<td>Options, Customize Appearance</td>
<td>121</td>
</tr>
<tr>
<td>Other Views</td>
<td>168</td>
</tr>
<tr>
<td>Project</td>
<td>159</td>
</tr>
<tr>
<td>Project Browser</td>
<td>75</td>
</tr>
<tr>
<td>Remove</td>
<td>111</td>
</tr>
<tr>
<td>Remove Commands</td>
<td>111</td>
</tr>
<tr>
<td>Rename</td>
<td>111</td>
</tr>
<tr>
<td>Rich Text Notes</td>
<td>170</td>
</tr>
<tr>
<td>Screen Tips</td>
<td>121</td>
</tr>
<tr>
<td>Show</td>
<td>111</td>
</tr>
<tr>
<td>Submenu</td>
<td>96</td>
</tr>
<tr>
<td>UML Elements</td>
<td>162</td>
</tr>
<tr>
<td>Workspace</td>
<td>156</td>
</tr>
<tr>
<td>Workspace Views</td>
<td>167</td>
</tr>
<tr>
<td>Toolbox</td>
<td></td>
</tr>
<tr>
<td>Customize</td>
<td>1465</td>
</tr>
<tr>
<td>Connectors For Extending</td>
<td>1468</td>
</tr>
<tr>
<td>Elements For Extending</td>
<td>1467</td>
</tr>
<tr>
<td>Items, Assign Icons For</td>
<td>1467</td>
</tr>
<tr>
<td>Override Default</td>
<td>1466</td>
</tr>
</tbody>
</table>
Index

Trigger
Create In Transition Properties 1423
Element 1330
For Transition 1423
Locate In State Machine Diagram 1227
Locate In State Machine Table 1227
Operation, What Is A? 1070
Reposition In State Machine Table 1227
State Machine Table Conventions 1227

Trigger Operation
Create 1070
What Is A? 1070

Type
Element Context Menu 343
Type Hierarchy Dialog 354
Type Specific Menu Section Connector 440

- U -

UI Control
Element 1369
Element, Create 1370

UML
1.3 604, 636, 638, 640
1.4 638, 640
1.5 238
2.0 640
2.0 Migration 604
2.1 - Definition 6
Analysis Tool - Enterprise Architect 5
Build Systems 6
Connectors 1373
Data Modeling Profile 1039
Definition 1210
Design Systems 6
Design Tool - Enterprise Architect 5
Dictionary 1210
DTD 645
Elements 1278
Enterprise Architect Modeling Platform 285
Extend 1210
Manage Complexity 6
Mappings To XSD 1019
Model Complexity 6
Model Structure, Manage - Enterprise Architect 6
Modeling Within, Enterprise Architect Overview 32
Models Under Single Root 642
Pattern 507
Recommended Reading 1210

Shared Model Development - Enterprise Architect 6
Support - Enterprise Architect 6
Syntax Checking Option 234
Visualize Systems - Enterprise Architect 6

UML Behavioral Diagram
Overview 1213

UML Business Process Model
In Enterprise Architect 59

UML Class Model
In Enterprise Architect 63

UML Component Model
In Enterprise Architect 65

UML Database Model
In Enterprise Architect 64

UML Deployment Model
In Enterprise Architect 66

UML Diagram
Add To Project 299
Copy, Deep 306
Copy, Shallow 306
Create 299
Duplicate 306
Extended 1212, 1269
In Enterprise Architect 294
MDG Technology 1212
Overview 1212
Paste 306
Types 1212
What Is A? 1212

UML Domain Model
In Enterprise Architect 62

UML Element
Behavioral Diagram Elements 1279
Structural Diagram Elements 1335
Toolbar 162

UML Maintenance Model
In Enterprise Architect 68

UML Model
Add Connectors In Enterprise Architect, Tutorial 41
Add Diagram In Enterprise Architect, Tutorial 39
Add Element In Enterprise Architect, Tutorial 40
Add Element To Diagram, Tutorial 40
Add Package In Enterprise Architect, Tutorial 38
Add View, Tutorial 37
CSV Import And Export In Enterprise Architect 653
Define Connector Properties In Enterprise Architect 42
UML Model
  Define Element Properties In Enterprise Architect 42
  Manage 542
  Project Roles And Tasks In Enterprise Architect 48
  What Is A? 542
UML Model Management
  Add-Ins 542
  Auditing 542
  Automation Interface 542
  Baselines And Differences 542
  In Enterprise Architect 542
  Introduction 542
  Model Files 542
  Project Data Transfer 542
  Project Discussion Forum 542
  Project Sharing 542
  Replication 542
  User Security 542
  Version Control 542
UML Model Pattern
  Introduction 58
  Select In Enterprise Architect 35
UML Model Template
  In Enterprise Architect 58
  Introduction 58
UML Modeling
  And MDG Technologies 285
  And Requirements Management In Enterprise Architect 285
  Build Models 285
  Business Modeling In Enterprise Architect 285
  Enterprise Architect Processes 30
  Enterprise Architect Tools and Features 51
  Enterprise Architect, Tutorial 29
  Quick Start - Enterprise Architect 29
  Tasks With Enterprise Architect 30
  Tutorial - Enterprise Architect 29
  What Is? 285
  With Diagrams 294
  With Elements 338
  With Enterprise Architect 285
  With Packages 287
  With UML Patterns In Enterprise Architect 285
  With UML Profiles In Enterprise Architect 285
  With UML Stereotypes In Enterprise Architect 285
  Working With Enterprise Architect 29
UML Modeling Tool
  Enterprise Architect 3
  Enterprise Architect - Key Features 8
UML Pattern
  Actions 508
  Add To Diagram 512
  Create From Diagram 508
  In Resources View 508
  Save 508
  Save From Diagram 508
UML Profile
  And Element Templates 488, 1428
  Create 1431
  Example File 496
  Export 1441
  Import 490
  Import From XML 488, 1428
  Save From Diagram 1442
  Save From Package 1442
  Stereotypes 488, 1428
  Structure 495
  Synchronize Tagged Values and Constraints 492
  Work With 1431
  XSD, Stereotypes 1011
UML Profile Connector
  Add To Diagram 492
UML Project
  Create In Enterprise Architect, Tutorial 35
UML Project Model
  In Enterprise Architect 69
UML Requirements Model
  In Enterprise Architect 60
UML Resources
  Patterns 507, 511
UML Structural Diagram
  Overview 1258
UML Syntax Compliance
  Turn Off 622
UML Testing Model
  In Enterprise Architect 67
UML Toolbox
  Activity Group 142
  Analysis Group 147
  Appearance Options 128
  BPMN Group 529
  Business Modeling Group 1276
  Class Group 135
  Collapse Page 126
  Common Group 133
  Communication Group 138
  Component Group 143
  Composite Group 137
  Create Elements And Connectors 126
  Custom Group 148
  Data Flow Diagram Group 524
Index

Use Case Diagram
- Description 1215
- Elements And Connectors 1215
- Example 1215

Use Element Extras
- Automation Interface Code Example 1671

Use Repository Extras
- Automation Interface Code Example 1673

User
- Default Fonts 239
- Directory 766
- Groups 711
- Lock, Apply 728
- Settings 896

User Dictionary
- For Spell Checker 267

User Forum 15

User Interface
- Components 53
- Control Element 1369
- Element 1369
- Elements and Connectors 151
- Group, Enterprise Architect UML Toolbox 151
- Screen Prototype 1368

User Interface Design 1270

User Interface Diagram
- Description 1274
- Elements And Connectors 1274
- Example 1274

User Lock
- Identify Owner 730
- Indicators 729

User Security
- Basics 708
- Disable 709
- Enable 709
- Maintain Groups 716
- Maintain Users 711
- Policy 710
- Re-enable 709
- Tasks 708
- What Is User Security? 708

User Security Groups
- Set Up 713

User Settings
- Options Dialog 229

Using Enterprise Architect
- Application Workspace 53
- Remove Recent Project 57
- Start Page 55

Utility
- Compare 745

Diff 745

UUCP
- Estimate Project Size 811

- V -

Validation
- Cancel 101
- Of Diagram 620
- Of Element 620
- Of Model, Configure 622
- Of Model, Configure For MDG Technology 1477
- Of Package 620
- Rules, Model 623

Validation, Properties
- Attribute 624
- Element 624
- Feature 624
- Relationship 624

Validation, Well Formedness
- Attribute 623
- Diagram 623
- Element 623
- Feature 623
- Relationship 623

Value Lifeline Element 1333
- Add States 1234
- Add To Timing Diagram 1234
- Add Transitions 1234
- Change Transition Time 1234
- Delete Transitions 1234
- Edit Transitions 1234
- Sizing and Scale 1234
- States 1234
- Transitions 1234

Variable
- Debug 986
- Definitions, Code Template Syntax 1524
- Definitions, Examples 1524
- References, Code Template Syntax 1524
- References, Examples 1524

VB
- Set Up In Automation Interface 1585

VB.Net
- Modeling Conventions 937
- Options 911

Version Control 646
- Apply To Enterprise Architect Model 670
- Apply To Model Branch 701
- Basics 669
- Branching 671
Version Control 646
Check In and Check Out Packages 699
Configuration 673, 703
Configuration, Add Previously-Defined 698
Configuration, Team Deployment 671
Configure In Subversion 689
Copy-Move-Modify Policy 669
CVS Options 679, 683
CVS Remote Repository 679
Discussion Of File Control 670
Export Model Branch 702
File History 695
File Properties 695
Import Model Branch 703
In Team Deployment 671
Include Other Users’ Packages 700
Introduction 667
Locking – Necessary? 669
Lock-Modify-Unlock Policy 669
Manual, With XMI 651
Menu 695
Nested Packages 670
Offline 706
Policies 669
Project Browser Indicators 667
Refresh View Of Shared Model 705
Review Version Control History 704
SCC Options 675
SCC, Providers DialogSCC, Providers Dialog 675
SCC, Upgrade For Enterprise Architect 4.5 678
Set Up 673
Settings 673
Setup Menu 672
Submenu (Project Menu) 102
Subversion Options 686
Subversion, Create Local Working Copy 688
Subversion, Create New Repository Subtree 687
Subversion, Setting Up 686
Subversion, TortoiseSVN 691
Subversion, Using With Enterprise Architect Under WINE Crossover 688
TFS Options 691
Use Nested Version Control Packages 675
Using 695
Who Has Checked Out A Package? 695
Version Controlled Package
Icon 76
View
Add To UML Model, Quick Start 37
Class 1067
Create 1067
Database 1067
Element List 174
Locks 720
Menu 95
Next Diagram 325
Of Model 177
Previous Diagram 325
Project Statistics 98
Stereotype 1067
Submenus 96
View Options 172
Diagram View 173
Element List 174
Views
Add 617
Class 617
Component 617
Delete 619
Deployment 617
Dynamic 617
Manage 617
Rename 618
Simple 617
Use Case 617
Virtual Document
Add Packages As Attributes 1200
Change Package Sequence 1201
Create Master Document For 1197
Create Model Document For 1199
Delete Package Attributes 1200
Delete Packages 1200
Document Order 1201
Generate RTF Report From 1202
Introduction 1196
Master Document Element 1196
Model Document Element 1196
Model Search Sequence 1201
Move Package Attributes Between Elements 1201
RTF Bookmarking In 1196
Sequence of Model Document Elements 1201
Sequence of Package Attributes 1201
tagged Values For Master Document 1197
Visibility Indicators 330
Values 923
Visibility Of Elements
Customize 363
Visible Class Members
Set Diagram Appearance Options 332
Visio
Visio
  MDG Link For, Enterprise Architect 12
Visual
  Layout 96, 245
  Layout (View Menu Option) 95
  Styles 96
  Styles (View Menu Option) 95
Visual Analyzer
  Overview 941
Visual Basic
  Connect To Automation Interface 1584
  Import, Reverse Engineering 872
  Modeling Conventions 939
  Options 910
  Set Up In Automation Interface 1585
Visual Basic.Net
  Import, Reverse Engineering 872
Visual Layouts
  Submenu 96
Visual Studio 2005/2008
  MDG Integration For, Enterprise Architect 12
Visual Studio.NET
  MDG Link For, Enterprise Architect 12
Visual Styles
  Options 246
  Submenu 96
  Windows Style 246
  XP Style 246
Visualize Systems
  Enterprise Architect 6
VM
  Attach To In Java Debug 952

- W -

W3C XML
  Technologies, Introduction 1008
W3C XSD
  Elements and Connectors, Enterprise Architect 154
W3CWSDL
  Elements And Connectors, Enterprise Architect 153
Web
  Page Modeling 1371
  Stereotypes 1371
  Style Templates 1207
  Templates 1207
Web Browser
  Home Website, Access 194
  Internet Email, Access 194
  Web Search Engine, Access 194
Web Page
  Prototype 1368
Web Server
  Java 970
Web Service Definition Language 1026
Web Services
  Submenu (Project Menu) 101
Web Services (WSDL) 1026
  Generate WSDL 1036
  Import WSDL 1037
  Model WSDL 1027
  Model WSDL, Binding 1032
  Model WSDL, Document 1029
  Model WSDL, Message 1032
  Model WSDL, Message Part 1035
  Model WSDL, Namespaces 1027
  Model WSDL, Port Type 1031
  Model WSDL, Port Type Operation 1034
  Model WSDL, Service 1030
Web Site
  Access Any 194
  Access Home 194
  Home, Define Default 231
  Sparx Systems 13
Well Formedness Validation
  Attribute 623
  Diagram 623
  Element 623
  Feature 623
  Relationship 623
What Is
  A Check Constraint? 1070
  A Connector? 1373
  A Foreign Key? 1055
  A Pattern? 507
  A Primary Key? 1052
  A Project? 547
  A Stored Procedure? 1061
  A Trigger Operation? 1070
  A UML Model? 542
  An Index? 1070
  Enterprise Architect? 5
  User Security? 708
  XMI? 636
Window
  Autohide 199
  Bring To Top 124
  Close All 124
  Close All Except Current 124
  Close Current 124
  Combine In Frame 196
  Connections 211
Index 1790

Window
- Constraints 212
- Debug Workbench 981
- Dockable 196
- Element Browser 210
- Floating 196
- Hide 124
- Hierarchy 213
- Linked Requirements 212
- Links 211
- Make Active 124
- Menu 124
- Model Views 177
- Notes 206
- Output 221
- Pan And Zoom 224
- Project Management 220
- Properties (Element) 202
- Rearrange 195
- Relationships 211
- Reload 124
- Remove From Frame 196
- Requirements 212
- Resources 204
- Reveal Autohidden 199
- Rules & Scenarios 212
- System 207
- Tagged Value 214
- Tasks Pane 222

Windows
- Authentication 711
- Service, Apache Tomcat 975

WINE-Crossover
- Using Subversion With Enterprise Architect Under 688

Word
- Special Considerations 1186

Word Substitution
- RTF Report Generation 1171

Wordpad
- Open 109

Work With
- Enterprise Architect 30

Work With Attributes
- Automation Interface Code Example 1675

Work With Methods
- Automation Interface Code Example 1675

Workbench
- Create Variables 987
- Debug Workbench Window 986
- Supported Platforms 986
- Variable Arguments 987
- Variable Constructors 987
- Variables, Create 987
- Variables, Delete 986
- Variables, Description 986
- Variables, Requirements 986

Working
- With MDG Technologies 520
- With UML Connectors 438

Workspace
- Toolbars, Docked 156
- Toolbars, Introduction 156

Workspace Views
- Show/Hide From Toolbar 167
- Toolbar 167

WSDL
- Binding Diagram 1032
- Binding Element 1032
- Document Element 1029
- Elements And Connectors 153
- Group, Enterprise Architect UML Toolbox 153
- Import 1036, 1037
- Message Diagram 1032
- Message Element 1032
- Model 1027
- Namespace Element 1027
- Overview Diagram 1027
- Port Type Diagram 1031
- Port Type Element 1031
- Port Type Operation 1034
- Service Diagram 1030
- Service Element 1030
- Transformation 1113
- UML Toolbox Pages 1027
- Web Services 1027

WSDL Support
- Introduction 1026

- X -

XMI
- Export 636, 638
- Export MOF To 666
- Import 636, 640
- Import And Auditing 743
- Import MOF From 666
- Limitations 644
- Manual Version Control 651
- Specifications 636
- UML DTD 645

XMIType Enum
- Automation Interface 1595

© 1998-2008 Sparx Systems Pty Ltd
XML
  Code Page 243
  Default XML Version 243
  Default XML Directory 243
  Documents, Default Editor 243
  Import Referenced Schema 1023
  Package 646
  Pattern File 507, 508
  Specifications, Options Dialog 243
  Technologies, Introduction 1008
  XMI 1.0 Prefix 243

XML Schema
  Elements and Connectors 154
  Generate In Garden Of Eden Style 1021
  Group, Enterprise Architect UML Toolbox 154
  Submenu (Project Menu) 101
  UML Profile For XSD 1011
  XSD 1009

XSD
  Abstract Models 1017
  Datatype Packages 1017
  Generate 1020
  Import 1023
  Import, Global Element Behaviour 1024
  Model 1010
  Transformation 1114
  XML Schema 1009
XSDany 1011
XSDattribute 1011
XSDattributeGroup 1011
XSDchoice 1011
XSDcomplexType 1011
XSDelement 1011
XSDgroup 1011
XSDrestriction 1011
XSDschema 1011
XSDsequence 1011
XSDsimpleType 1011
XSTopLevelAttribute 1011
XSTopLevelElement 1011
XSDunion 1011

- Z -

Z Order
  Elements 306

Zachman
  Profile 317

Zachman Framework
  MDG Technology For, Enterprise Architect 12

Zoom

Enterprise Architect User Guide