



Version 4.51 User Guide

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



© Copyright 1998-2005 Sparx Systems

Enterprise Architect

Introduction

by Geoffrey Sparks

Enterprise Architect 4.51 is a complete UML based solution for analysing, designing, managing, sharing and building software systems.

Enterprise Architect 4.51 User Guide

© 1998-2005 Sparx Systems

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

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

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

Printed: March 2005

Publisher

Sparx Systems

Managing Editor

Geoffrey Sparks

Technical Editors

Geoffrey Sparks

Paul Mathers

Emma Newman

Melanie Coffey

Dermot O'Bryan

Deborah Johnson

John Redfern

Neil Capey

Evan Sparks

Special thanks to:

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

Table of Contents

Foreword	1
Part I Introduction	3
1 What is Enterprise Architect?	3
2 EA Features	4
3 Copyright Notice	5
4 Trademarks	5
5 Differences Between Desktop, Professional and Corporate Editions	6
6 EA Software Product License Agreement	8
7 Ordering Enterprise Architect	10
8 Support	10
9 Available Helpfile Formats	11
10 Your Feedback	11
11 Zicom Mentor	11
Part II Project Roles and EA	13
1 Business Analyst	13
2 Software Architects	15
3 Software Engineer	16
4 Developer	17
5 Project Manager	19
6 Testers	20
7 Deployment and Rollout	22
8 Technology Developer	23
9 Database Administrator	24
10 Perspectives	26
Part III Getting Started	28
1 Installation	28
2 Starting the Application	29
3 Licence Management	29
Add License Key	30
4 Registering a Full License	32
5 Upgrade an Existing License	33
6 Finding Your License Information	34
Part IV Using Enterprise Architect	37
1 The Start Page	37

The Start Page Options	38
Removing Recent Models	39
Manage My Profile	39
2 The Application Workspace	41
3 Arranging Windows and Menus	42
Dockable Windows	42
Dockable Window 2005 Style	45
Autohide Windows	47
Tear Off Menus	48
4 The Main Menu	48
The File Menu	49
The Edit Menu	50
The View Menu	51
The Project Menu	54
The Diagram Menu	57
The Element Menu	59
The Tools Menu	63
The Customize Window.....	65
Customize Commands.....	66
Customize Toolbars.....	66
Custom Tools	70
Opening External Tools.....	73
Passing Parameters to External Applications.....	74
Customize Keyboard.....	75
Customize Menu.....	78
Customize Options.....	79
The Configuration Menu	79
The Help Menu	82
5 Workspace Toolbars	83
Default Tools Toolbar	84
Project Toolbar	85
Code Generation Toolbar	85
UML Elements Toolbar	87
Diagram Toolbar	88
Current Element Toolbar	89
Current Connector Toolbar	89
Format Toolbar	90
Workspace Views	91
Customize Toolbars	91
Diagram Tabs	92
Status Bar	93
6 The Project Browser	93
Model Context Menu	95
Root Node Package Control Sub-Menu.....	96
Package Context Menu	96
Package Contol Sub-Menu.....	97
Insert Sub-Menu.....	98
Documentation Sub-Menu.....	98
Code Engineering Sub-Menu.....	99
Import/Export Sub-Menu.....	100
Contents Sub-Menu.....	100
Element Context Menu - Project Browser	101

Diagram Context Menu - Project Browser	102
Order Package Contents	103
Setting Default Project Browser Behavior	103
7 Dockable Windows Available	105
The Properties Window	106
The Resource Window	108
The Resource View	109
Favorites	110
The Notes Window	110
The System Window	111
The Source Code Viewer	111
The Element Browser	113
The Relationships Window	114
The Rules Window	115
The Hierarchy Window	116
The Scenario Window	117
The Tagged Values Window	118
Predefined Tagged Value Types	120
Predefined Reference Data	121
Creating a CustomTagged Value Type	124
Assigning defined Tagged Value to an Item	125
Assigning Information to a Tagged Value	127
Allow Duplicate Tags	128
Project Management Window	129
Output Window	130
Instant Help Window	131
8 The UML Toolbox	132
UML Toolbox Shortcut Menu	134
Analysis Group	135
Use Case Group	135
Structure Group	136
Composite Group	137
Communication Group	138
Interaction Group	139
Timing Group	139
State Group	140
Activity Group	141
Component Group	141
Deployment Group	142
Custom Group	143
Profile Group	144
9 View Options	145
Diagram View	145
Report View	146
10 Package Tasks	146
Open a Package	147
Adding Packages	147
Renaming Packages	148
Drag a Package onto a Diagram	148
Show or Hide Package Contents	148
Delete a Package	149
11 Diagram Tasks	149

Select a Diagram	150
Add a Diagram	150
Zoom a Diagram View	151
View Last and Next Diagram	151
Set Diagram Page Size	152
Scale Image to Page Size	153
Lock Diagram	154
Show or Hide Attributes and Operations	155
Layout a Diagram	155
Set Appearance Options	158
Visible Class Members.....	159
Undo Last Action	160
Highlight Context Element	160
12 Diagram Views	161
Close Current Diagram	161
Close All Except Current Diagram	162
Reload Current Diagram	162
Save All Modified Diagrams	162
Close All Diagrams	163
13 Element Tasks	163
Add an Element	163
Connecting Elements	164
Moving Elements and Packages	165
Auto Counters	166
Searching a Project	166
Default Element Templates	168
14 Element Inplace Editing Options	170
Inplace Element Item Tasks	170
Edit Element Name	172
Edit Attribute or Operation Stereotype	173
Edit Attribute and Operation Scope	175
Edit Attribute Keyword	176
Edit Operation Parameter Keyword	177
Edit Parameter Kind	179
Insert new Attribute or Operation	180
Insert Operation Parameter	182
Insert Maintenance Feature	184
Insert Testing Feature	188
15 Defaults and User Settings	192
Configure Local Options	192
General	193
Standard Colors.....	195
Diagram.....	196
Behavior	197
Diagrams	198
Objects	199
Element Visibility.....	201
Communication Colors	201
XML Specifications.....	202
UML Element Toolbox.....	203
Generation.....	205
Custom Layouts	205
Visual Styles	206

16 Register Add-In	206
Register Zicom Mentor	207
17 Keyboard Shortcuts	209
Part V The UML Language	213
1 What is UML	213
2 UML Diagrams	214
Behavioral Diagrams	215
Activity Diagram	217
Use Case Diagram	219
State Machine Diagram (formerly State Diagram)	221
Regions	222
Pseudo-States	224
Interaction Diagrams	225
Timing Diagram	225
Sequence Diagram	226
Denote Lifecycle of an Element	228
Layout of Sequence Diagrams	229
Sequence Element Activation	230
Lifeline Activation Levels	231
Sequence Elements	235
Sequence Message Label Visibility	236
Communication Diagram (formerly Collaboration Diagram)	236
Communication Diagrams in Color	238
Interaction Overview Diagram	239
Structural Diagrams	241
Package Diagram	242
Class Diagram	244
Object Diagram	246
Composite Structure Diagram	248
Properties	250
Component Diagram	252
Deployment Diagram	253
Additional Diagrams	255
Analysis Diagram	255
Custom Diagram (Extended Class)	256
Database Schema	257
Robustness Diagram	258
3 UML Elements	259
Behavioral Diagram Elements	259
Structural Diagram Elements	262
Basic Elements	263
Action	264
Action Notation	264
Action Expansion Node	266
Action Pin	268
Local Pre/Post Conditions	270
Activity	271
Activity Notation	273
Activity Parameter Nodes	274
Activity Partition	276
Actor	277

Artifact	278
Choice	278
Class	280
Active Classes.....	281
Parameterized Classes (Templates).....	281
Collaboration	283
Collaboration Occurrence.....	285
Combined Fragment.....	286
Create a Combined Fragment.....	288
Interaction Operators.....	288
Component.....	291
Datastore.....	291
Decision.....	292
Deployment Spec.....	293
Diagram Gate.....	294
Endpoint.....	295
Entry Point.....	296
Exception.....	297
Expansion Region.....	298
Add Expansion Region.....	300
Exit Point.....	300
Final	301
Flow Final.....	302
Fork/Join	302
Fork	304
Join	305
History	306
Initial	308
Interaction Occurrence.....	308
Interface.....	309
Interruptible Activity Region.....	310
Add Interruptible Activity Region.....	311
Junction.....	311
Lifeline	313
Node	313
Object	314
Instance Classifier.....	315
Run-time State.....	316
Define a Run-time Variable.....	316
Remove a Defined Variable.....	318
Define a Run-time State.....	318
Package.....	319
Part	319
Partition.....	320
Port	321
Adding a Port to an Element.....	322
Managing Inherited and Redefined Ports.....	322
Qualifiers.....	323
Receive.....	325
Region	326
Send	326
State	327
Composite State.....	328
State Lifeline.....	329

State/Continuation.....	331
Continuation	331
State Invariant.....	332
SubActivity.....	333
SubMachine.....	334
Synch	335
System Boundary.....	335
Terminate.....	336
Use Case.....	337
Use Case Extension Points.....	338
Using Rectangle Notation.....	340
Value Lifeline.....	340
Inbuilt and Extended Stereotypes	342
Analysis Stereotypes.....	342
Boundary.....	343
Create a Boundary.....	344
Business Modeling Stereotypes 1.....	344
Business Modeling Stereotypes 2.....	345
Call	346
Composite Elements.....	346
Control	347
Create a Control Element.....	347
Entity	348
Create an Entity.....	349
Event	349
Hyperlinks.....	349
N-Ary Association.....	353
Other Elements.....	353
Process.....	354
Recursion.....	354
Requirements.....	354
Screen	355
Self-Message.....	357
Table	357
UI Element.....	357
Web Stereotypes.....	359
Worker	360
4 UML Connections	360
Aggregate	363
Change Aggregation Link Form.....	363
Assembly	364
Associate	364
Diagonal Representation.....	365
Association Class	366
Link a New Class to an Existing Association.....	367
Compose	368
Connector	368
Control Flow	369
Delegate	370
Dependency	371
Applying a Stereotype.....	371
Deployment	372
Expose Interface	373
Extend	373

Generalize	374
Include	374
Interrupt Flow	375
Manifest	376
Message	376
Message (Communication Diagram)	377
Create a Communication Message	377
Re-Order Messages	378
Message (Sequence Diagram)	380
Message (Timing Diagram)	381
Create a Timing Message	382
Message Endpoint	384
Message Label	385
Nesting	386
Object Flow	386
Using Object Flows in Activity Diagrams	387
Occurrence	388
Package Import	389
Package Merge	389
Realize	390
Role Binding	391
Represents	391
Trace	392
Transition	392
Use	393

Part VI Modeling with UML 395

1 Working with UML Diagrams	396
Diagram Context Menu	398
Set the Default Diagram	399
Common Diagram Tasks	400
New Diagrams	401
Deleting Diagrams	402
Diagram Properties	402
Renaming Diagrams	404
Copy Diagram Element	404
Convert Linked Element to Local Copy	405
Z Order Elements	405
Copy Image to Disk	406
Copy Diagram Image to Clipboard	406
Change Diagram Type	406
Open a Package	407
Duplicate a Diagram	408
Set Feature Visibility	410
Insert Diagram Properties Note	411
Autosize Elements	411
Drop Elements from the Project Browser	411
Pasting from the Project Browser	413
Pasting Multiple Items from the Project Browser	414
Pasting Composite Elements	415
Pasting Activities	416
Place Related Elements on Current Diagram	416
Swimlanes	417

Using the Image Manager.....	419
Select Alternate Image.....	421
Create Custom Diagram Background.....	421
Import Image Library.....	421
Show Realized Interfaces for a Class.....	423
Label Menu Section.....	424
2 Working with UML Elements	425
Element Context Menu	427
Properties Menu Section.....	429
Advanced Properties Dialog.....	429
Embedding and Features Menu Section.....	430
Embedded Elements.....	430
Type Menu Section.....	432
Code Engineering Menu Section.....	432
Appearance Menu Section.....	432
Common Actions Menu Section.....	433
Insert Related Elements.....	433
Element Context Menu - Multiple Selection	434
Common Element Tasks	435
Creating Elements.....	435
New Elements.....	437
Set Element Parent.....	438
Show Element Usage.....	439
Set Up References (Cross References).....	440
Copying Attributes and Operations Between Elements.....	442
Moving Attributes and Operations between Elements.....	443
Moving Elements between Packages.....	445
Changing Element Type.....	445
Moving Elements.....	446
Aligning Elements.....	447
Resizing Elements.....	448
Deleting Elements.....	449
Customize Visible Elements.....	450
Creating Notes and Text.....	451
Configure an Element's Default Appearance.....	453
Get/Set Project Custom Colors.....	455
Attributes and Operations	457
Adding Use Case Attributes and Operations to Activity Diagrams.....	458
Attributes.....	458
Attributes Main Page.....	459
Attributes Detail.....	461
Attribute Constraints.....	462
Attribute Tagged Values.....	463
Creating Properties.....	464
Displaying Inherited Attributes.....	466
Operations.....	468
Operations Main Page.....	469
Operations Detail.....	471
Operation Parameters.....	473
Operation Parameter Tagged Values.....	474
Operation Parameters by Reference.....	475
Operation Constraints.....	477
Operation Tagged Values.....	477
Override Parent Operations.....	478

Displaying Inherited Operations.....	480
Element Properties	482
Properties.....	482
General Settings.....	483
Advanced Settings.....	484
Requirements.....	485
External Requirements.....	486
Constraints.....	487
Links	489
Scenarios.....	489
Tagged Values.....	490
Advanced Tag Management.....	491
Quick Add of Tagged Values.....	493
Associated Files.....	495
Object Classifiers	496
Using Classifiers.....	497
Boundary Element Settings.....	498
Compartments	499
Responsibility Compartment.....	500
Constraint Compartment.....	501
Tag Compartment.....	501
Linking Notes	501
Linking Notes to Internal Documentation.....	501
Link a Note to an Element.....	502
3 Working with UML Connections	504
Connection Context Menu	504
Properties Menu Section.....	505
Type Specific Menu Section.....	505
Common Actions Menu Section.....	505
Style Menu Section.....	506
Appearance Menu Section.....	506
Common Connection Tasks	507
Connecting Elements.....	507
Connector Styles.....	507
Arranging Connections.....	509
Change Connector Type.....	510
Reverse Connector.....	511
Deleting Connections.....	511
Hide/Show Connectors.....	512
Hide/Show Labels.....	513
Change the Source or Target Element.....	514
Set Relation Visibility.....	515
Add a Note to a Link.....	516
Tree Style Hierarchy.....	517
Create Link.....	518
Show Uses Arrow Head.....	520
Association Properties	520
Connection Constraints.....	522
Source Role.....	523
Destination Role.....	525
Role Tagged Values.....	525
Message Scope.....	526
4 UML Stereotypes	527

Standard Element Stereotypes	530
Stereotypes with Alternate Images	533
5 MDG Technologies	533
Creating MDG Technologies	534
Adding Profile in MDG Technology Wizard.....	538
Adding Pattern in MDG Technology Wizard.....	539
Adding Tagged Values in MDG Technology Wizard.....	540
Adding Code Modules in MDG Technology Wizard.....	541
Adding Images in MDG Technology Wizard.....	542
Working with MDG Technologies	543
Import MDG Technologies	545
Add MDG Technologies to UML Toolbox	546
6 UML Profiles	548
Creating Profiles	551
Create a UML Profile.....	551
Add Stereotypes and Metaclasses to UML Profiles.....	552
Define Stereotype Tags.....	554
Define Stereotype Constraints.....	556
Add Enumerations to UML Profiles.....	557
Export a UML Profile.....	558
Using Profiles	560
Import a UML Profile.....	561
Add Profiles to UML Toolbox.....	562
Example Use of UML Profile.....	564
Tagged Values in Profiles.....	564
Add Profile Connector to Diagram.....	565
Synchronize Tags and Constraints.....	566
Deprecated Profile Tasks	567
Save a Diagram as a UML Profile (Deprecated).....	567
Applying Profiles to Current Model (Deprecated).....	569
Profile References	571
Supported Types.....	571
Profile Structure.....	574
Supported Attributes.....	575
Example Profile.....	575
7 UML Patterns	577
Create a Pattern	577
Import a Pattern	580
Use a Pattern	581
8 Requirements Management	583
Creating Requirements	584
External Requirements	584
Color Coded External Requirements	586
Internal Requirements	587
Move Internal Requirements to External Requirement	588
Requirement Properties	590
Composition	591
Implementation	591
Requirements Hierarchy	592
The Matrix	592
Dependency Report	592
9 Element Relationship Matrix	593

Open the Relationship Matrix	594
Setting Source and Target Package	595
Setting Element Type	595
Setting the Link Type and Direction	596
Matrix Options	597
Modifying Relationships in Matrix	597
Exporting to CSV	599
Printing the Matrix	600
Profiles	601
10 Business Modeling	602

Part VII Model Management 608

1 Create and Open Model Files	609
Open Project Dialog	609
Open an Existing Project	610
Create a Model	611
Create a New Model File.....	611
Setting Up a Database Repository	612
Setting Up an ODBC Driver.....	613
Setup a MySQL ODBC Driver.....	613
Setup a PostgreSQL ODBC Driver.....	616
Setup an Adaptive Server Anywhere ODBC Driver.....	619
Connecting to a Data Repository.....	624
Connect to a MySQL Data Repository (Corporate Edition only).....	625
Connect to a SQL Server Data Repository (Corporate Edition only).....	627
Connect to an Oracle9i Data Repository (Corporate Edition only).....	630
Connect to a PostgreSQL Data Repository (Corporate Edition only).....	633
Connect to an Adaptive Server Anywhere Data Repository (Corporate Edition only)	636
Connect to a MSDE Server Data Repository (Corporate Edition only).....	639
Creating a Repository.....	639
Create a New MySQL Repository (Corporate Edition only).....	639
Create a New SQL Server Repository (Corporate Edition only).....	642
Create a New Oracle9i Server Repository (Corporate Edition only).....	644
Create a New PostgreSQL Repository (Corporate Edition only).....	645
Create a New Adaptive Server Anywhere Repository (Corporate Edition Only)	647
Create a New MSDE Server Repository (Corporate Edition only).....	651
Create and Open Model Files Discussion	651
2 Upgrading Models	652
The Upgrade Wizard	652
Upgrade Replicas	653
3 Data and Model Integrity	654
Checking Data Integrity	654
Running SQL Patches	656
4 Data Transfer	656
Perform a Data Transfer	657
Why Compare Models?	658
Comparing Models	658
Copy Packages from One Project to Another	659
5 Upsizing Models	661
Upsizing to MySQL	662

Upsizing to SQL Server	664
Upsizing to Oracle	666
Upsizing to PostgreSQL	668
Upsizing to Sybase Adaptive Server Anywhere (ASA)	670
Upsizing to SQL Server Desktop Engine (MSDE)	672
6 Model Maintenance	672
Rename a Project	673
Compact a Project	673
Repair a Project	673
7 Manage Views	675
Add Additional Views	675
Rename Views	676
Delete Views	678
8 XMI Import and Export	679
Export to XMI	680
Import from XMI	682
Limitations of XMI	683
The UML DTD	683
Controlled Packages	683
Controlled Package Menu	684
Configure Packages	685
Get Shared File from Repository Dialog	687
Disconnect Controlled Package	687
Save a Package	688
Load a Package	689
Batch XMI Export	689
Batch XMI Import	690
Manual Version Control with XMI	691
9 CSV Import and Export	692
CSV Specifications	692
CSV Export	694
CSV Import	696
10 Version Control Options	697
Version Control	697
Version Control Overview	698
Version Control Setup	698
Using Version Control	699
Version Control Reference	699
Version Control Setup Menu	700
Version Control Options Dialog	700
Version Control Menu	700
SCC Providers Dialog	702
Version Control Options SCC	703
SCC Version Control User Options	706
SCC Version Control Show Interface	706
CVS with Remote Repositories	707
CVS with Local Repositories	709
Configure Package for Version Control	710
Add Previously defined Version Control Configurations	712
Checking In and Checking out Packages	713
Review Package History	714
Remove Package from Version Control	715

Including Other Users Packages	716
SCC Version Control Upgrade for 4.5	716
11 Model Sharing and Team Deployment	717
Sharing an Enterprise Architect Project	718
Sharing a Project	719
Distributed Development	719
User Security	720
Security Policy.....	721
Enabling Security.....	722
Maintaining Users.....	723
Set Up User Groups.....	724
Set Up Single Permissions.....	725
View All User Permissions.....	726
Maintaining Groups.....	727
Set Group Permissions.....	728
List of Available Permissions.....	729
View and Manage Locks.....	730
Password Encryption.....	731
Locking Model Elements.....	733
Locking Packages.....	733
Apply a User Lock.....	734
Managing Your Own Locks.....	736
Replication	737
Creating Replicas	738
Design Masters.....	738
Synchronizing Replicas	739
Remove Replication.....	740
Upgrading Replicas.....	740
Resolving Conflicts.....	740
12 Spell Checking	741
Using the Spell Checker	742
Correcting Words	743
User Dictionaries	744
Select Language	744
13 Reference Data	745
People	746
Model Authors.....	746
Clients	747
Resources.....	748
Roles	749
General Types	750
Requirement Types.....	750
Status Types.....	751
Constraint Status Types.....	751
Constraint Types.....	752
Scenario Types.....	753
Maintenance	754
Problem Types.....	754
Testing Types.....	755
Metrics and Estimation	756
UML	756
Standard Element Stereotypes.....	757
Tagged Values.....	757

Data Types	758
Import and Export Reference Data	760
Export Reference Data	760
Import Reference Data	761

Part VIII Project Management 763

1 Estimation	763
TCF Dialog	763
ECF Dialog	765
Estimating Project Size	766
Default Hours	767
2 Resources	768
Resource Management	769
Effort Management	770
Risk Management	771
Metrics	772
Resource Report	772
Roles	773
System Clients	774
Effort Types	775
Metric Types	776
Risk Types	777
3 Testing	778
The Testing Workspace	779
The Test Details Dialog	779
Unit Testing	780
Integration Testing	782
System Testing	783
Acceptance Testing	784
Scenario Testing	786
Import Scenario as Test	787
Import Test from other elements	788
Test Details Report	790
Show Test Scripts in Compartments	791
Test Documentation	792
4 Maintenance	793
The Maintenance Workspace	793
Show Maintenance Scripts in Compartments	794
Maintenance Element Properties	794
5 Changes and Defects	796
Defects (Issues)	796
Changes	797
Element Properties	799
Assign People to Defects or Changes	799
6 Model Tasks List	800
Model Tasks Tab	800
Adding, Modifying and Deleting Tasks	801
7 Project and Model Issues	802
Project Issues Dialog	802
Model Issues Tab	804
Add, Delete and Modify Issues	805

Using the Project Issues Dialog.....	805
Using the Model Issues Tab.....	806
Generate a Report	807
Reports - Using the Project Issues Dialog.....	807
Reports - Using the Model Issues Tab.....	807
Report Output Sample.....	809
8 Model Glossary	810
The Glossary Dialog	811
Model Glossary Tab	811
Add, Delete and Modify Glossary Entries	812
Using the Glossary Dialog.....	812
Using the Model Glossary Tab.....	813
Generate a Report	814
Glossary Report Output Sample	816
9 Update Package Status	817
10 Manage Bookmarks	819
Part IX Code Engineering	821
1 Reverse Engineer Source Code	821
Import Source Code	822
Import a Directory Structure	824
Import C++	825
Import C#	825
Import Delphi	826
Import Java	826
Import PHP	826
Import Visual Basic	826
Import VB.Net	826
2 Generate Source Code	827
How to Generate Code	827
Generate a Single Class	828
The Code Generation Dialog	829
Generate a Group of Classes	830
Generate a Package	831
Generate Package Dialog	833
Update Package Contents	833
Namespaces	834
3 Code Engineering Settings	835
General Options	835
Options - Generation	836
Options - Code Editors	837
Options - Object Lifetimes	837
Options - Attribute/Operations	838
Code Page for Source Editing.....	839
Local Paths	840
Local Path Dialog	841
Language Macros	842
Setting Collection Classes	843
Language Options	845
Options - C++.....	846
Options - C#.....	846
Options - Delphi.....	847

Delphi Properties.....	848
Options - Java.....	850
Options - PHP.....	851
Options - Visual Basic.....	851
Options - VB.Net.....	852
Reset Options.....	853
4 Synchronize Model and Code	854
5 MDG Link and Code Engineering	855
6 Code Template Framework	855
Code Templates	856
Base Templates.....	856
Custom Templates.....	859
Execution of Code Templates.....	861
Code Template Syntax.....	862
Literal Text	862
Macros	862
Template Substitution Macros.....	863
Field Substitution Macros.....	863
Tagged Value Macros.....	870
Function Macros.....	871
Control Macros.....	874
Variables	877
Variable Definitions.....	877
Variable References.....	878
The Code Template Editor	879
Overriding Default Templates.....	880
Adding New Stereotyped Templates.....	881
Creating Templates For Custom Languages.....	882
Importing and Exporting Code Templates.....	882
Synchronizing Code	882
Synchronizing Existing Sections.....	883
Adding New Sections to Existing Features.....	883
Adding New Features and Elements.....	884
7 XML Schema Generation	884
Getting Started	884
Steps to Generate XSD	885
Example	885
UML Profile for XML	887
Part X Data Modeling	897
1 A Data Model Diagram	897
2 Create a Table	898
3 Set Table Properties	899
Set Table Owner	901
Set MySQL Table Type	903
4 Create Columns	905
5 Primary Key	907
Primary Key Extended Properties	907
6 Foreign Keys	908
7 Indexes and Triggers	912

8 Generate DDL	913
9 Generate DDL for a Package	915
10 Data Type Conversion Procedure	916
11 Data Type Conversion for a Package	917
12 DBMS Datatypes	919
13 Import Database Schema from ODBC	920
Select a Data Source	922
Select Tables	923
The Imported Class Elements	924

Part XI Creating Documents 927

1 RTF Documents	927
The RTF Dialog	927
Create a Rich Text Document.....	928
Document a Single Element.....	929
The RTF Report Dialog.....	929
Set the Main RTF Properties.....	930
Apply a Filter.....	931
Exclude Elements.....	932
RTF Diagram Format.....	932
Model Include.....	933
RTF Options.....	933
RTF Selections.....	934
Generate the Report.....	935
Diagram Only Report.....	936
Report Templates.....	936
Include or Exclude a Package from Report.....	937
Save as Document.....	938
Using MS Word	939
Open a Report in Microsoft Word.....	939
Changing Linked Images to Embedded Images.....	939
Bookmarks.....	940
Other Features of Word.....	943
Add Table of Contents.....	943
Add Table of Figures.....	944
Add Headers and Footers.....	945
Manipulating Tables in Word.....	946
Refresh Links	948
Other Documents	949
Dependency Report.....	949
Implementation Report.....	950
Set Target Types Dialog.....	952
Testing Report.....	952
RTF Style Templates	953
Custom Language Settings	955
2 HTML Reports	957
Creating an HTML Report	957
The Generate HTML Report Dialog	958
Making the Output Available on the Web	960
Web Style Templates	960

3 Virtual Documents	963
Create a Document Object	964
Add Packages to Your Document Object	964
Rearrange the Package Order	965
Delete a Package from Your Document Object	966
Generate the Document	967
Part XII Automation and Scripting	970
1 The Automation Interface	970
Using the Automation Interface	971
Connecting to the Interface.....	971
Set Up VB	972
Examples and Tips.....	974
Call from EA	975
Available Resources.....	976
Reference	976
Interface Overview.....	977
App	980
Enumerations.....	981
MDGMenu Enum.....	981
EnumRelationSetType Enum.....	981
ConstLayoutStyles Enum.....	981
ObjectType Enum.....	982
XMIType Enum.....	982
Repository.....	983
Repository	985
Author	990
Client	991
Collection	992
Datatype	993
Package	994
ProjectIssues.....	996
ProjectResource.....	997
Reference	998
Stereotype	999
Task	1000
Term	1001
EventProperty.....	1002
EventProperties.....	1002
Property Type.....	1003
Element.....	1004
Constraint	1006
Effort	1006
Element	1007
File	1012
Issue (Maintenance).....	1013
Metric	1014
Requirement	1014
Resource	1015
Risk	1016
Scenario	1017
TaggedValue.....	1018
Test	1019

Element Features.....	1019
Attribute	1020
Attribute Constraint.....	1022
Attribute Tag	1023
Method	1024
Method Constraint.....	1026
Method Tag	1027
Parameter	1027
Partitions	1028
Embedded Elements.....	1029
Transitions	1030
Custom Properties.....	1031
Connector.....	1031
Connector	1032
Connector Constraints.....	1035
Connector End.....	1035
Connector Tag.....	1037
Role Tag	1037
Diagram.....	1038
Diagram	1039
Diagram Links.....	1042
Diagram Objects.....	1043
Project Interface.....	1044
Project	1044
Code Samples.....	1051
Open the Repository.....	1051
Iterate Through an EAP File.....	1052
Add and Manage Packages.....	1052
Add and Manage Elements	1053
Add a Connector.....	1053
Add and Manage Diagrams	1054
Adding and Deleting Attributes and Methods.....	1055
Element Extras.....	1055
RepositoryExtras	1058
Stereotypes	1059
Working with Attributes.....	1060
Working with Methods	1060
2 The Project Interface (XML)	1062
The Read Only Interface	1062
Automation Interfaces	1062
Opening a Project	1063
Enumerating Views	1064
Enumerating	1065
Enumerating Diagrams	1066
Get Diagrams	1067
Enumerating Elements	1068
Get Element	1069
Enumerating Links	1077
Get Link	1078
Get Diagram Image	1080
3 Add-Ins	1080
Overview	1081
Add-In tasks	1081

Creating Add-Ins.....	1082
Defining Menu Items.....	1082
Deploying Add-Ins.....	1083
Tricks and Traps.....	1085
Add-In Events	1086
EA_Connect.....	1086
EA_Disconnect.....	1087
EA_GetMenuItems.....	1088
EA_MenuClick.....	1088
EA_GetMenuState.....	1089
EA_ShowHelp.....	1090
Broadcast Events	1091
EA_FileOpen.....	1091
EA_OnPreNewElement.....	1092
EA_OnPreNewConnector.....	1093
EA_OnPostNewElement.....	1094
EA_OnPostNewConnector.....	1094
EA_OnDeleteTechnology.....	1095
EA_OnImportTechnology.....	1096
Custom Views	1096
Creating a Custom View.....	1097
MDG Add-Ins	1097
MDG Events.....	1098
MDG Add-Ins MDGBuild Project.....	1098
MDG Add-Ins MDGConnect.....	1099
MDG Add-Ins MDGDisconnect.....	1100
MDG Add-Ins MDGGetConnectedPackages.....	1101
MDG Add-Ins MDGGetProperty.....	1101
MDG Add-Ins MDGMerge.....	1102
MDG Add-Ins MDGNewClass.....	1104
MDG Add-Ins MDGPostGenerate.....	1105
MDG Add-Ins MDGPostMerge.....	1106
MDG Add-Ins MDGPreGenerate.....	1106
MDG Add-Ins MDGPreMerge.....	1107
MDG Add-Ins MDGPreReverse.....	1108
MDG Add-Ins MDGRunExe.....	1108
MDG Add-Ins MDGView.....	1109

Part XIII Glossary of Terms 1112

1 A (Glossary)	1112
2 B (Glossary)	1114
3 C (Glossary)	1115
4 D (Glossary)	1119
5 E (Glossary)	1121
6 F (Glossary)	1122
7 G (Glossary)	1123
8 H (Glossary)	1124
9 I (Glossary)	1124
10 J (Glossary)	1126
11 L (Glossary)	1126

12 M (Glossary)	1126
13 N (Glossary)	1129
14 O (Glossary)	1129
15 P (Glossary)	1130
16 Q (Glossary)	1133
17 R (Glossary)	1133
18 S (Glossary)	1135
19 T (Glossary)	1139
20 U (Glossary)	1140
21 V (Glossary)	1142
Part XIV Acknowledgements	1144
Index	1145

Foreword

This user guide provides an introduction to the features contained in Enterprise Architect 4.51 - a UML CASE tool for developing and building software systems with UML.

Part



1 Introduction



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

Quick Start:

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

See Also:

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

1.1 What is Enterprise Architect?

What is Enterprise Architect?

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

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

Some of the key features of Enterprise Architect are:

- Create UML model elements for a wide range of purposes

- Place those elements in diagrams and packages
- Create connections between elements
- Document the elements you have created
- Generate code for the software you are building
- Reverse engineer existing code in several languages

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

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

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

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

1.2 EA Features

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

Enterprise Architect Features:

- Model complex software and hardware systems in UML-compliant notation
- Generate and reverse engineer C++, Java, C#, VB.NET, Delphi, PHP, Visual Basic and VB.NET (Professional and Corporate Editions only)
- Model databases and generate DDL scripts. Reverse database schema from ODBC connections (Professional and Corporate Editions only)
- Manage change, maintenance, test scripts and more
- Model dependencies between elements
- Set object classifiers
- Model system dynamics and state
- Model class hierarchies
- Model deployment, components and implementation details
- Collect project issues, tasks and system glossary
- Assign resources to model elements and track effort expended versus required effort
- Output detailed and quality documentation in RTF and HTML formats
- Output models in XMI 1.0, XMI 1.1 and XMI 1.2 compatible format for import or export to other XMI compliant tools
- Import models in XMI 1.0, XMI 1.1 and XMI 1.2 format from other tools
- Version Control through XMI using SCC and CVS version control configurations.
- UML Profiles are available to create custom modeling extensions
- Save and Load complete diagrams as UML Patterns
- Show links between elements in tabular format using the Relationship Matrix

- Script and work with the UML elements and automate common tasks using a detailed Automation Interface
- Connect to SQL Server, MySQL or Oracle9i databases (Corporate edition only)
- Migrate changes across a distributed environment with JET Replication
- Use Controlled Packages based on XMI import/export

Plus much more!

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

1.3 Copyright Notice

Copyright © 1998-2005 Sparx Systems Pty. Ltd. All rights reserved.

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

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

Sparx Systems Pty. Ltd.

7 Curtis St,
Creswick, Victoria 3363,
AUSTRALIA

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

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

Website: www.sparxsystems.com.au

1.4 Trademarks

Acknowledgement of Trademarks

The following are Trademarks of Microsoft:

- Microsoft Word
- Microsoft Office
- Windows®
- ActiveX

The following are Registered Trademarks of OMG:

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

The following are Trademarks of the OMG:

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

1.5 Differences Between Desktop, Professional and Corporate Editions

Enterprise Architect is available in three editions, the **Desktop**, **Professional** and **Corporate**. The desktop edition does not support the code generation functions of the Professional and Corporate editions. The Professional Edition has all of the features available in the Desktop edition with the addition of code engineering and model sharing. The Corporate Edition has all of the capabilities of the Desktop and Professional Editions but adds the ability to connect to MySQL, SQL Server, PostgreSQL, Sybase Adaptive Server Anywhere and Oracle9i DBMS backends as the shared repository. This edition also supports user security, user logins, user groups and user level locking of elements.

Functionality for each version is described below:

Functionality	Corporate Edition	Professional Edition	Desktop Edition
.EAP Files	Yes	Yes	Yes
Shared Models	Yes	Yes	No
Source Code Engineering	Yes	Yes	No
Database Engineering	Yes	Yes	No
Microsoft Access Repository	Yes	Yes	Yes
SQL Server, MySQL, Oracle9i, PostgreSQL, MSDE, Adaptive Server Anywhere Database Repositories	Yes	No	No
Version Control	Yes	Yes	Yes
Replication	Yes	Yes	No
MDG Technologies	Yes	Yes	No
MDG Link for Eclipse and MDG Link for Visual Studio.NET	Yes	Yes	No
Security	Yes	No	No

EA Desktop Edition

The Desktop edition is targeted at single developers producing UML analysis and design models. This edition includes all the features of the Professional edition except code engineering (import/export of source code and DDL), the Active-X interface and the ability to share a model amongst multiple users.

EA Professional Edition

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

EA Corporate Edition

Aimed at larger development teams, the Corporate edition supports everything in the Desktop and Professional versions, plus the ability to connect to MySQL, SQL Server, PostgreSQL, Sybase Adaptive Server Anywhere and Oracle9i DBMS backends as the shared repository. This provides additional scalability and improved concurrency over the shared .EAP file approach to model sharing. Support for MDG Technologies and MDG Link is included with the Corporate version of EA. This edition also supports user security, user logins, user groups and user level locking of elements. The Corporate edition support for user/group based security - with locking at diagram and element levels. Security comes in two modes: in the first mode, all elements are considered 'writeable' until explicitly locked by a user or group; in mode two, all elements are considered locked until checked out with a user lock.

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

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

See also:

Ordering Enterprise Architect

1.6 EA Software Product License Agreement

SOFTWARE PRODUCT LICENSE AGREEMENT

Enterprise Architect - UML CASE Tool - Desktop, Professional and Corporate Editions, Version 4.51
Copyright (C) 1998-2005 Sparx Systems Pty Ltd. All Rights Reserved

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

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

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

DEFINITIONS

In this End User Licence Agreement, unless the contrary intention appears,

"ACADEMIC EDITION" means an edition of the Software Product purchased for educational purposes at an academic discount price.

"EULA" means this End User Licence Agreement

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

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

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

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

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

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

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

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

GRANT OF LICENCE

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

- a) to install and use one copy of the SOFTWARE PRODUCT, or in its place, any prior version for the same operating system, on a single computer. As the primary user of the computer on which the SOFTWARE PRODUCT is installed, YOU may make a second copy for your exclusive use on either a home or portable computer.
- b) to store or install a copy of the SOFTWARE PRODUCT on a storage device, such as a network server, used only to install or run the SOFTWARE PRODUCT over an internal network. If YOU wish to increase the number of users entitled to concurrently access the SOFTWARE PRODUCT, YOU must notify SPARX and agree to pay an additional fee.
- c) to make copies of the SOFTWARE PRODUCT for backup and archival purposes.

EVALUATION LICENCE

The Trial version of Enterprise Architect is not free software. Subject to the terms of this agreement, YOU are hereby licenced to use this software for evaluation purposes without charge for a period of 30 days. Upon expiration of the 30 days, the Software Product must be removed from the computer. Unregistered

use of Enterprise Architect after the 30-day evaluation period is in violation of Australian, U.S. and international copyright laws.

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

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

EA LITE

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

ADDITIONAL RIGHTS AND LIMITATIONS

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

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

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

ASSIGNMENT

YOU may only assign all your rights and obligations under this EULA to another party if YOU supply to the transferee a copy of this EULA and all other documentation including proof of ownership. Your licence is then terminated.

TERMINATION

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

WARRANTIES AND LIABILITY

WARRANTIES

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

EXCLUSIONS

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

YOUR use or misuse of the SOFTWARE PRODUCT;

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

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

Termination of this EULA by either party for any reason.

LIMITATION

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

legislation be limited, at SPARX'S option to any one or more of the following upon return of the SOFTWARE PRODUCT and a copy of the receipt:

If the breach relates to the SOFTWARE PRODUCT:

the replacement of the SOFTWARE PRODUCT or the supply of an equivalent SOFTWARE PRODUCT;
the repair of such SOFTWARE PRODUCT; or the payment of the cost of replacing the SOFTWARE PRODUCT or of acquiring an equivalent SOFTWARE PRODUCT; or
the payment of the cost of having the SOFTWARE PRODUCT repaired;

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

the supplying of the services again; or

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

TRADEMARKS

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

GOVERNING LAW

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

1.7 Ordering Enterprise Architect

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

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

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

Purchase Options

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

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

1.8 Support

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

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

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

1.9 Available Helpfile Formats

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

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

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

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

1.10 Your Feedback

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

You can access our online feedback pages at

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

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

1.11 Zicom Mentor

Want to learn more about UML? Looking for a refresher course perhaps? Or keen to get started with EA immediately, but not sure how to begin?

Self-paced training is available now at www.sparxsystems.com.au/zm/ZicomMentor.htm

Zicom Mentor is a comprehensive UML reference/assistant tool which integrates perfectly with EA. Zicom Mentor will immediately answer your queries, teach and test your knowledge, and most importantly jumpstart your project. Zicom Mentor includes interactive courses and quizzes, a huge body of UML reference material and fast, reliable performance.

Zicom Mentor is priced at US\$49.95. The **free trial version** is available from from www.sparxsystems.com.au/zm/ZicomMentorEvaluate.htm

Further information on how to register Zicom Mentor can be found on the [Register Zicom Mentor](#) page.

Part



2 Project Roles and EA

Enterprise Architect and your Role

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

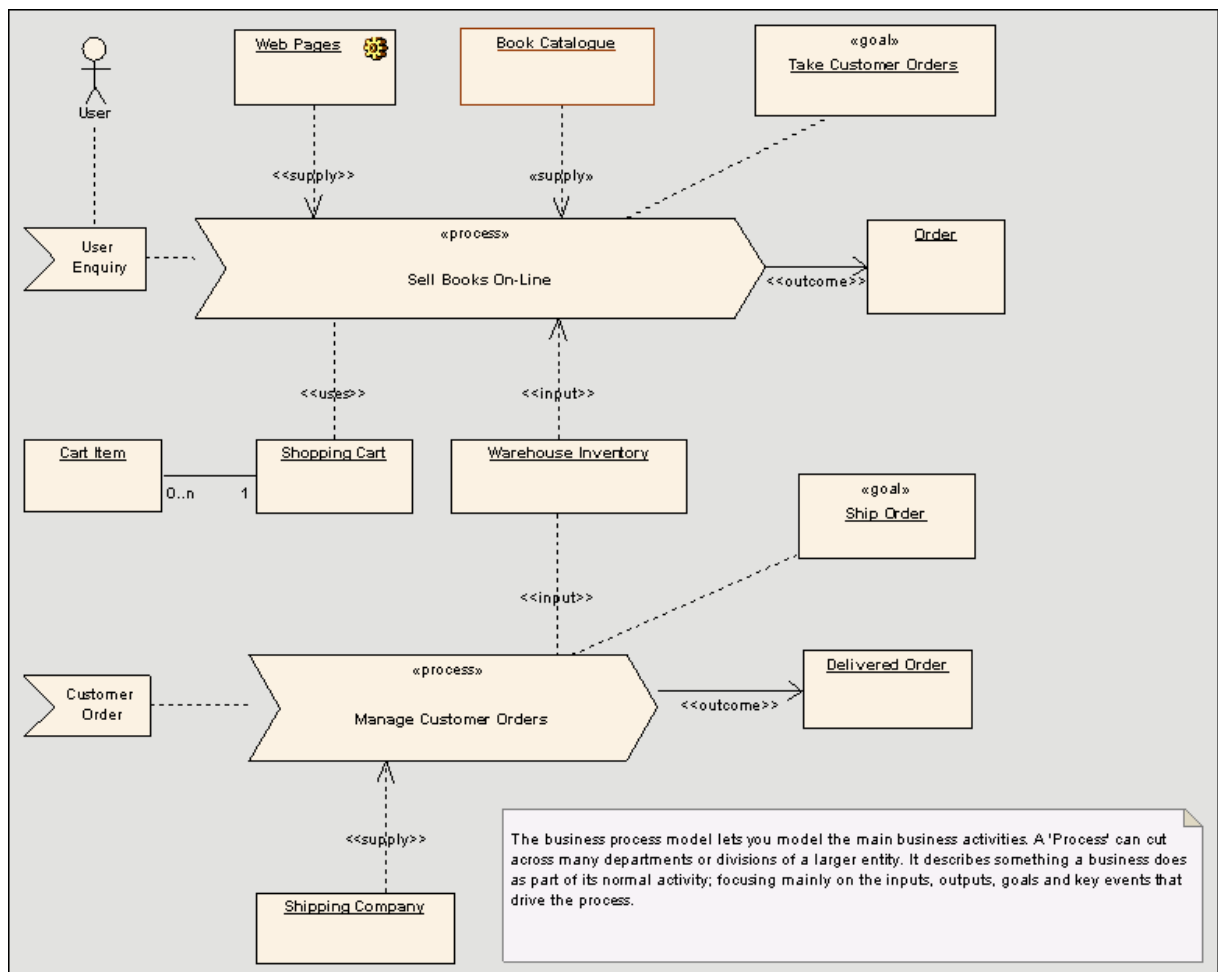
Use the following links to explore how EA can assist you in carrying out your role within a model driven project.

See:

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

2.1 Business Analyst

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



Model High Level Business Processes

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

Model Requirements

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

Model Business Activities

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

Model Work Flow

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

Display System Behavior

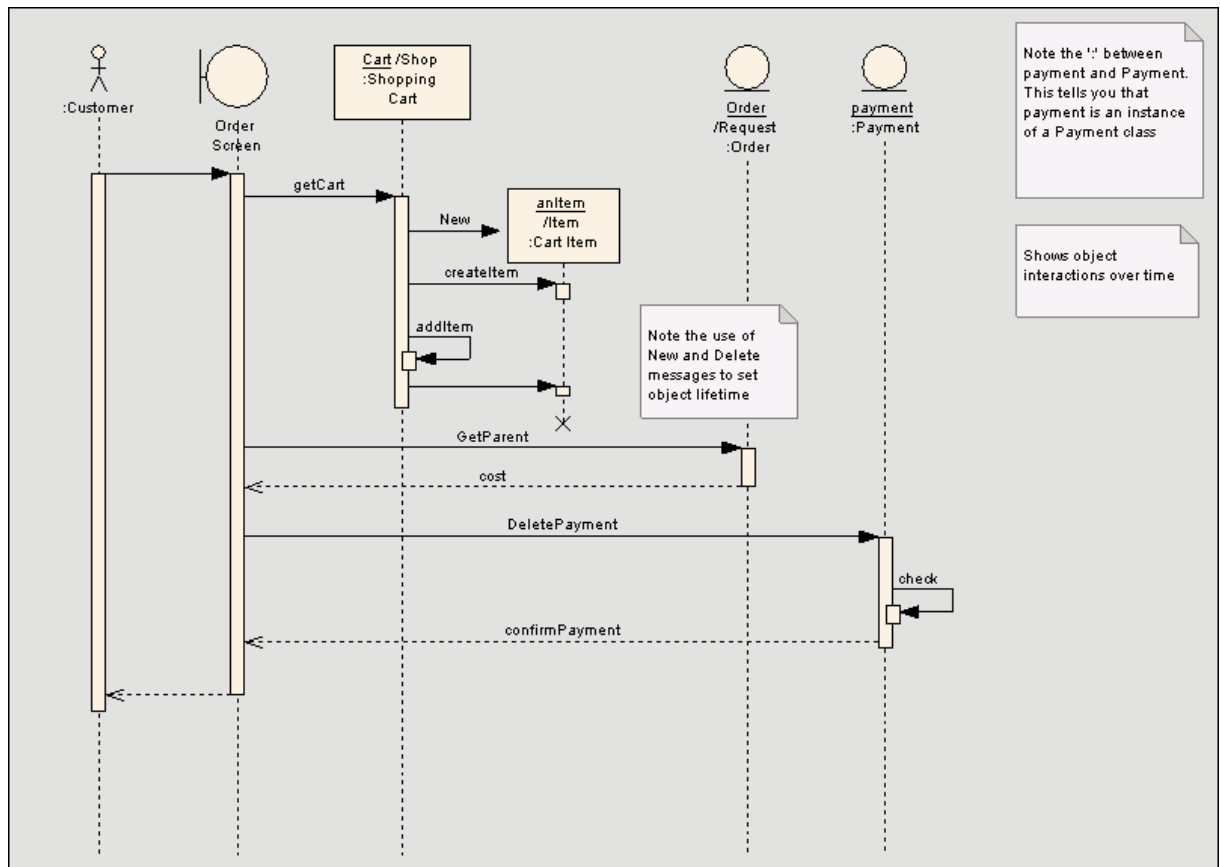
Displaying the behavior of a system as a Use Case gives the Business Analyst an easily understood tool for mapping the functional requirements and behavior of a system.

See:

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

2.2 Software Architects

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



Map functional requirements of the system

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

Map objects in real time

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

Map deployment of objects

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

Detail deliverable components

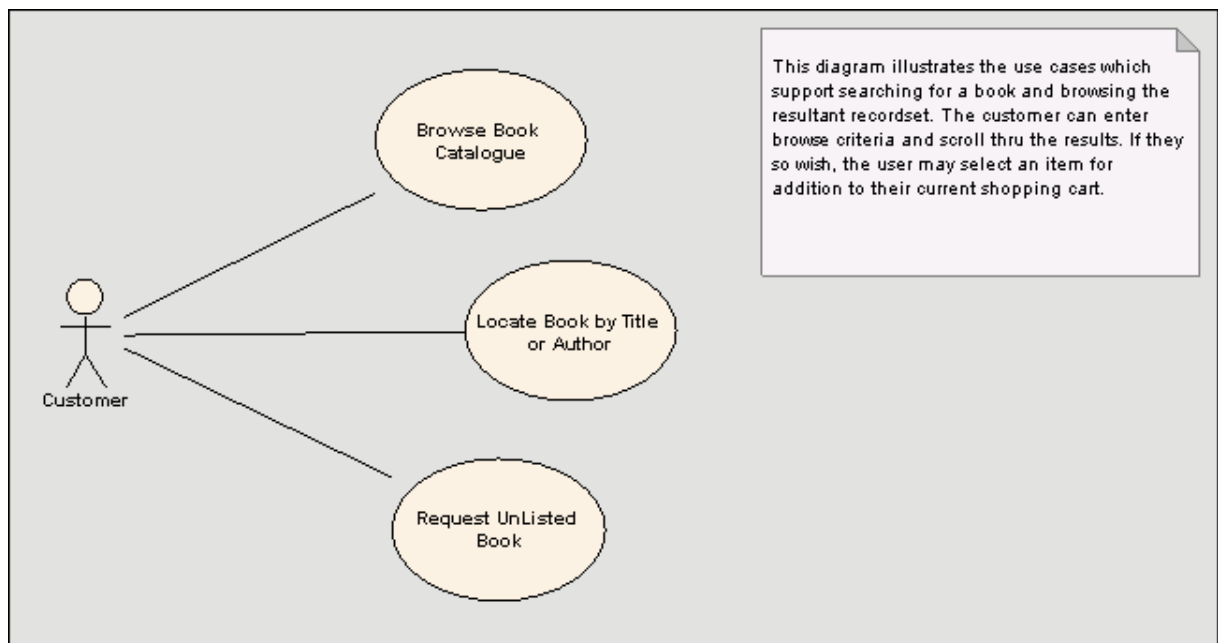
Using Component diagrams allows the Software Architect to model the physical aspects of a system. Components can be executables, libraries, data files or another physical resource that is part of a system. The component model can be developed from scratch from the class model or may be brought in from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

See:

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

2.3 Software Engineer

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



Map Use Cases into Detailed Classes

With Enterprise Architect the Software Engineer can take Use Cases developed by the Software Architect, and create classes which fulfil the objectives defined in the use cases. A class is one of the standard UML constructs which is used to detail the pattern from which objects will be produced at run time.

Detail Interaction between Classes

Interaction Diagrams (Sequence and Communication Diagrams) allow the Software Engineer to model the dynamic design of the system. Sequence Diagrams are used to detail the messages that are passed between

objects and the lifetimes of the objects. Communication Diagrams are similar to Sequence Diagrams, but are used instead to display the way in which objects interact with other objects.

Define System Deployment

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

Define Software Packages

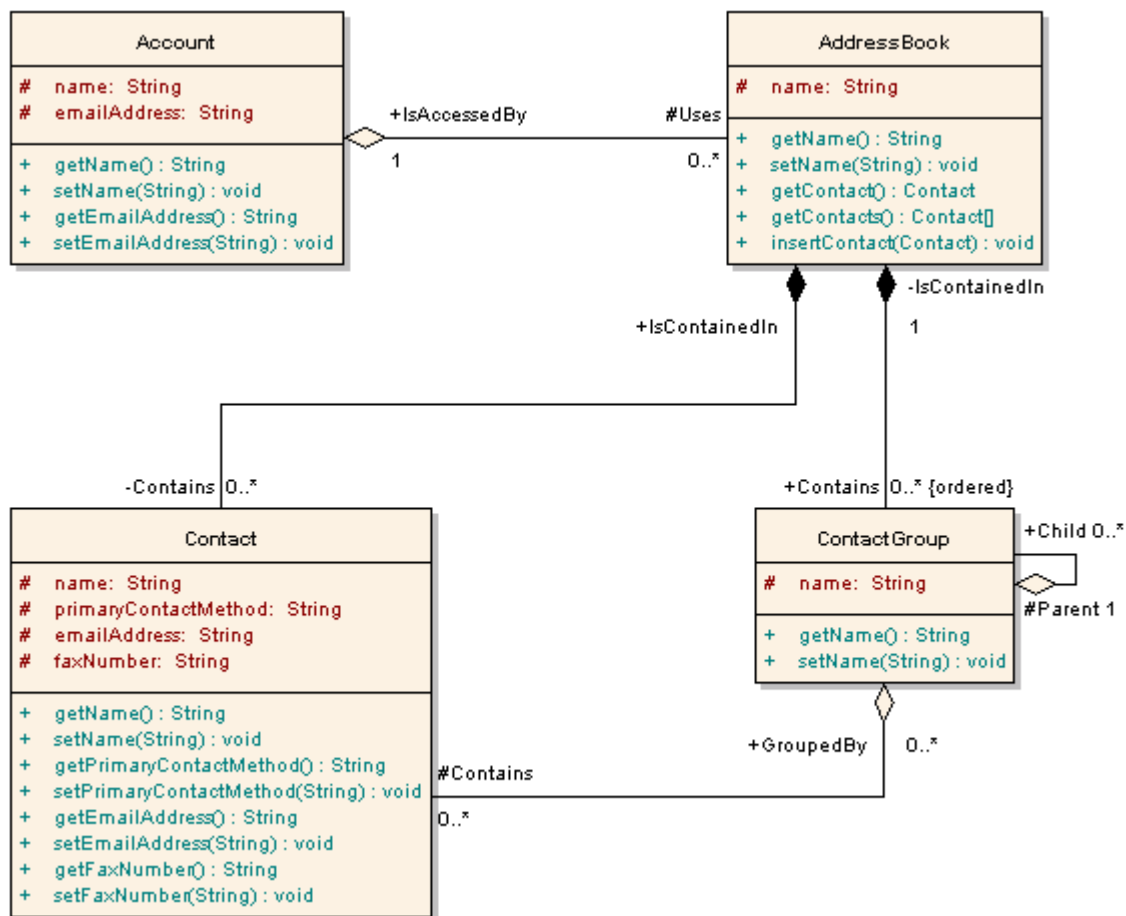
Using Package diagrams allows the Software Engineer to detail the software architecture. Package diagrams are used to organize diagrams and elements into manageable groups declaring the dependencies.

See:

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

2.4 Developer

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



Round trip engineering

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

Reverse engineering

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

Forward engineering

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

Determine the system state

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

Visualize package arrangement

Package diagrams are used to help design the architecture of the system. They are used to organize the

diagrams and elements in manageable groups, and to declare their dependencies.

Follow the flow of code

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

See:

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

2.5 Project Manager

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

Technical Complexity Factors

Set up Technical factors for estimation

Factor Number: Description: Weight: Assigned Value:

TCF04	Complex internal processing	1.00	4.00	
-------	-----------------------------	------	------	--

Defined Technical Types

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

Unadjusted TCF: 47.00

Provide project estimates

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

Resource management

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

Risk management

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

Maintenance

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

See:

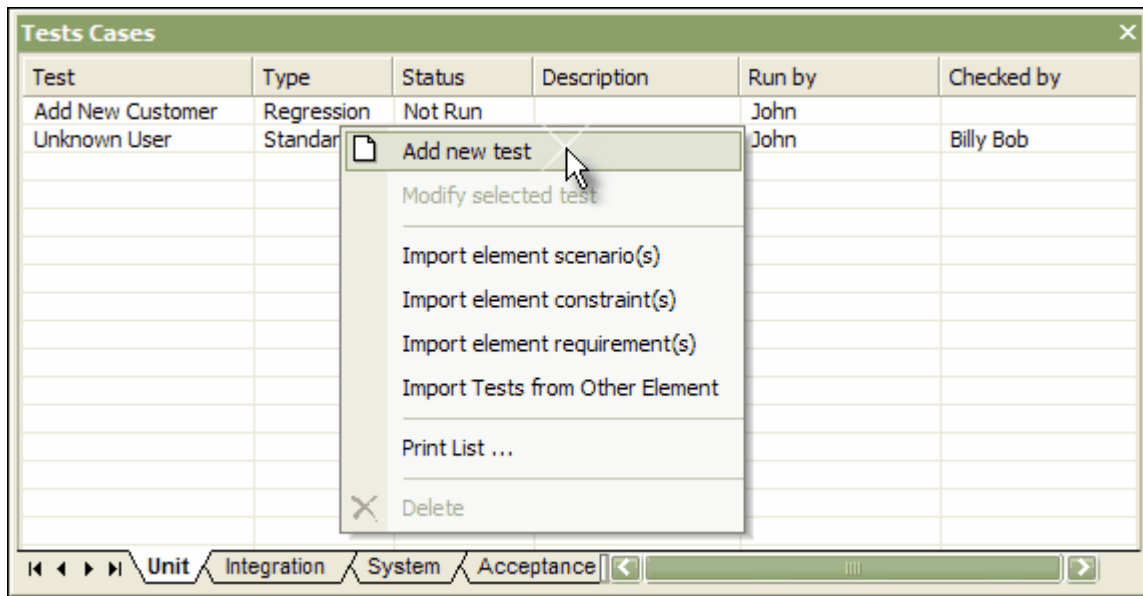
- [Estimation](#)
- [Resources](#)
- [Testing](#)
- [Life Cycle](#)
- [Changes and Defects](#)
- [Model Tasks List](#)
- [Project and Model Issues](#)
- [Model Glossary](#)
- [Update Package Status](#)
- [Manage Bookmarks](#)

2.6 Testers

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

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

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



Test cases

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

Import requirements, constraints and scenarios

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

Create quality test documentation

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

Element defects changes

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

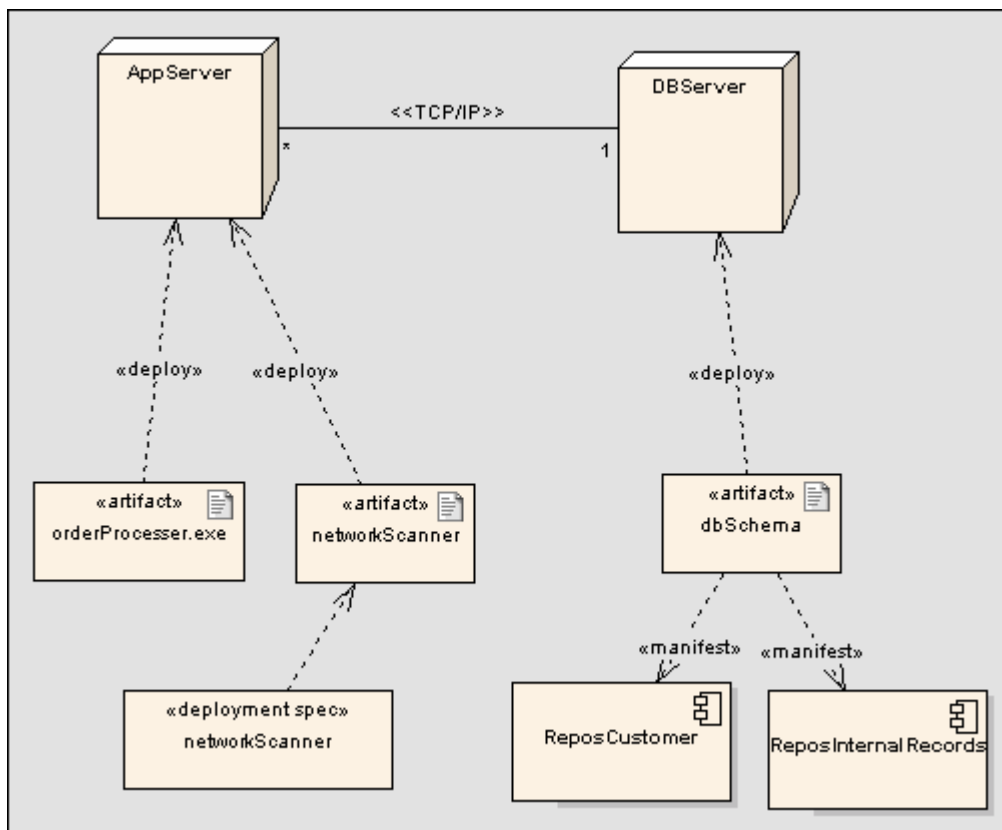
See:

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

- [Show Test Scripts in Compartments](#)
- [Test Documentation](#)

2.7 Deployment and Rollout

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



Display Network Deployment

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

Work Station Deployment

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

Maintenance

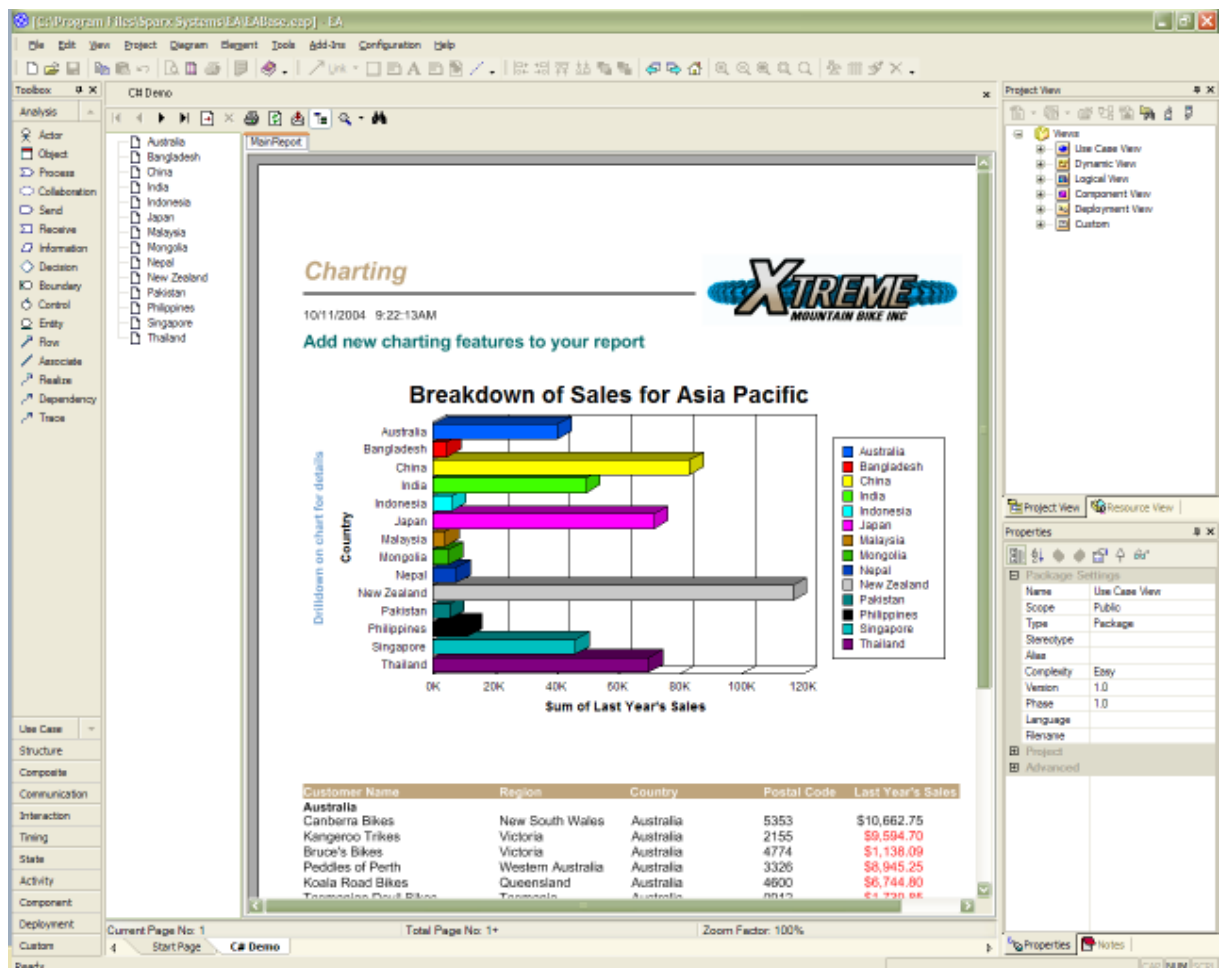
EA allows the user to track and assign maintenance related items to elements within EA. This allows for the rapid capture and record keeping of maintenance tasks such as issues, changes, defects and tasks. By providing a centralized facility for each element involved in the deployment process EA offers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

See:

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

2.8 Technology Developer

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



UML Profiles

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

UML Patterns

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

Code Templates

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

Tagged Values

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

MDG Technologies

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

EA Add-In

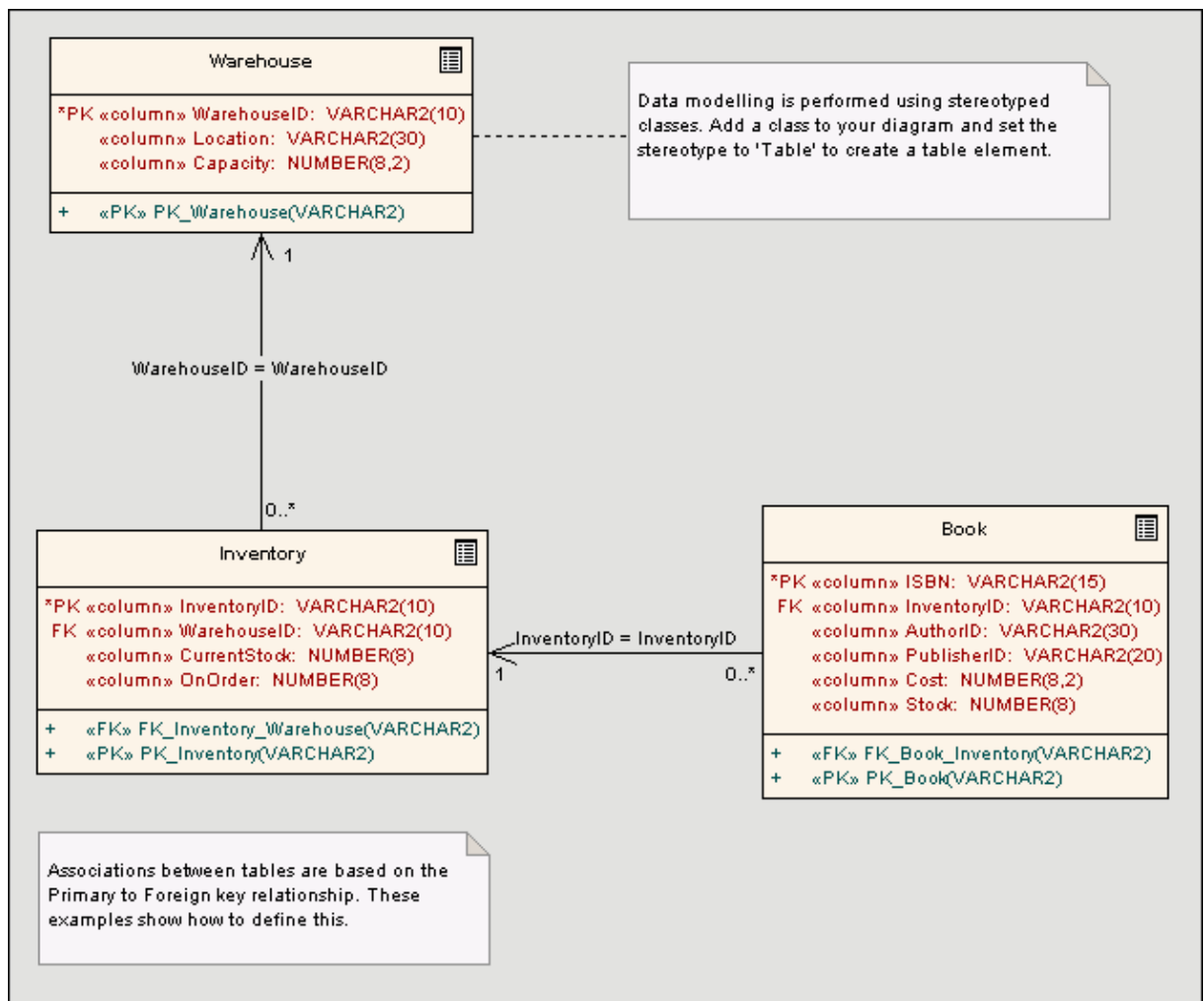
EA Add-Ins allows users to build their own functionality into EA, creating their own mini programs which can extend the capabilities of EA, defining their own menus, and creating their own [Custom Views](#).

See:

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

2.9 Database Administrator

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



Create Logical Data Models

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

Generate Schema

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

Reverse Engineer Database

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

See:

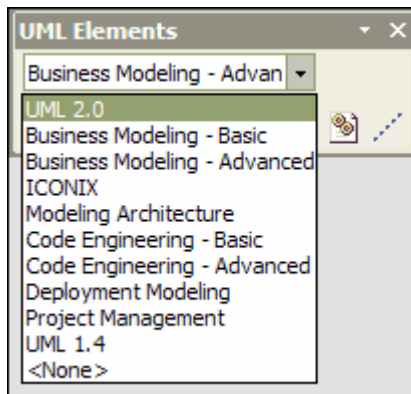
- [Database Schema](#)
- [Creating a Data Model Diagram](#)
- [Creating a Table](#)
- [Setting Properties of a Table](#)
- [Creating Columns](#)
- [Creating Primary Keys](#)

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

2.10 Perspectives

Workspace Perspective

The Workspace perspective tool allows the users of EA to alter the contents of the UML Toolbox to suit a particular modeling role. For example selecting the Code Engineering Basic option will cause the UML toolbox to only include the basic tools used in code engineering and removes the tools which do not apply to code engineering. The Workspace Perspective is selected from the [UML Elements Toolbar](#).



Part



3 Getting Started

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

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

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

See:

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

3.1 Installation

Enterprise Architect is distributed as a single executable setup file (.exe). The Corporate Edition requires additional files and supplementary installation processes if you plan to use the SQL Server, MySQL, PostgreSQL, Sybase Adaptive Server Anywhere or Oracle9i options (see below). Please note that installation and maintenance of these DBMS systems is not covered under the support agreement.

The latest edition of the evaluation and registered versions of EA are always available from [Sparx Systems](#). Access to the registered version requires a username and password to gain access to the registered user area of the web site. These will be provided upon purchase of a license.

Installing EA

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

Corporate Edition users planning to use SQL Server, MySQL, PostgreSQL, Sybase Adaptive Server Anywhere or Oracle9i as their model repository can access scripts that will create the required tables etc for the choice of DBMS. These can be found at one of the following pages:

- The Corporate Edition Resources page at www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.
- The Trial Corporate Edition Resources page at http://www.sparxsystems.com.au/ea_corp_ed.htm

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

3.2 Starting the Application

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

Starting Enterprise Architect

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

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

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

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

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

See also:

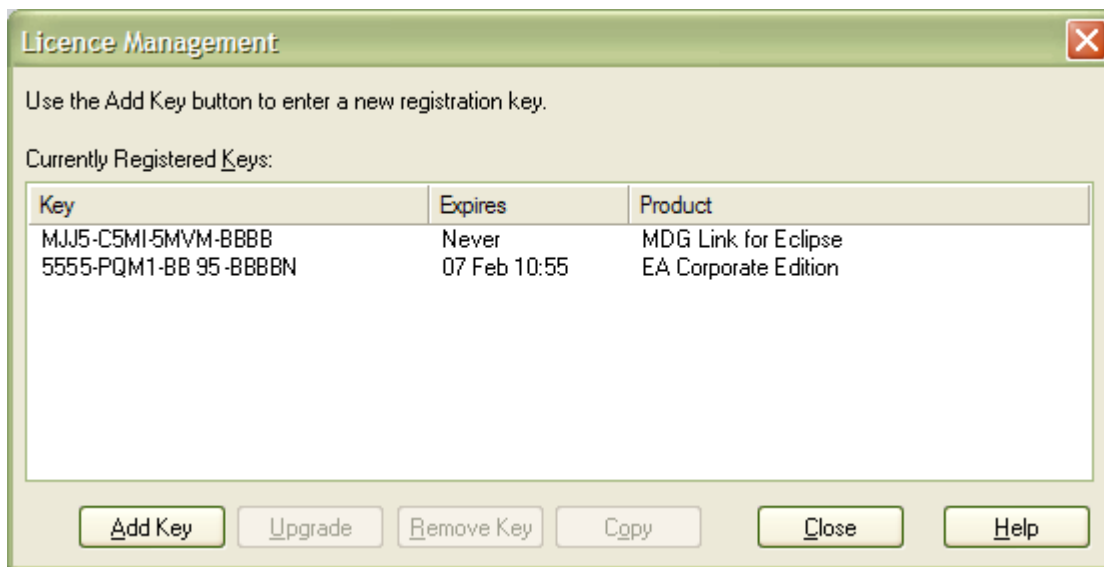
- [Connect to a MySQL Repository](#)
- [Connect to a SQL Server Repository](#)
- [Connect to an Oracle 9i Repository](#)
- [Connect to PostgreSQL Repository](#)
- [Connect to an Adaptive Sever Anywhere Repository](#)
- [Connect to a MSDE Server Repository](#)

3.3 Licence Management

Licence Management

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

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



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

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

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

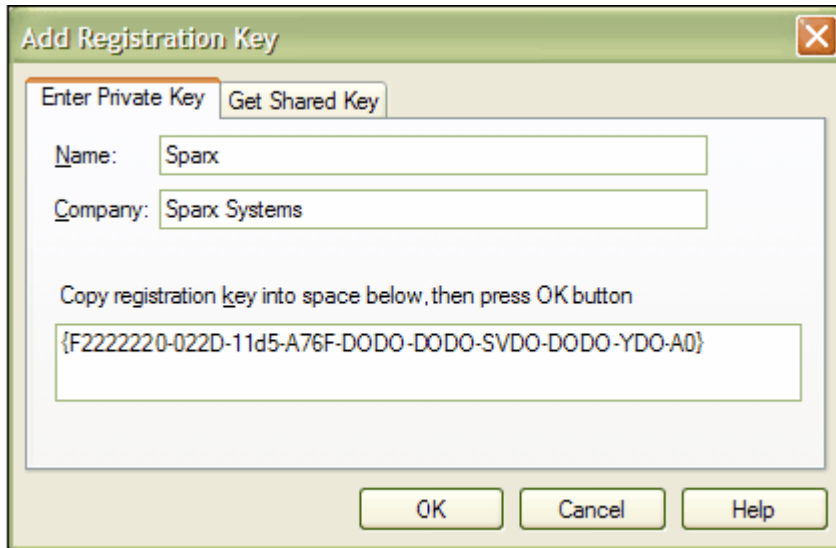
3.3.1 Add License Key

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

Add a Private Key

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

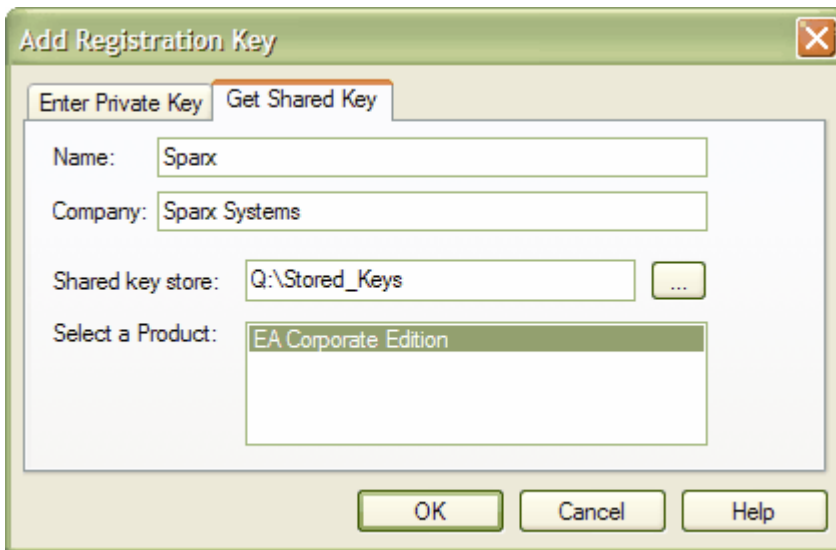
field. Press the **OK** button to confirm the key selection.



The screenshot shows the 'Add Registration Key' dialog box with the 'Enter Private Key' tab selected. The 'Name' field contains 'Sparx' and the 'Company' field contains 'Sparx Systems'. Below these fields, there is a text area with the instruction 'Copy registration key into space below, then press OK button' and a text box containing the key '{F2222220-022D-11d5-A76F-DODO-DODO-SVDO-DODO-YDO-A0}'. At the bottom, there are 'OK', 'Cancel', and 'Help' buttons.

Add a Shared Key

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



The screenshot shows the 'Add Registration Key' dialog box with the 'Get Shared Key' tab selected. The 'Name' field contains 'Sparx' and the 'Company' field contains 'Sparx Systems'. The 'Shared key store' field contains 'Q:\Stored_Keys' and has a browse button (...). The 'Select a Product' list box shows 'EA Corporate Edition' selected. At the bottom, there are 'OK', 'Cancel', and 'Help' buttons.

Note: Shared Keys are only available with a floating license using the Sparx Enterprise Key Store. Shared Keys require EA version 4.51 or higher.

3.4 Registering a Full License

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

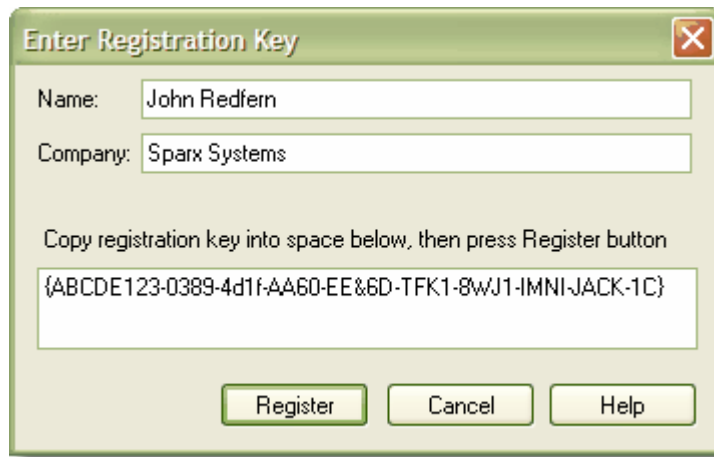
Registering EA

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

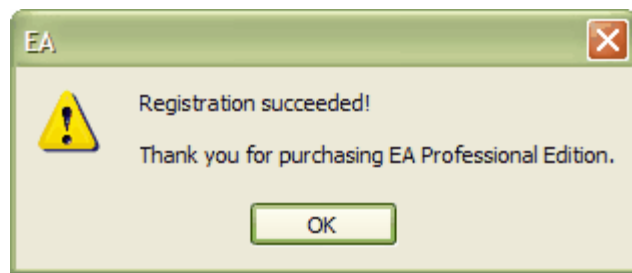
1. Purchase one or more licenses.
2. Once you have paid for a licensed version of Enterprise Architect, you will receive (via email or other suitable means)
 - a license key(s)
 - the address of a web site from which to download the full version
3. Save the license key and download the latest full install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



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



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



See also:

[Upgrading an Existing License](#)

3.5 Upgrade an Existing License

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

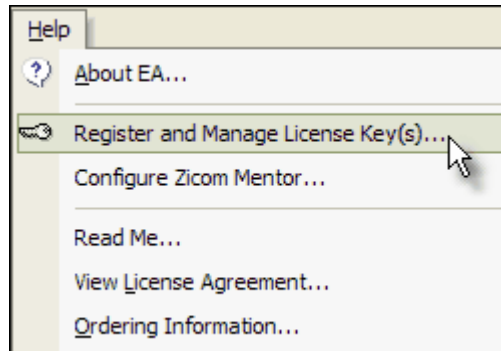
An upgrade key is a special key that will upgrade an existing license to a higher edition. Once you have purchased and received the appropriate key, follow the instructions below to unlock additional features.

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

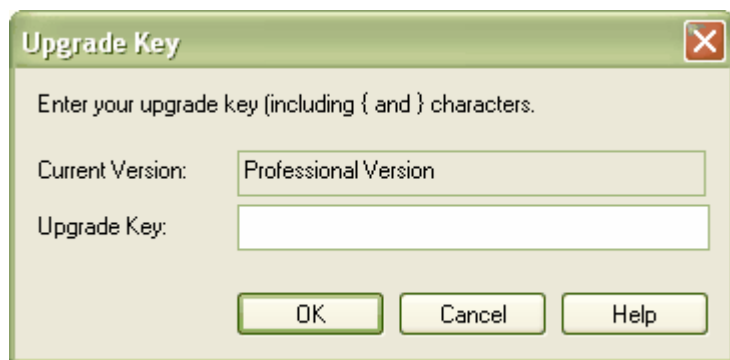
Upgrading EA

Follow these steps to upgrade from one edition to another:

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



4. This will bring up the Licence Management Dialog, press the **Add Key** or the **Upgrade** button to enter a new licence key.
5. If the Add key option was taken the user will be presented with the **Enter Registration Key** dialog (enter the key you received for the upgraded edition of EA - **including the { and } characters** (use copy and paste from an email to avoid typing mistakes).
6. If the Upgrade option was taken the user will be presented with the Upgrade Key dialog (enter the key you received for the upgraded edition of EA - **including the { and } characters** (use copy and paste from an email to avoid typing mistakes).

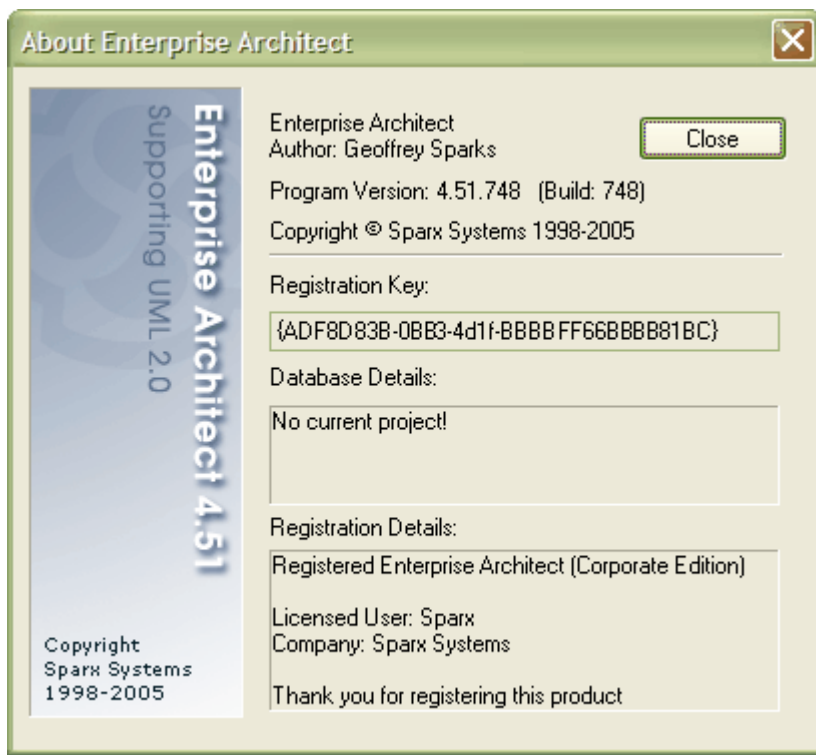


7. Press **OK**. If the key is valid, EA will modify the **Current Version** to reflect the upgrade.
8. Close EA and restart to enable the unlocked features.

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

3.6 Finding Your License Information

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



Part



4 Using Enterprise Architect

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

See:

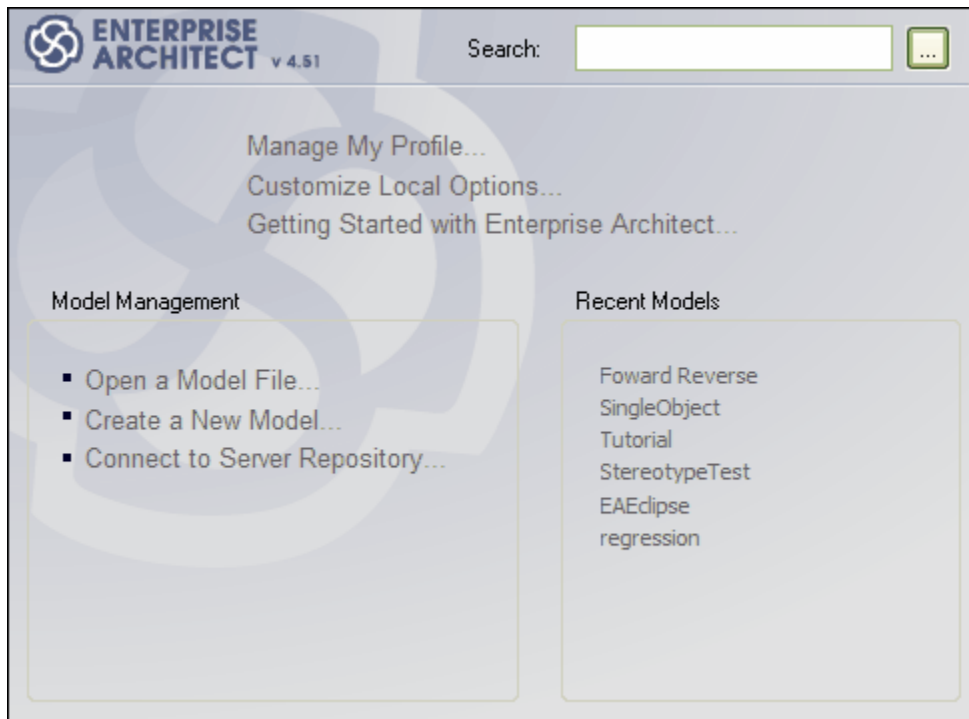
- [The Start Page](#)
- [The Application Workspace](#)
- [Arranging Windows and Menus](#)
- [The Main Menu](#)
- [Workspace Toolbars](#)
- [The Project Browser](#)
- [Dockable Windows Available](#)
- [The UML Toolbox](#)
- [View Options](#)
- [Package Tasks](#)
- [Diagram Tasks](#)
- [Diagram Views](#)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

4.1 The Start Page

When you start Enterprise Architect version 4.5, the first page displayed is the **Start Page** (below). This is a convenient jumping off point for some common activities, such as:

- [Opening a Model File](#) (.EAP file)
- [Create a New Model](#) (.EAP file)
- [Connect to Server Repository](#) (Corporate edition only)

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



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

4.1.1 The Start Page Options

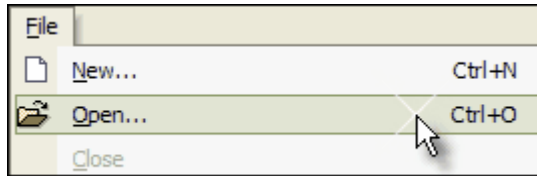
The Start Page gives you the following options which can be accessed using the hyperlinks shown:

Option	Description
Manage My Profile	Opens the <i>Edit My Profile</i> dialog where you can: <ul style="list-style-type: none"> • enter your name • narrow the tools shown in the UML toolbox to a particular subset if you wish • preferred visual layout • access the Custom Tools window • access the Keyboard Accelerator Map
Customize Local Options	Opens the Local Options dialog .
Getting Started with Enterprise Architect	Opens the helpfiles.
Open a Model File	Opens a browse window to let you select a model file to open.
Create a New Model	Opens the <i>Create New Enterprise Architect Project</i> dialog.
Connect to Server Repository	Allows you to specify a Data Source name to connect to. MySQL , PostgreSQL, Sybase Adaptive Server Anywhere SQL Server and Oracle9i repositories are supported. Note: This feature is available in the Corporate edition only.
Recent Models	A list of recently accessed models - click once to open.

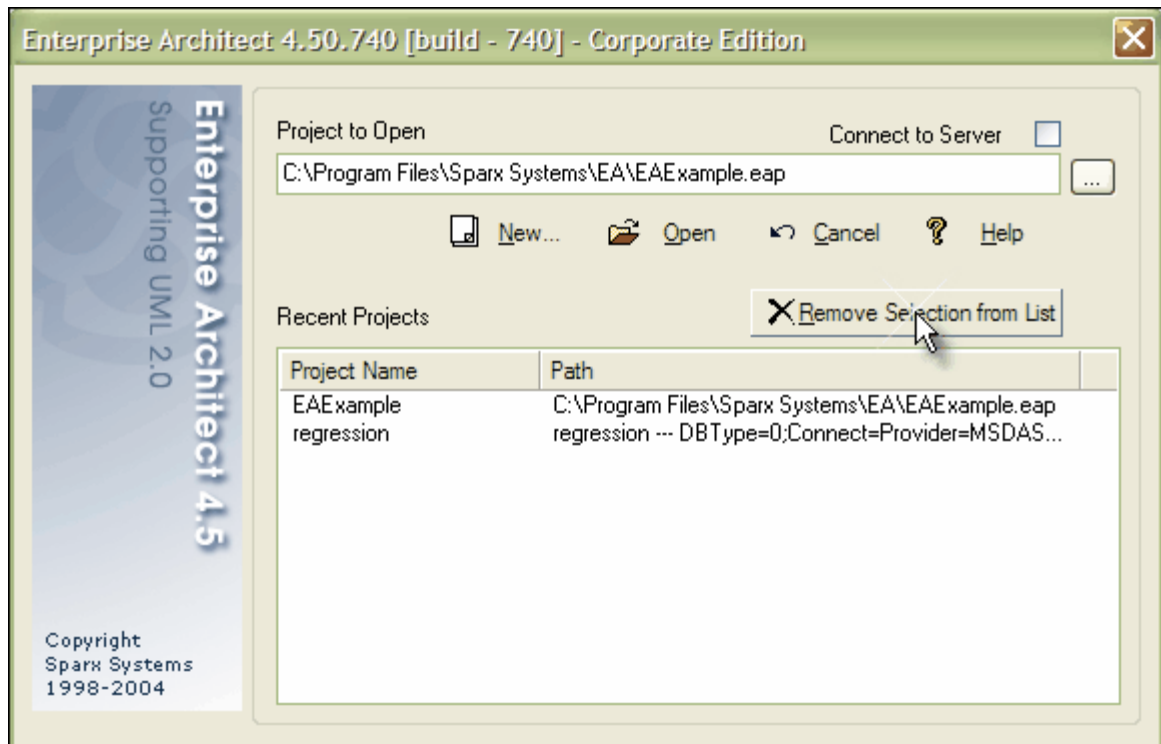
4.1.2 Removing Recent Models

To remove a model link from the Recent Models list of models on the Start Page of Enterprise Architect use the following instructions:

1. Go to the **File** menu and select the **Open** (or press the **Ctrl + O** key combination) option.



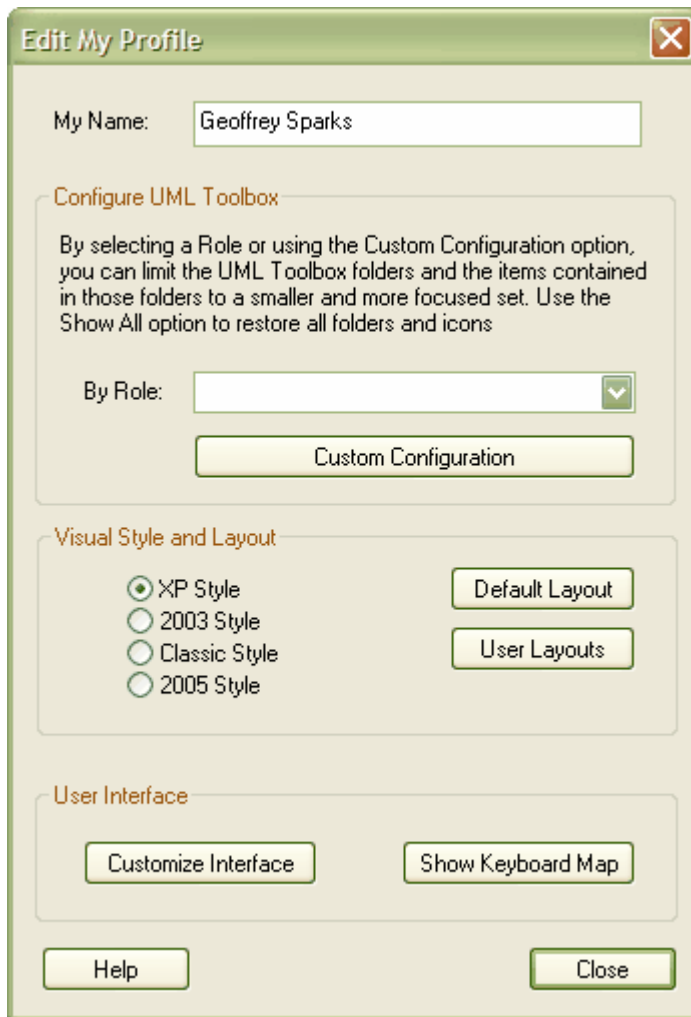
2. In the **Recent Projects** text field highlight the model(s) that you wish to remove from the Recent Models links on the Start Page.
3. Click on the **Remove Selection from List** button.



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

4.1.3 Manage My Profile

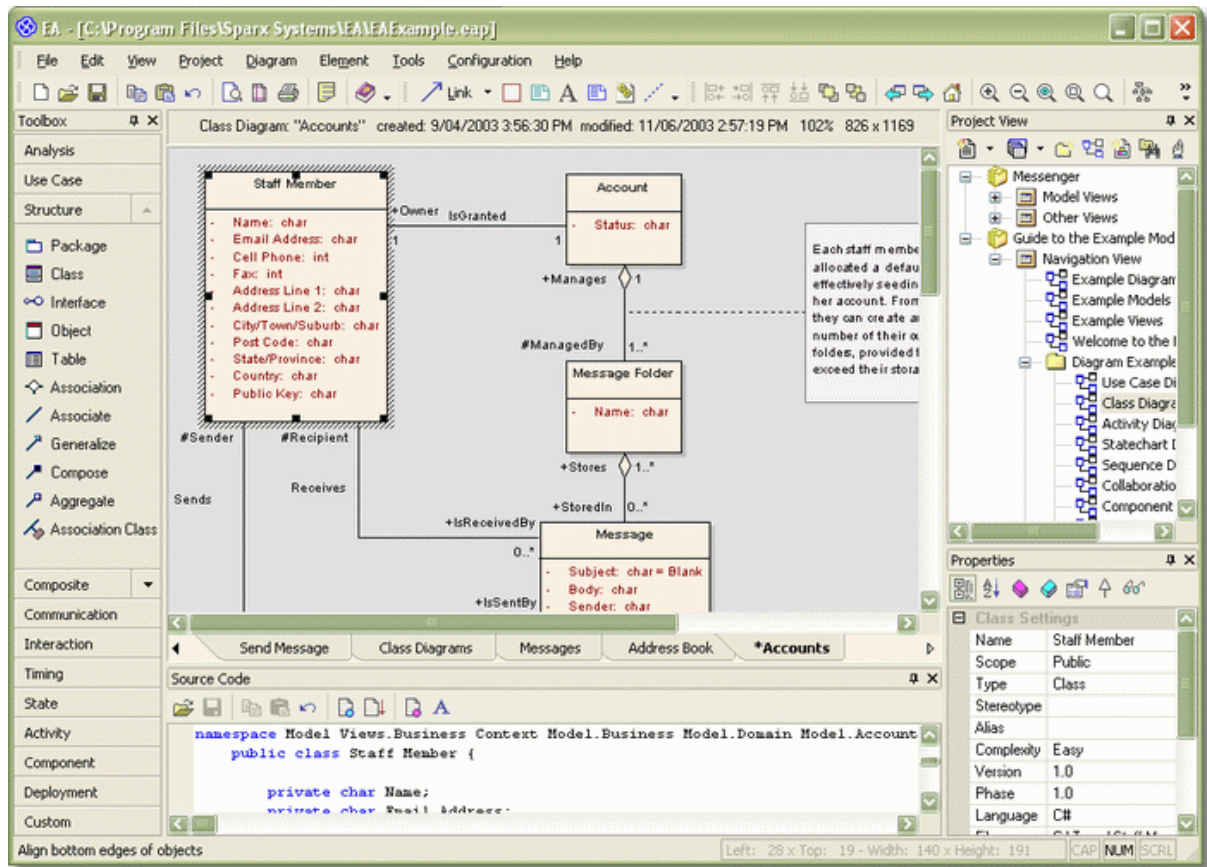
The **Manage My Profile** link opens the **Edit My Profile** dialog, in which you can set the following various options. See the table below.



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

4.2 The Application Workspace

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



Understanding the Application Workspace

The EA application workspace is comprised of the following components:

The Project Browser

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

The Property Browser

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

The Diagram View

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

The Main Menu and Toolbars

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

The UML Toolbox

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

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

4.3 Arranging Windows and Menus

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

See:

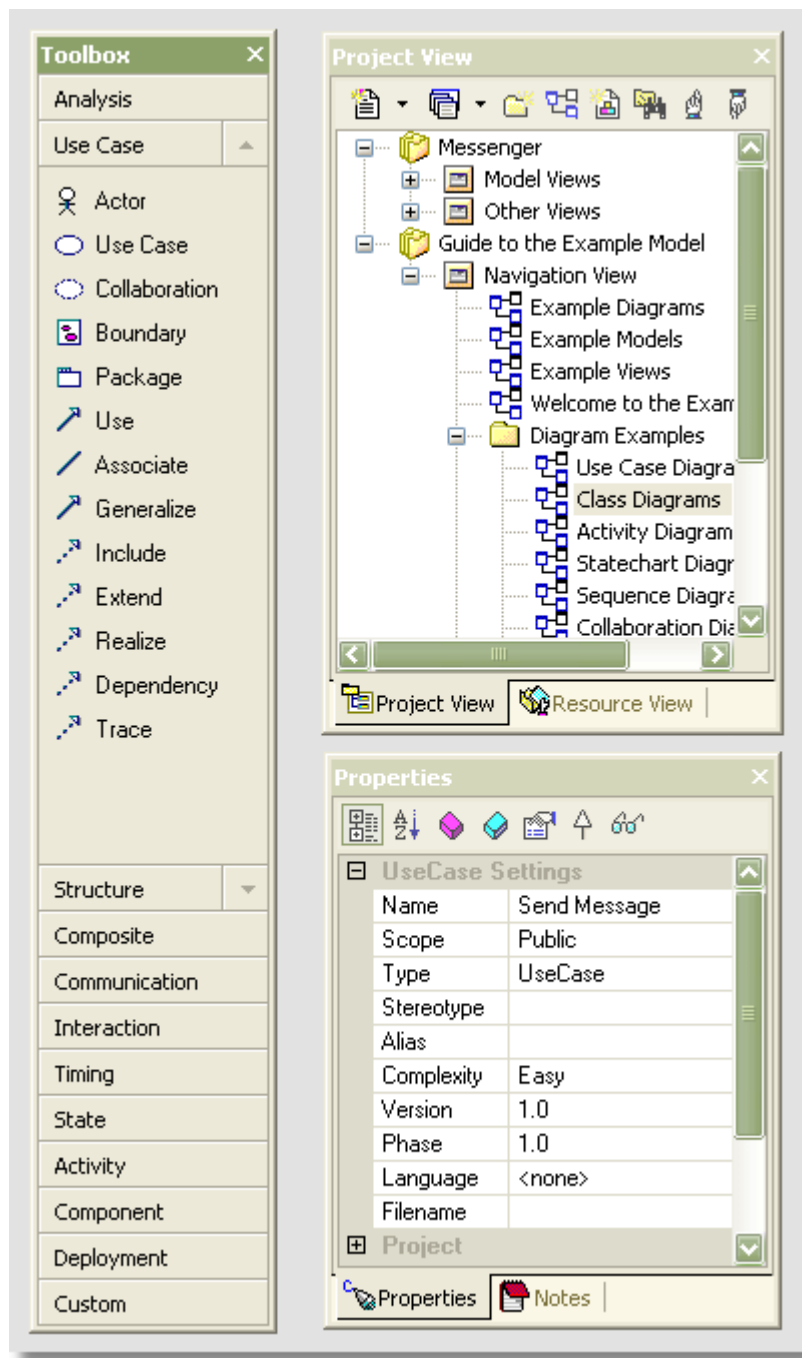
- [Dockable Windows](#)
- [Dockable Windows, 2005 Style](#)
- [Autohide Windows](#)
- [Tear Off Menus](#)

4.3.1 Dockable Windows

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

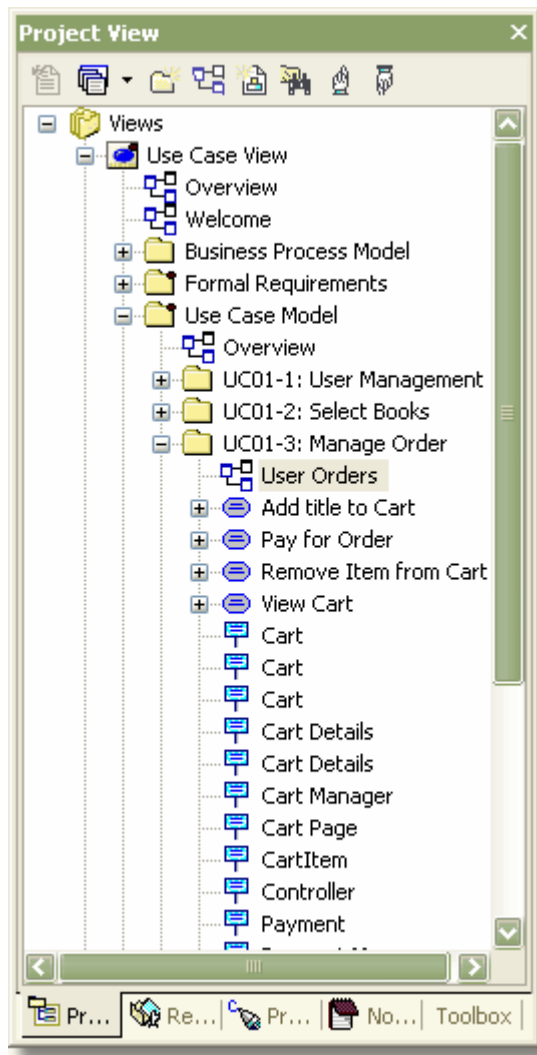
Floating Windows

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

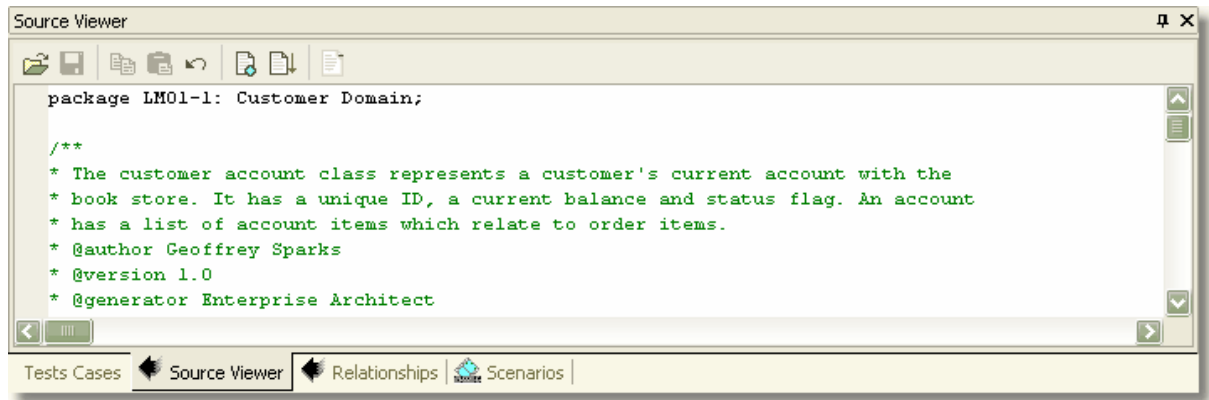


Dock Required Windows into One Frame

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



This can be done with all dockable windows. This next example shows the Test Cases, Source View, Relationships and Scenarios windows all docked into one frame.



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

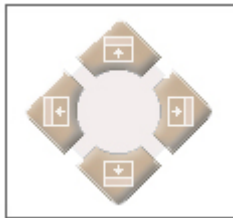
4.3.2 Dockable Window 2005 Style

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

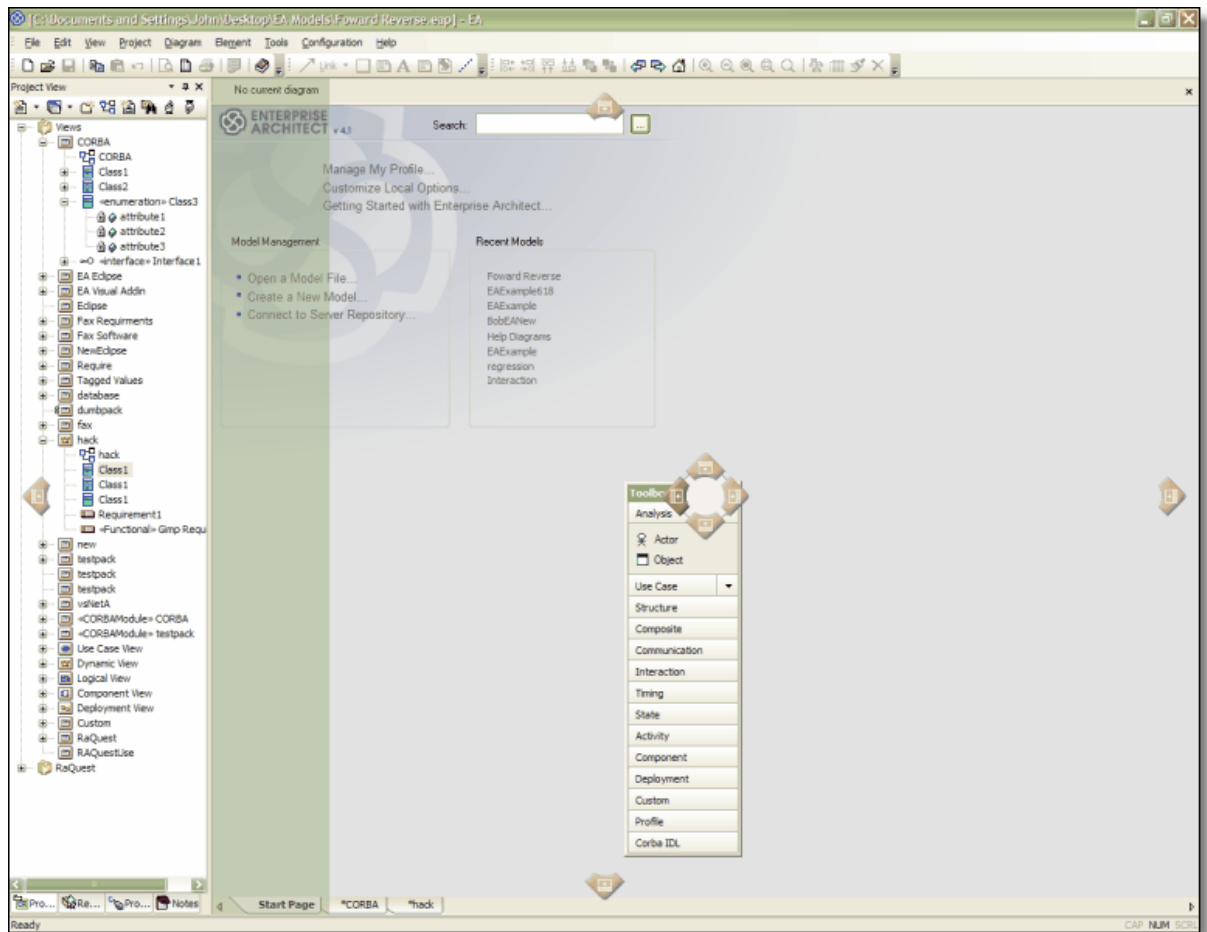
Moving a Window to a new location in EA

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

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



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



Moving a Window into a tabbed windows set

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

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





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

4.3.3 Autohide Windows

There are two ways to autohide your windows:

Autohide Using the Button

You can autohide 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 the  button.

Autohide Using the View Menu

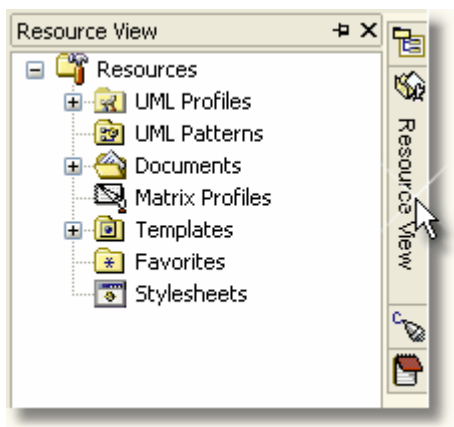
You can autohide all of your currently docked windows at once using the *View | Autohide All Docked Windows* menu option. Alternatively, you can just autohide the currently selected window using the *View | Autohide Active Window* menu option.

Using Automatically Hidden Windows

When you autohide a set of windows in a frame, the menu items contract to the outside of the application workspace as shown below.



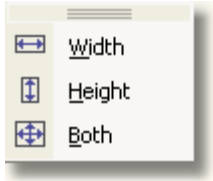
Hover over the icons with your mouse to access the menus.



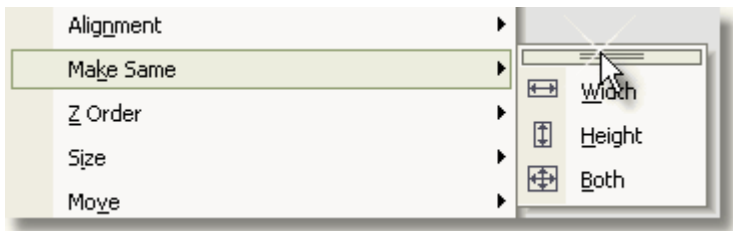
Tip: You can also use the *View | Visual Style | Animate Autohide Windows* option to animate windows which have been automatically hidden.

4.3.4 Tear Off Menus

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



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



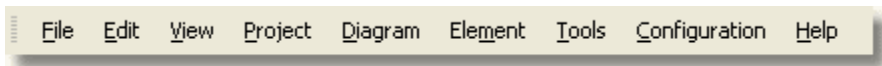
The menu will detach itself as shown here:



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

4.4 The Main Menu

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



In order, the menus available are:

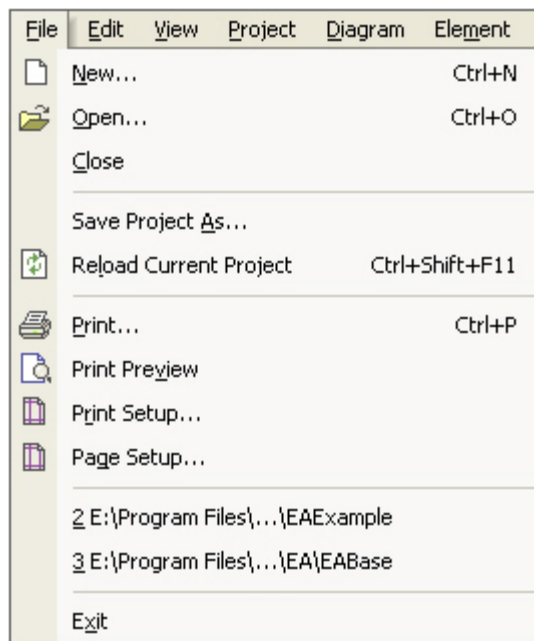
- The [File](#) menu
- The [Edit](#) menu
- The [View](#) menu
- The [Project](#) menu
- The [Diagram](#) menu

- The [Element](#) menu
- The [Tools](#) menu
- The [Configuration](#) menu
- The [Help](#) menu

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

4.4.1 The File Menu

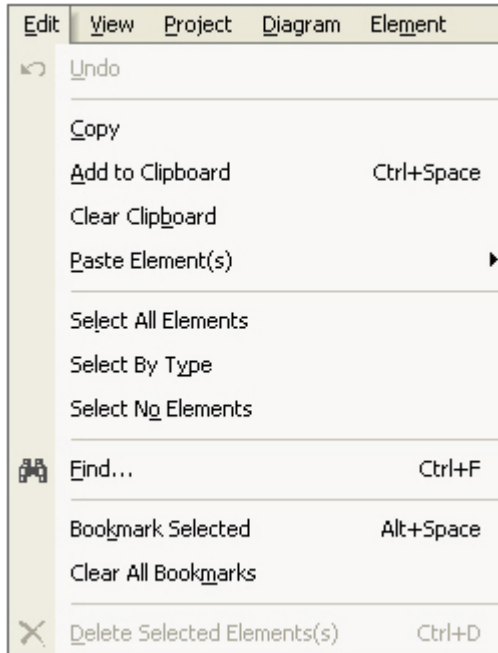
The *File Menu* provides options to create, open, close and save projects, and also to perform print tasks.



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

4.4.2 The Edit Menu

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



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

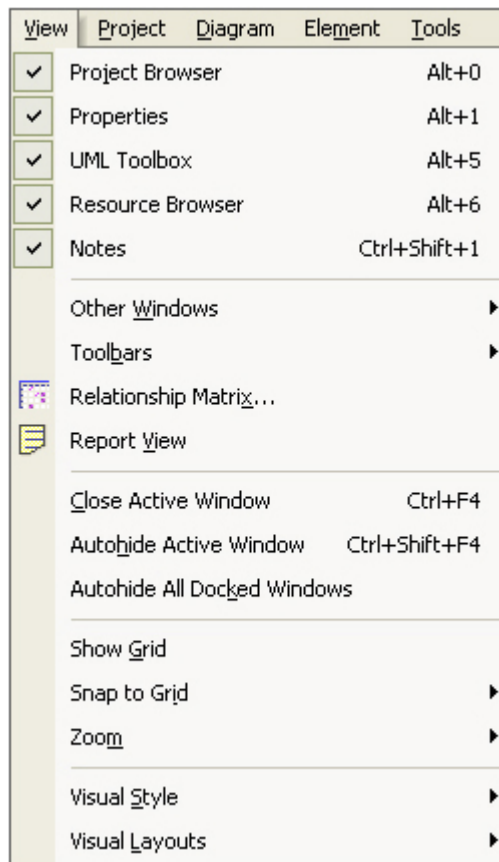
The Paste Elements Sub-Menu

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

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

4.4.3 The View Menu

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



Element	Functionality
Project Browser	Check to show the Project Browser, uncheck to hide it. [Alt+0]
Properties	Check to show the Properties window, uncheck to hide it. [Alt+1]
UML Toolbox	Check to show the UML Toolbox, uncheck to hide it. [Alt+5]
Resource Browser	Check to show the Resource Browser, uncheck to hide it. [Alt+6]
Notes	Check to show Notes, uncheck to hide them. [Ctrl+Shift+1]
Other Windows	See explanation below.
Toolbars	See explanation below.
Relationship Matrix	Open the relationship matrix to cross reference elements to each other by connection type.
Report View	Display the current diagram in a Microsoft Outlook style report list.
Close Active Window	Close the window which currently has focus. [Ctrl+F4]
Autohide Active Window	Autohide the window which currently has focus. [Ctrl+Shift+F4]
Autohide All Docked Windows	Autohide all windows that are docked.
Show Grid	Check to show the grid, uncheck to hide it.
Snap to Grid	See explanation below.
Zoom	Change the zoom factor on the current diagram - see below.
Visual Style	See explanation below.
Visual Layouts	See explanation below.

The Other Windows Sub-Menu

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

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

- [System window](#) [[Alt+2](#)]
- [Testing window](#) [[Alt+3](#)]
- [Maintenance window](#) [[Alt+4](#)]
- [Source Code Viewer](#) [[Alt+7](#)]
- [Element Browser](#)
- [Relationships window](#) [[Ctrl+Shift+2](#)]
- [Rules window](#) [[Ctrl+Shift+3](#)]
- [Hierarchy window](#) [[Ctrl+Shift+4](#)]
- [Scenarios window](#) [[Ctrl+Shift+5](#)]
- [Tagged Values window](#) [[Ctrl+Shift+6](#)]
- [Project Management window](#) [[Ctrl+Shift+7](#)]
- [Output window](#) [[Ctrl+Shift+8](#)]
- [Instant Help window](#) [[Ctrl+Shift+9](#)]

The Toolbars Sub-Menu

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

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

- [Default Tools toolbar](#)
- [Project toolbar](#)
- [Code Generation toolbar](#)
- [UML Elements toolbar](#)
- [Diagram toolbar](#)
- [Current Element toolbar](#)
- [Current Connector toolbar](#)

- [Format Tool toolbar](#)
- [Status Bar](#)
- [Workspace Views toolbar](#)

The Zoom Sub-Menu

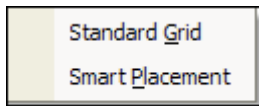
Note: This sub-menu is a [tear off menu](#).

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

The Snap to Grid Sub-Menu

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

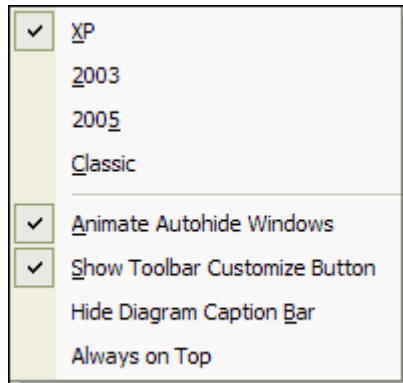
- Standard Grid constrains elements to the grid when they are added to diagrams.
- Smart Placement places elements even distances away from other elements and spaces elements evenly.
- If the Standard Grid or the Smart Placement options are not enabled the elements can be placed freely on the diagram.



The Visual Style Sub-Menu

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

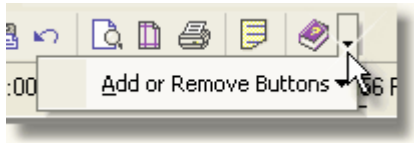
- A 'flat XP' look and feel
- The 2003 style look and feel
- The modern 2005 look and feel
- Classic Windows look and feel



You can also use the [Animate Autohide Windows](#) option to animate windows which have been [automatically hidden](#).

The [Always on Top](#) option ensures that the topmost status is preserved between sessions.

The [Show Toolbar Customize Button](#) option toggles the button on the end of the toolbar that allows you to customize the toolbar buttons, as shown below:



The Visual Layouts Sub-Menu

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

See also: [Custom Layouts](#)

4.4.4 The Project Menu

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



Menu Item	Functionality
New Package	Create a new package.
Issues	Add, edit or delete a project issue .
Glossary	Edit the project glossary .
Namespaces	Locate and delete model namespaces.
Documentation	See below.
Source Code Engineering	See below.
Database Engineering	See below.
Generate XML Schema	Use the XML Schema Generation facility.
Generate MDG Technology File	See Creating MDG Technologies .
Security	See below.
Version Control	See below.
Import/Export	See below.
Get project custom colors	Get the custom colors used in the Appearance dialog .
Set project custom colors	Set the custom colors used in the Appearance dialog .
Set Element Template Package	Configure or change the default template directory.
Use Case Metrics	Set Use Case Metrics to assist in estimating project size.
View Project Statistics	View some basic project statistics.

The Documentation Sub-Menu

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

Generate documentation of various types.

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

The Source Code Engineering Sub-Menu

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

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

Element	Description
Generate Current Element	Generate source code for the currently selected element. [Ctrl+G or F11]
Synchronize Current Element	Synchronize selected class with source code. [Ctrl+R or F7]
Batch Generate Selected Element(s)	Batch generate source code for the currently selected element(s). [Shift+F11]
Batch Synchronize Selected Element(s)	Batch synchronize the currently selected element(s) with source code. [Ctrl+Shift+F11]
Generate Package Source Code	Generate source code for the currently selected package.
Update Package Contents	Synchronize a directory tree.
Import Directory Structure	Reverse engineer an entire directory structure. [Ctrl+Shift+U]
Import C# Files	Import code written in the C# programming language with the file extension <code>.CS</code> .
Import C++ Files	Import code written in the C++ programming language with the file extension <code>.H</code> , <code>.HPP</code> , or <code>.HH</code> .
Import Java Files	Import code written in the Java programming language with the file extension <code>.JAVA</code> .
Import Delphi Files	Import code written in the Delphi programming language with the file extension <code>.PAS</code> .
Import VB.Net Files	Import code written in the VB.Net programming language with the file extension <code>.VB</code> .
Import Visual Basic Files	Import code written in the Visual Basic programming language with the file extension <code>.FRM</code> , <code>.CLS</code> , <code>.BAS</code> or <code>.CTL</code> .
Import PHP Files	Import code written in PHP with the file extension <code>.PHP</code> , <code>.PHP4</code> , <code>.INC</code> .

The Database Engineering Sub-Menu

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

The Security Sub-Menu

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

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

Configure security settings for your project.

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

The Version Control Sub-Menu

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

The Import/Export Sub-Menu

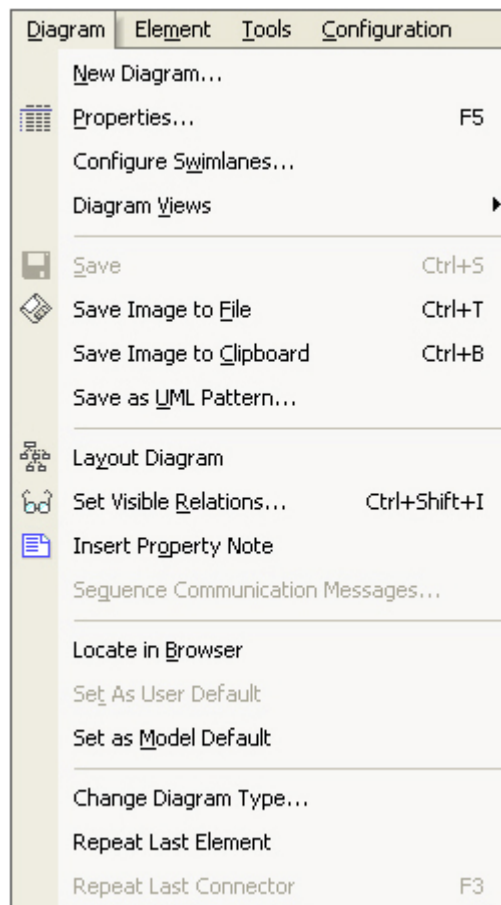
Note: This sub-menu is a [tear off menu](#).

Perform import and export to XMI and CSV.

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

4.4.5 The Diagram Menu

The [Diagram Menu](#) allows the user to save diagram images to file as well as configure diagram properties and options.



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

The Diagram Views Sub-Menu

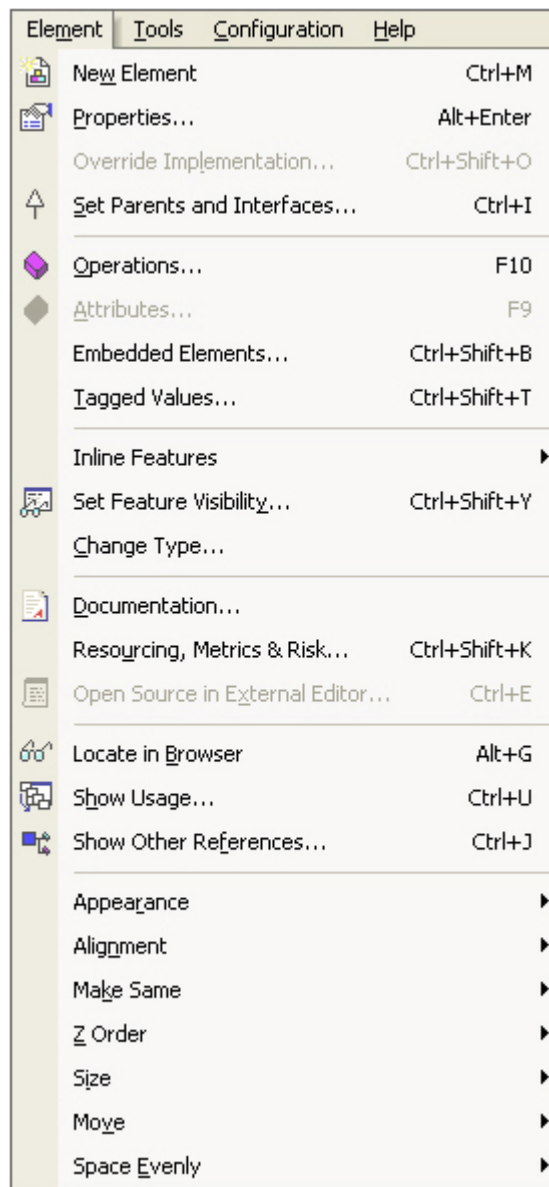
Gives the user the option to view and save diagrams:

- [Close Current](#)
- [Close All Except Current](#)
- [Reload Current](#)
- [Save All Modified](#)
- [Close All](#)

See also: [Diagram Tabs](#)

4.4.6 The Element Menu

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



Element	Description
New Element	Create a new element on the current diagram. [<i>Ctrl+M</i>]
Properties	View the properties window of the selected element. [<i>Alt+Enter</i>]
Override Implementation	Automatically override methods from parent classes and from realized interfaces.
Set Parents and Interfaces	Manually set an element's parent or an interface it realizes. [<i>Ctrl+I</i>]
Operations	View and edit the operations for the selected element. [<i>F10</i>]
Attributes	View and edit the attributes for the selected element. [<i>F9</i>]
Tagged Value	Add a tagged value to the currently selected element.
Inline Features	See below.
Set Feature Visibility	Set various states relating to the selected element(s) visibility.
Change Type	Change the element type of the selected element.
Documentation	Generate a report for the currently selected package in rich text format .
Resourcing, Metrics & Risk	Configure resource allocation, effort, risk and metrics. [<i>Ctrl + Shift + K</i>]
Open Source in External Editor	Open the source code of the selected class in the default external editor for that language. (Source code must have been generated , and the selected element must be a class.) [<i>Ctrl+E</i> or <i>F12</i>]
Locate in Browser	Locate the currently selected element in the Project Browser. [<i>Alt+G</i>]
Show Usage	Display all occurrences of the currently selected element. [<i>Ctrl+U</i>]
Show Other References	Show model element cross references . [<i>Ctrl+J</i>]
Appearance	See below.
Alignment	See below.
Make Same	See below.
Z Order	See below.
Size	See below.
Move	See below.
Space Evenly	See below.

The Inline Features Sub-Menu

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

Menu Option	Hotkey	Description
Edit Selected	<i>F2</i>	Attach a note or attach a constraint to the connection.
View Properties	<i>Enter</i>	Opens up the dialog window containing details of the element.
Insert New After Selected	<i>Ctrl + Shift + Insert</i>	Inserts a new item to the element.
Add Attribute	<i>Ctrl + Shift + F9</i>	Adds an attribute to the element.
Add Operation	<i>Ctrl + Shift + F10</i>	Adds an operation to the element.
Add Other	<i>Ctrl + F11</i>	Allows the user to insert a feature on the specific element item, such as Maintenance features and Testing features.
Delete Selected from Model	<i>Ctrl + Shift + Delete</i>	Deletes the selected item from the model.
	<i>Ctrl + Shift + Arrow key</i>	Navigate Diagram Element Selection, this allows the user to rapidly navigate elements in the diagram, without needing to use the mouse.
	<i>Shift + Enter</i>	Toggle element highlight option on and off

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

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

The Appearance Sub-Menu

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

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

The Alignment Sub-Menu

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

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

The Make Same Sub-Menu

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

The Z Order Sub-Menu

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

The Size Sub-Menu

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

The Move Sub-Menu

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

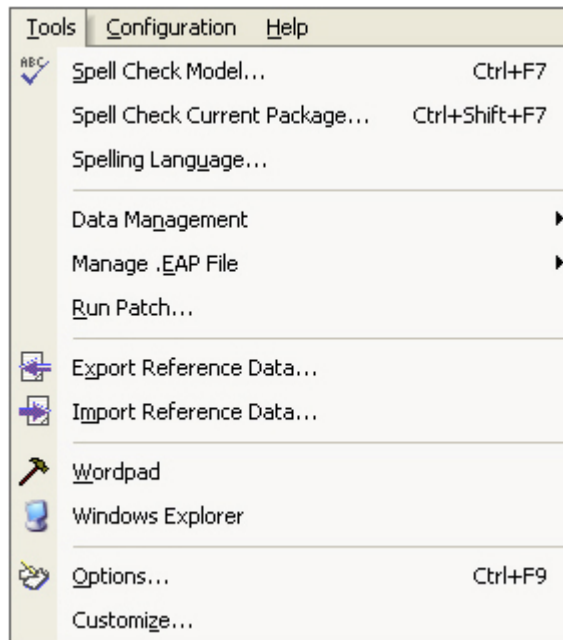
The Space Evenly Sub-Menu

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

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

4.4.7 The Tools Menu

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



Element	Description
Spell Check Model	Spell check the current model. [<i>Ctrl + F7</i>]
Spell Check Current Package	Spell check the current package. [<i>Ctrl +Shift + F7</i>]
Spelling Language	Specify language to use for spell checking.
Data Management	See below.
Manage .EAP File	See below.
Run Patch	Execute a SQL patch .
Export Reference Data	Export reference data to XML files for convenient model updating.
Import Reference Data	Import reference data from XML files for convenient model updating.
Wordpad	Open Wordpad.
Windows Explorer	Open Windows Explorer.
AutInt	Open the Automation Interface program.
Options	Customize your general settings through the local options dialog . [<i>Ctrl + F9</i>]
Customize	Open the Customize window, which allows you to customize commands , toolbars , tools , keyboard , menu and options .

The Data Management Sub-Menu

Manage your project's data.

Element	Description
Data Transfer	Move a complete model from one repository to another.
Data Compare	Compare the total model size of one model and another.
Data Integrity	Check data integrity.

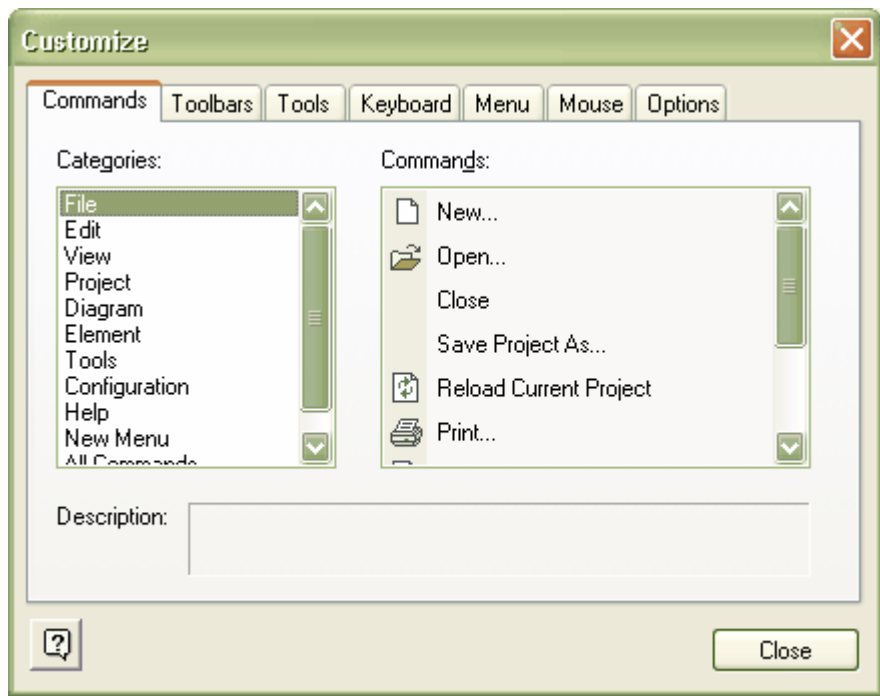
The Manage .EAP File Sub-Menu

Repair, compact or replicate your .EAP file.

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

4.4.7.1 The Customize Window

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



4.4.7.1.1 Customize Commands

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

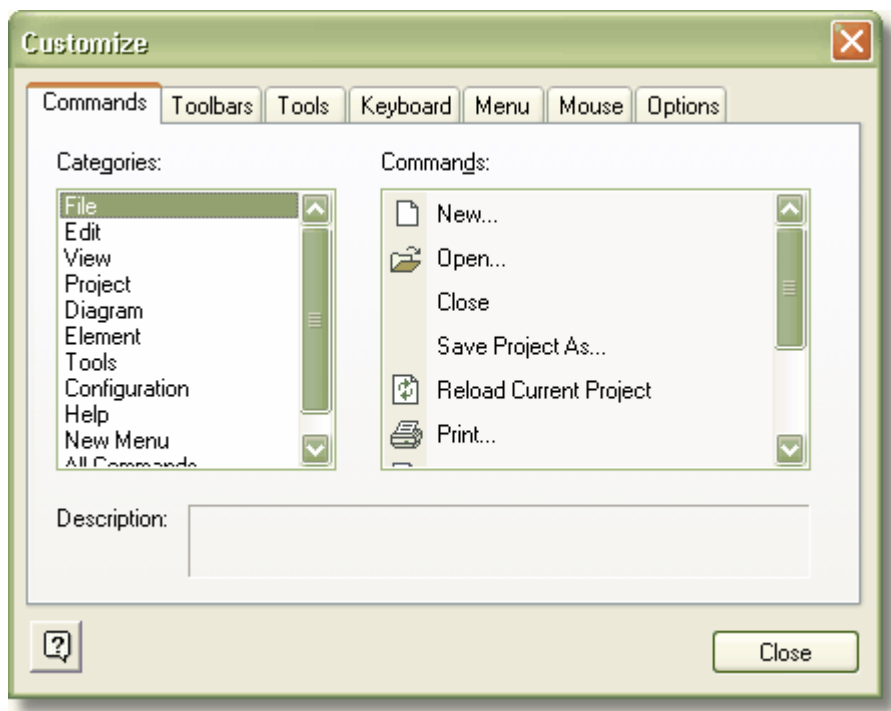
Right-clicking on the command icon in the toolbar while the customize window is open will display a context-sensitive menu. This menu offers options for deleting commands from a toolbar, and for changing the appearance of commands.

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

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

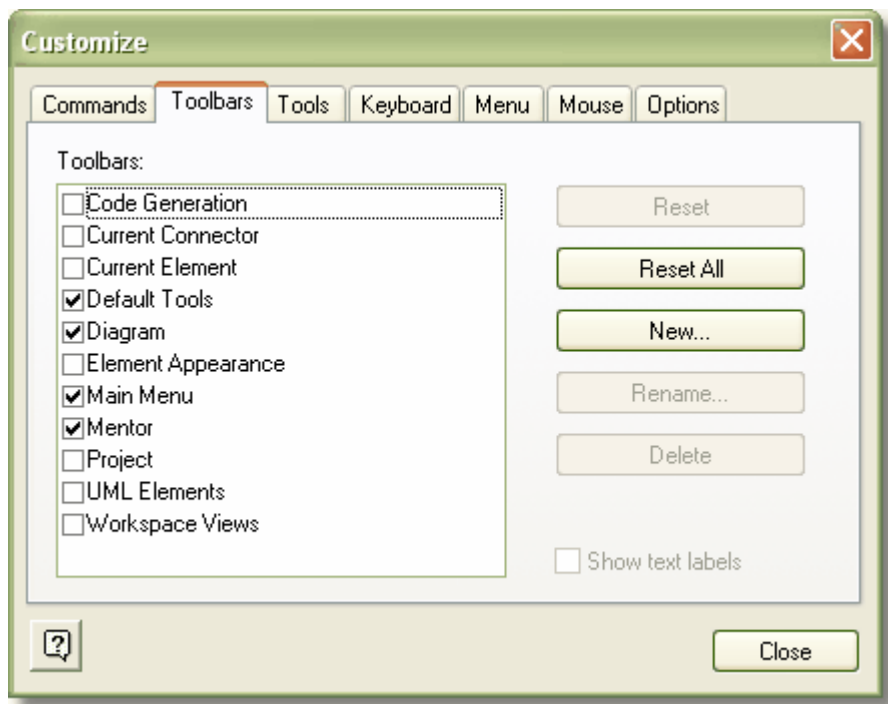
Note: Some commands do not come with a convenient icon, which will result in an empty toolbar button. Either avoid placing these commands on toolbars or use the context-sensitive menu to select an appropriate icon for the command.

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



4.4.7.1.2 Customize Toolbars

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



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

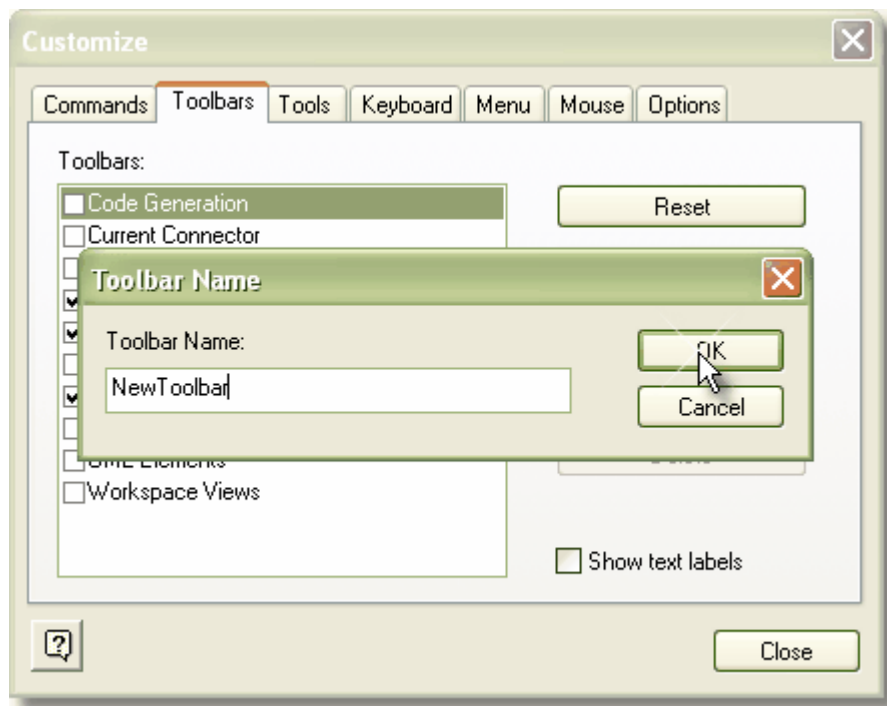
Using the *Toolbars* tab you can:

- Hide or show toolbars by checking the appropriate checkbox.
- Rename toolbars.
- Create new toolbars.
- Delete toolbars.
- Modify toolbar contents by dragging commands from the *Command* tab onto a visible toolbar.
- Reset a toolbar to its default contents and position.

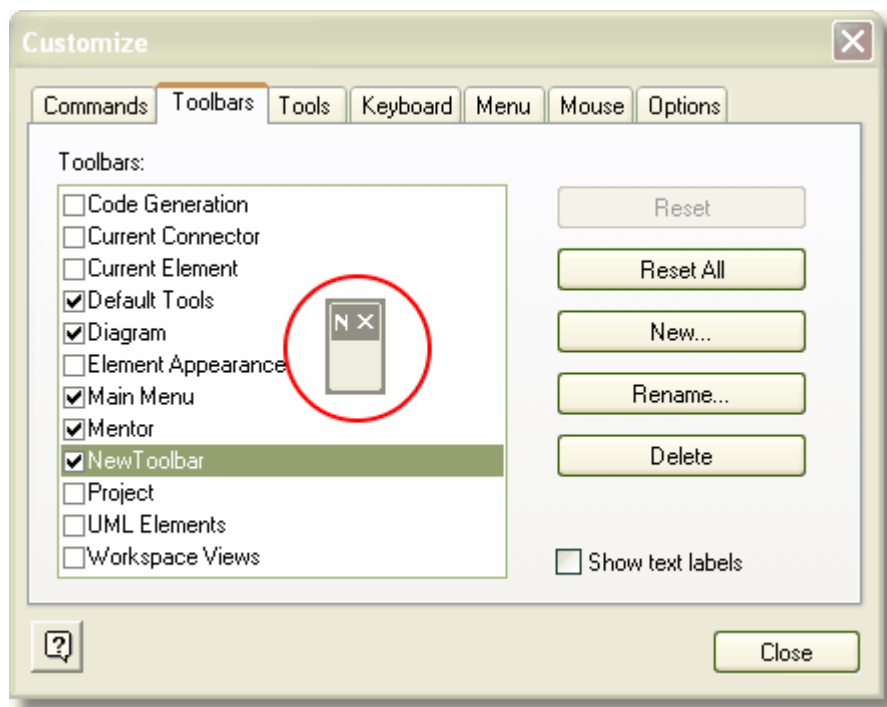
Create a New Toolbar and Populate it with Commands

To create a new toolbar and populate it with commands:

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



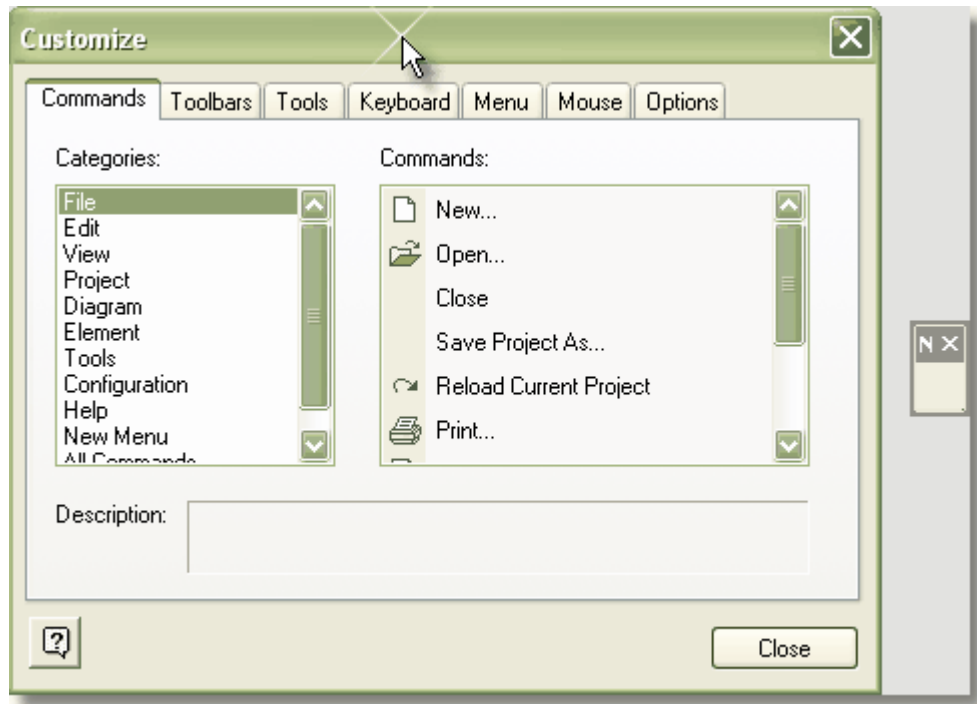
5. Your new toolbar will be created and shown at the front of the *Customize* dialog (see the red circle below).



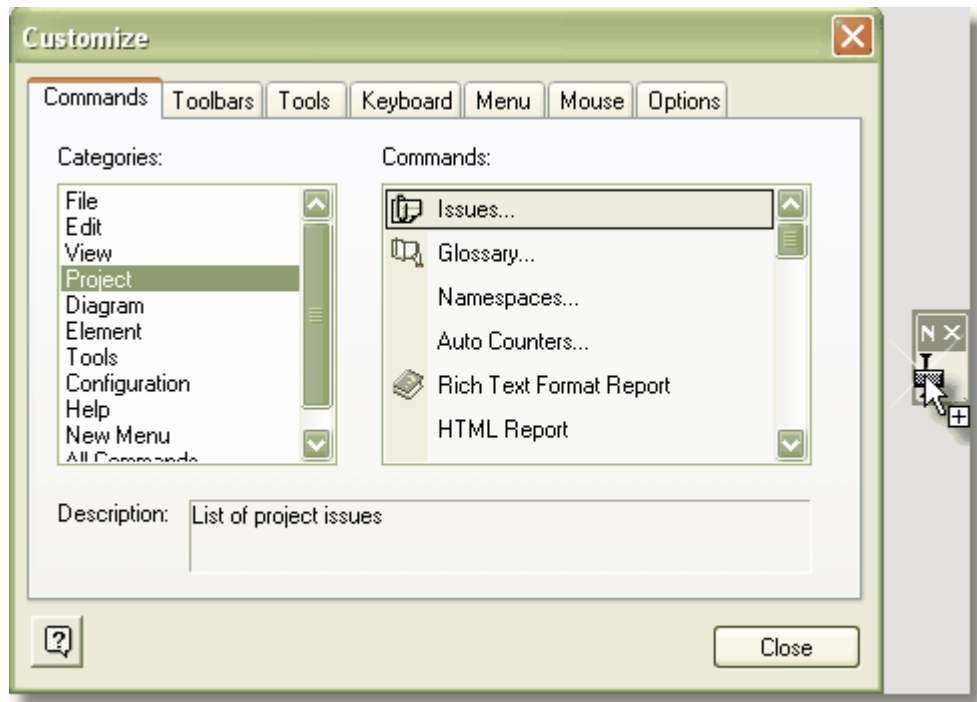
Note: You can check the *Show text labels* checkbox if you wish.

6. Now add commands to your toolbar. Go to the *Commands* tab. This will force your new toolbar behind

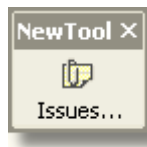
the Customize dialog, so you will need to drag the Customize dialog to the side to find your new toolbar.



7. Find the command you wish to add to your toolbar in the *Commands* list. The *Categories* list on the left represents the EA menu structure and the *Commands* list updates each time you click on a different category.
8. When you have located a command, drag it from the list into the new toolbar.



9. Your toolbar should look like this, if you checked the *Show Text Labels* checkbox:



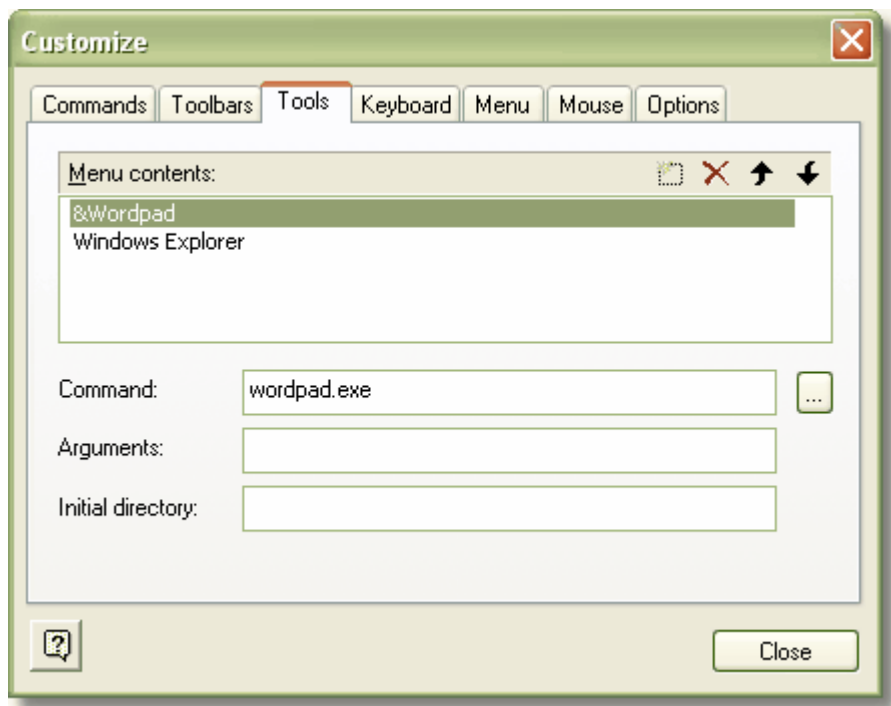
- or this - if you didn't select the *Show Text Labels* checkbox:



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

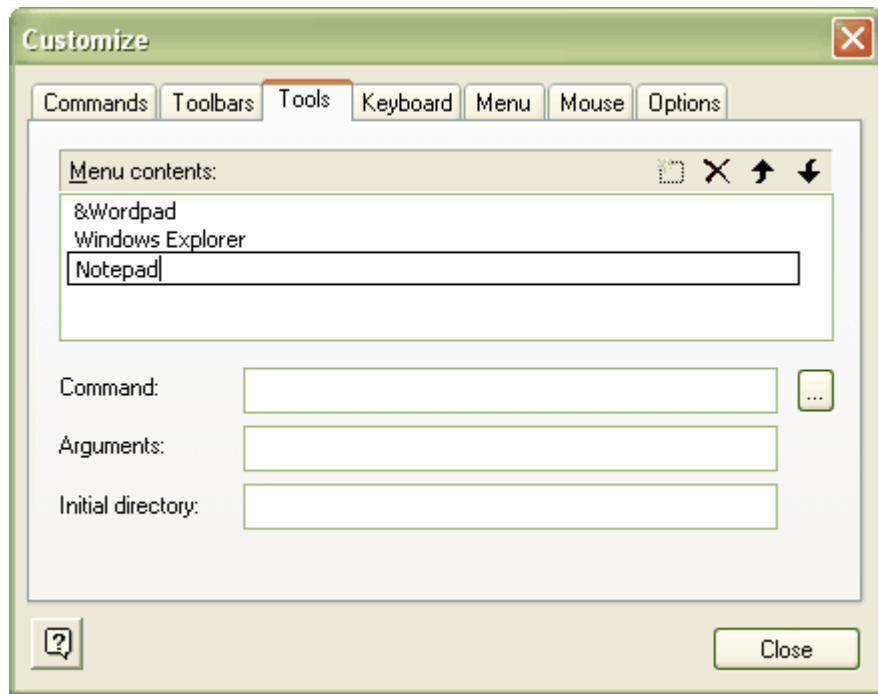
4.4.7.1.3 Custom Tools

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



Add and Configure Custom Tools

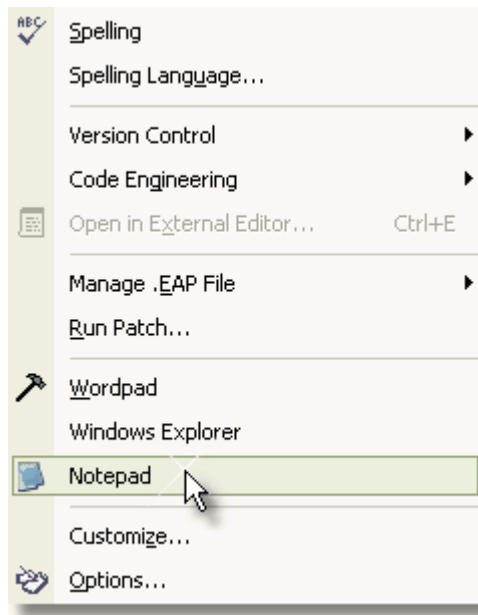
1. From the *Tools* menu, select the *Customize* option.
2. Select the *Tools* tab.
3. Press the *New* symbol (at left of the red 'X') and type in the name of the tool as you want it to appear in the menu.



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

Note: Programs which were installed with your operating system (eg. Notepad, WordPad) do not require a full file path. Programs installed separately (eg. Microsoft Visual Studio) require the full file path in the *Command* textbox. Use the *Browse [...]* button to locate the tool in the file system (see [Using Parameters](#)).

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



4.4.7.1.3.1 Opening External Tools

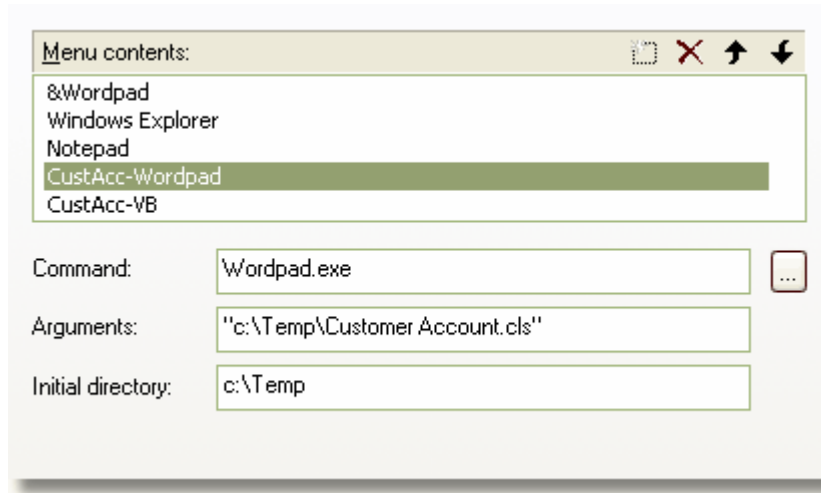
When configuring custom tools in EA, you may specify a file to be opened by the external application.

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

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

Example 1

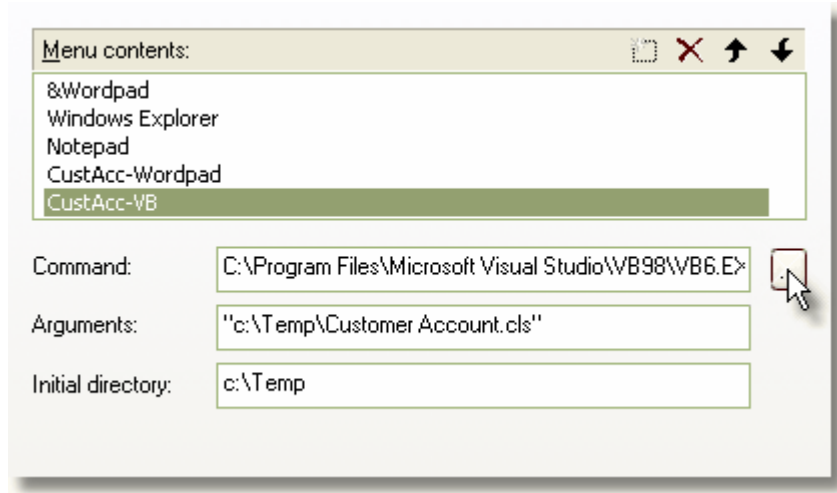
This example opens the file `c:\Temp\Customer Account.cls` using WordPad. If you save from within WordPad the initial directory will be `c:\Temp`.



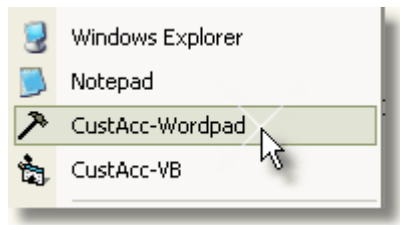
Tip: If there are any spaces in the paths for **Command**, **Argument** or **Initial Directory**, you must enclose the whole path in double quotes. For example: `"c:\Temp\Customer Account.cls"` needs quotes but `c:\Temp\CustomerAccount.cls` does not.

Example 2

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



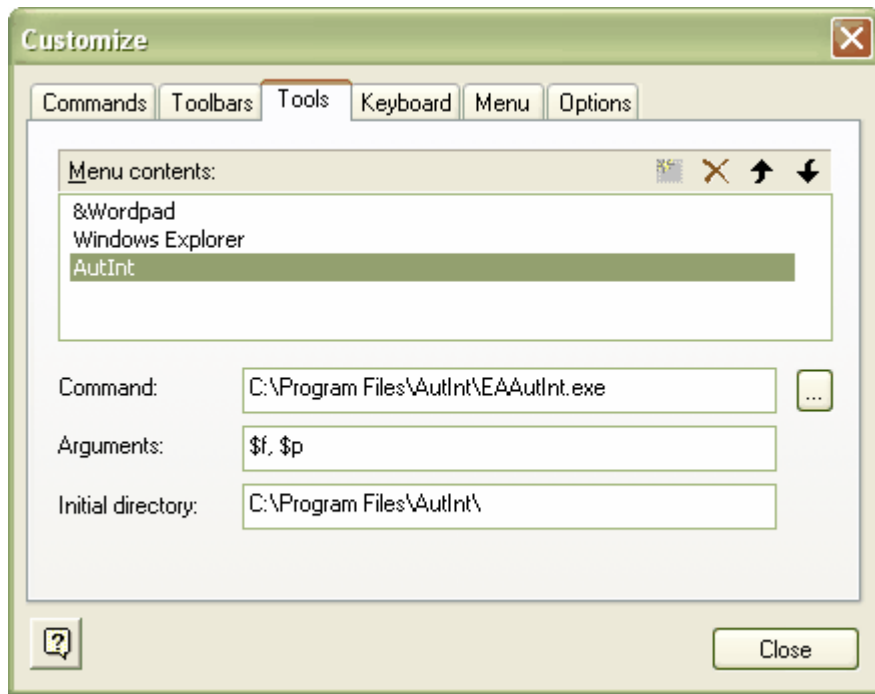
The two examples above would result in 'CustAcc-Wordpad' and 'CustAcc-VB' being added to the **Tools** menu as shown:

**4.4.7.1.3.2 Passing Parameters to External Applications**

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

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

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



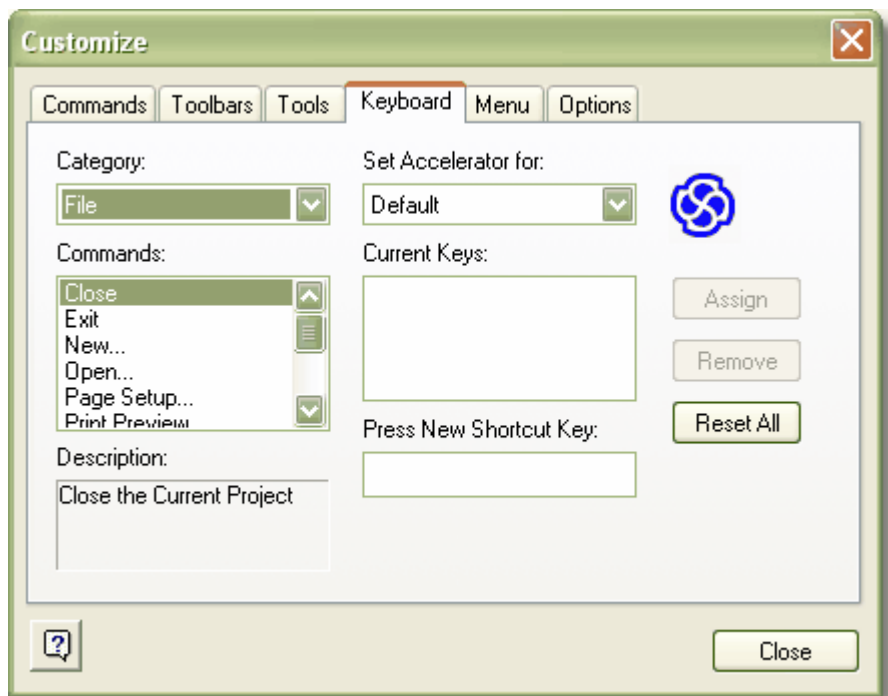
The available parameters for passing information to external applications are:

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

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

4.4.7.1.4 Customize Keyboard

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

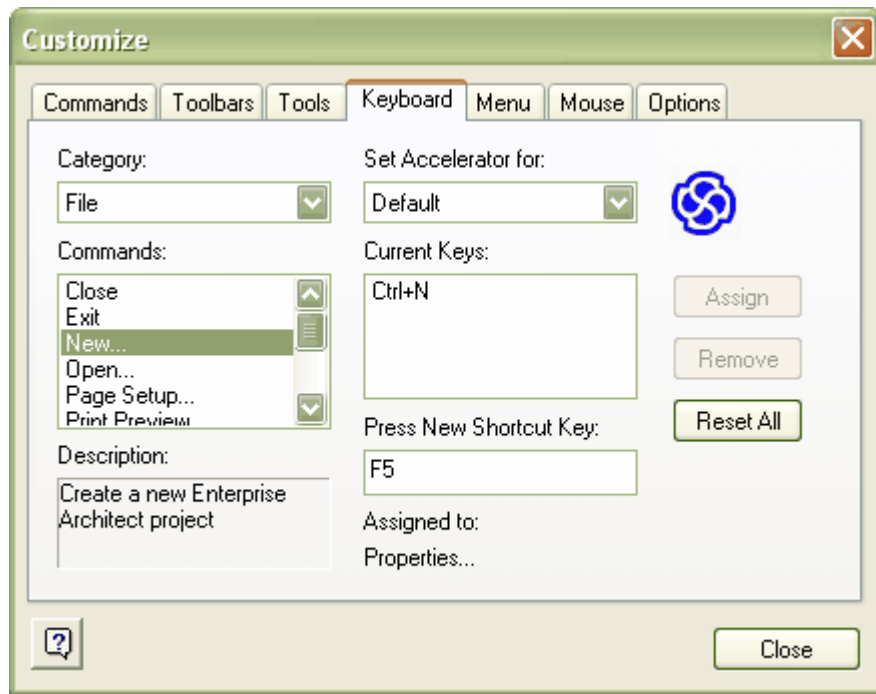


Modifying Keyboard Shortcuts

To modify a keyboard shortcut, follow these steps:

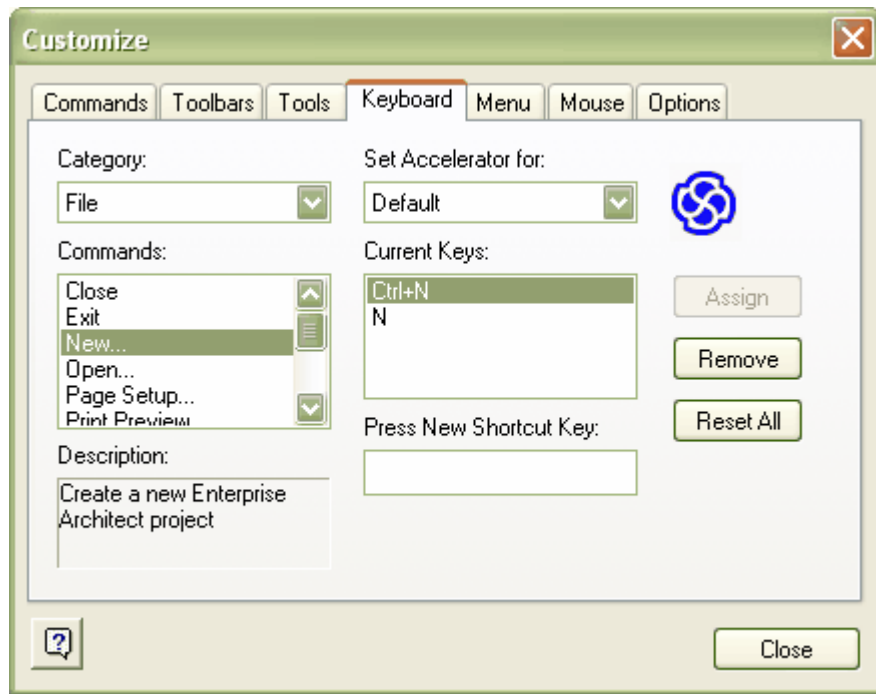
1. From the main menu select the **Tools | Customize** option to open the **Customize** dialog. Select the **Keyboard** tab.
2. Select the command you wish to modify: first select the menu it belongs to from the **Category** drop down list, then select the command from the **Commands** list. The current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

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



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

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



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

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

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

Warning: About Custom Layouts and Keyboard Shortcuts

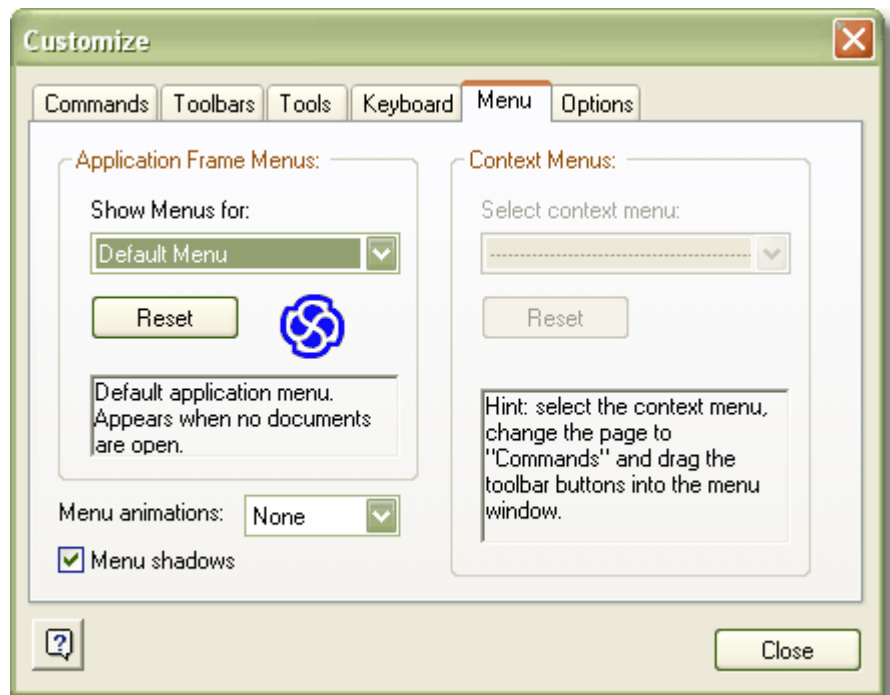
If you have set keyboard shortcuts, these will not be overridden if you switch to the Default Layout or the Iconix Layout option.

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

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

4.4.7.1.5 Customize Menu

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



Application Frame Menus

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

Context Menus

Currently this feature is disabled.

Menu Animations

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

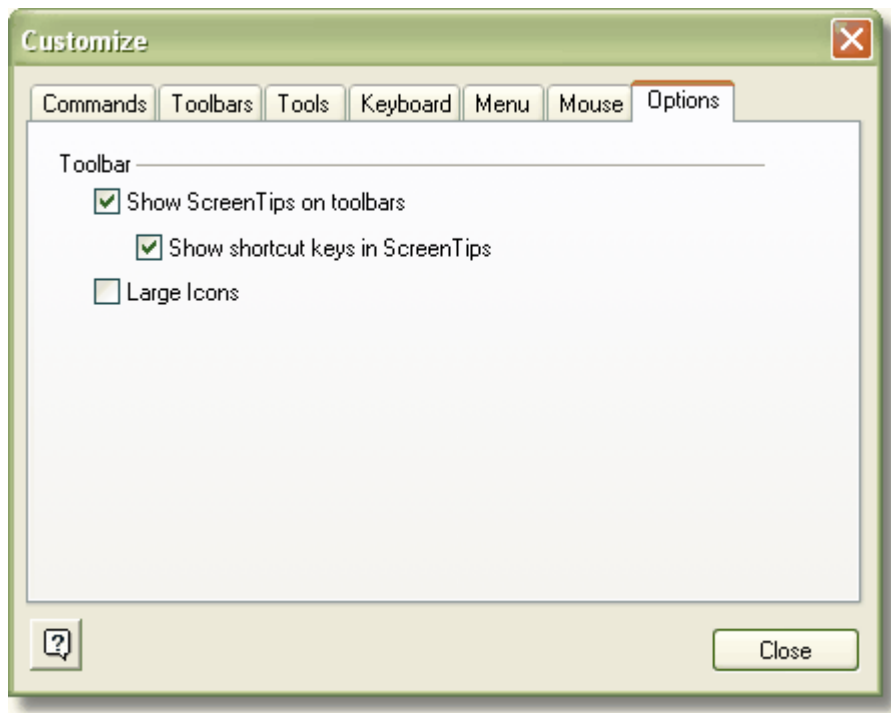


Menu Shadows

Menu shadows can be toggled on or off by checking or unchecking the *Menu shadows* checkbox.

4.4.7.1.6 Customize Options

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

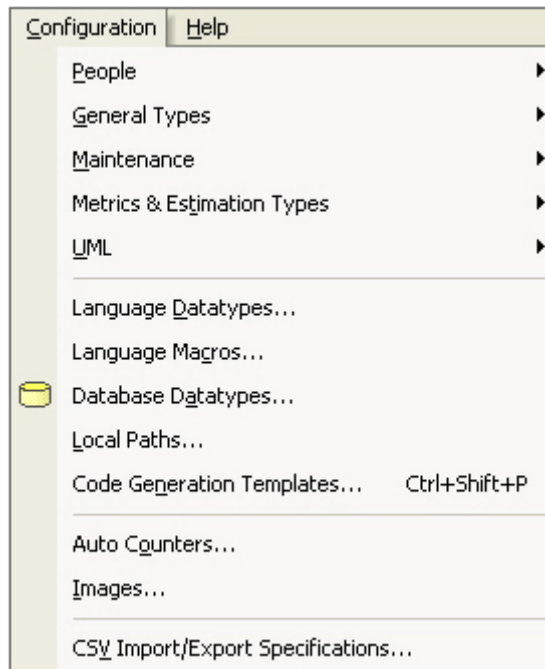


You can toggle the following options by checking or unchecking the checkboxes:

- Show screentips on toolbars.
- Show shortcut keys in screentips.
- Show large icons.

4.4.8 The Configuration Menu

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



Element	Description
People	See below.
General Types	See below.
Maintenance	See below.
Metrics & Estimation Types	See below.
UML	See below.
Language Datatypes	Add, modify and delete programming languages datatypes .
Language Macros	Add and delete language macros .
Database Datatypes	Add, modify and delete database datatypes .
Local Paths	Configure local directories and paths.
Code Generation Templates	Modify code generation templates using the Code Templates Editor . [Ctrl+Shift+P]
Auto Counters	Configure automatic naming for elements.
Images	Open Image Manager .
CSV Import/Export Specifications	Configure CSV import and export specifications.

The People Sub-Menu

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

Element	Description
Model Authors	Add, modify and delete project author(s).
Clients	Add, modify and delete project client(s).
Resources	Add, modify and delete project resource(s).
Roles	Add, modify and delete project role(s).

The General Types Sub-Menu

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

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

The Maintenance Sub-Menu

Keep track of problem types and test types.

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

The Metrics and Estimation Types Sub-Menu

Configure metrics and estimation types for your project.

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

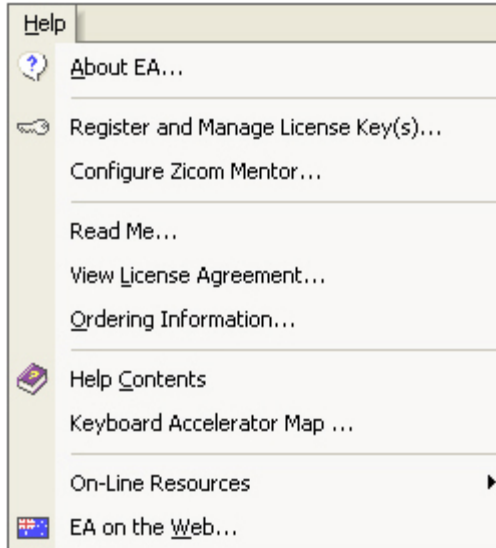
The UML Sub-Menu

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

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

4.4.9 The Help Menu

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



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

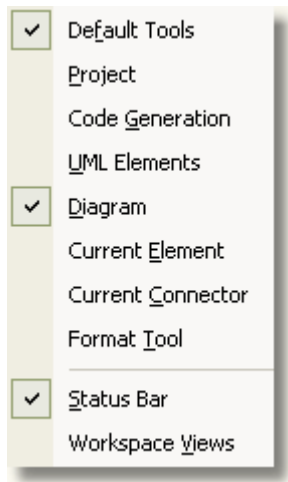
The On-Line Resources Sub-Menu

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

Element	Description
User Forum and News	Visit the EA user discussion forum .
Request-a-Feature	Request a feature you would like to see in EA.
Bug Report Page	Report the details of a bug you have found in EA.
Latest Version Details	Find out the details of the latest EA build .
Automation Interface	Access the EA Automation Interface guide.
Introducing UML	Access the Sparx Systems online UML tutorials .
Pricing and Purchase Options	Purchase or upgrade EA over the internet.

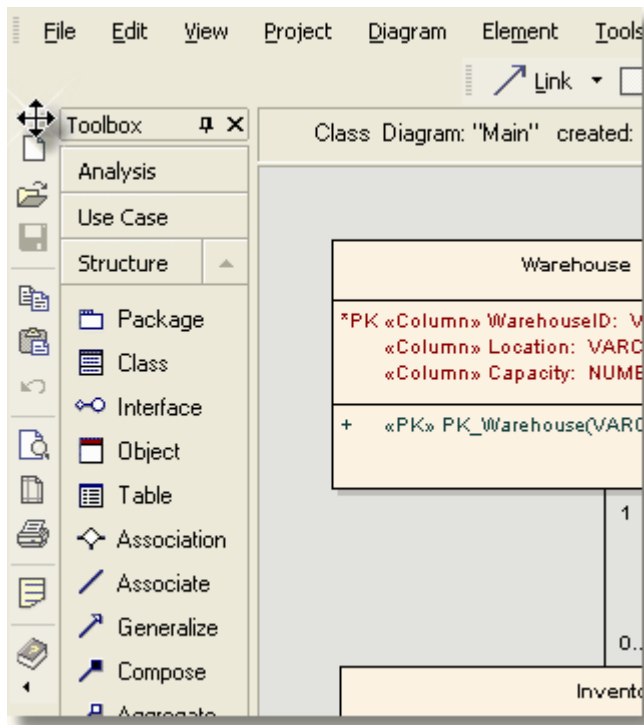
4.5 Workspace Toolbars

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



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

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



4.5.1 Default Tools Toolbar



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

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

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

4.5.2 Project Toolbar



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

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

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

4.5.3 Code Generation Toolbar

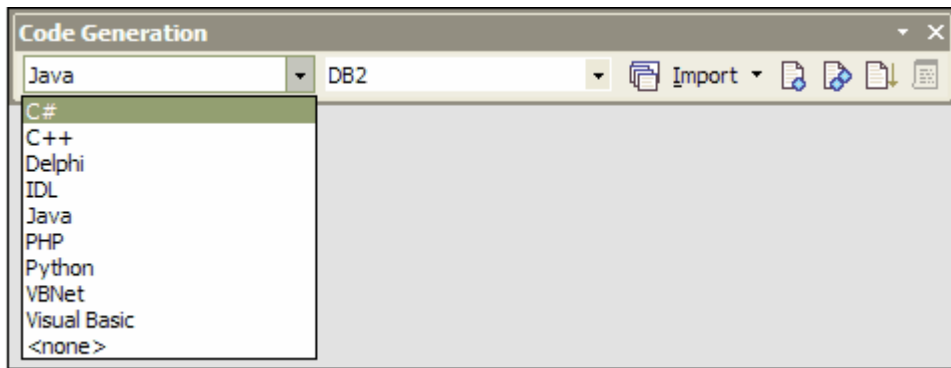


The *Code Generation* toolbar provides quick access to the following functions (in order):

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

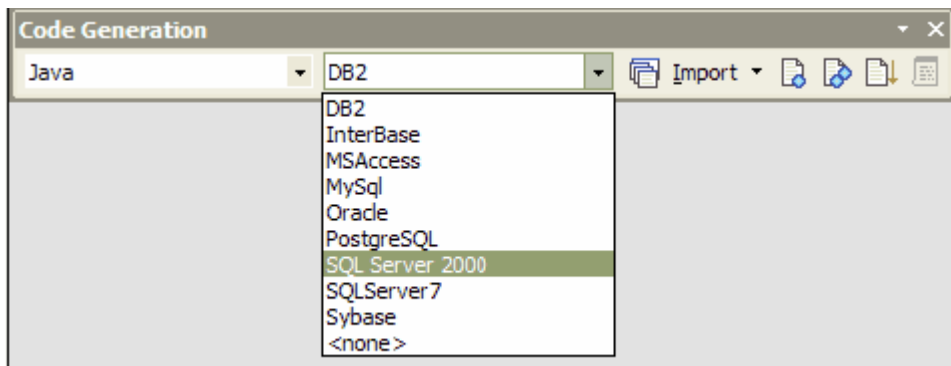
Set Default Code Language

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



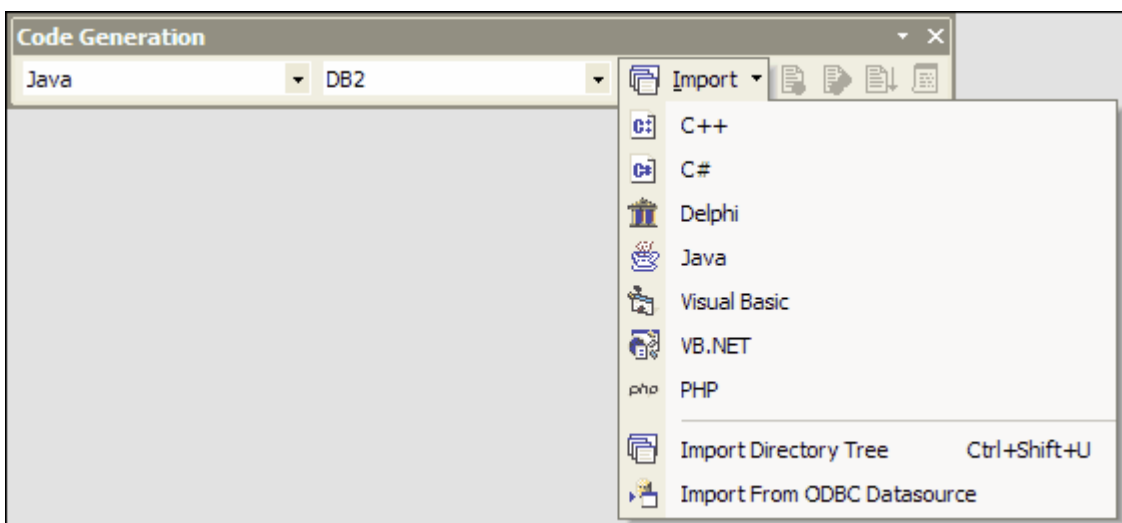
Set Default Database

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



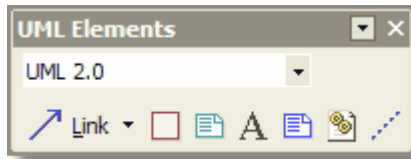
Import Code

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



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

4.5.4 UML Elements Toolbar

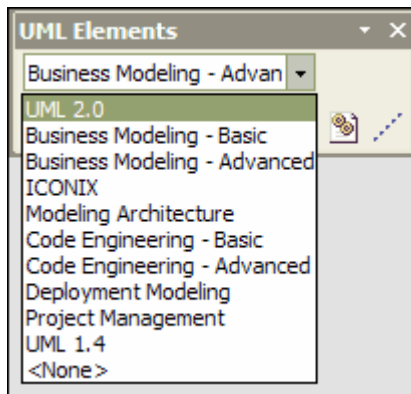


The **UML Elements** toolbar provides quick access to the following functions (in order):

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

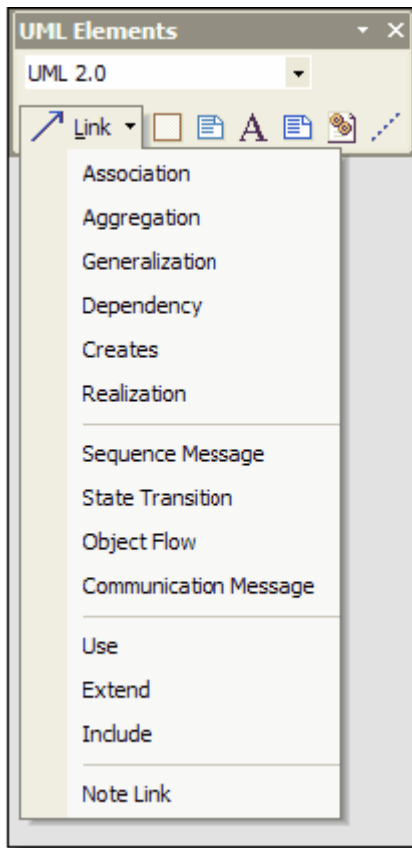
Select Workspace Perspective

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



Selecting Link type

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



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

4.5.5 Diagram Toolbar



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

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

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

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

4.5.6 Current Element Toolbar

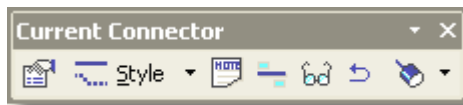


The *Current Element* toolbar provides quick access to the following functions (in order):

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

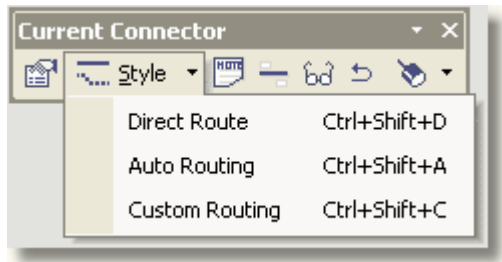
You may move this toolbar to any dockable position and it will retain that position in subsequent sessions. You may hide or show this toolbar from the *View | Toolbars* menu option.

4.5.7 Current Connector Toolbar

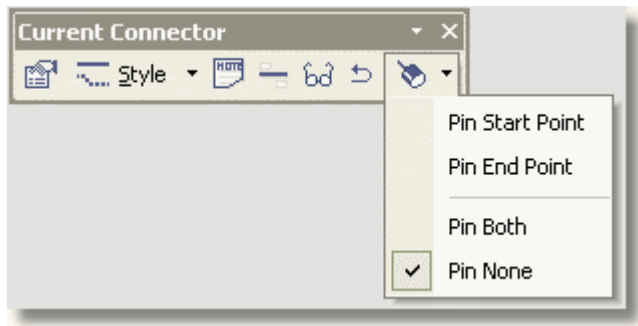


The *Current Connector* toolbar provides quick access to the following functions (in order):

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



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



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

4.5.8 Format Toolbar

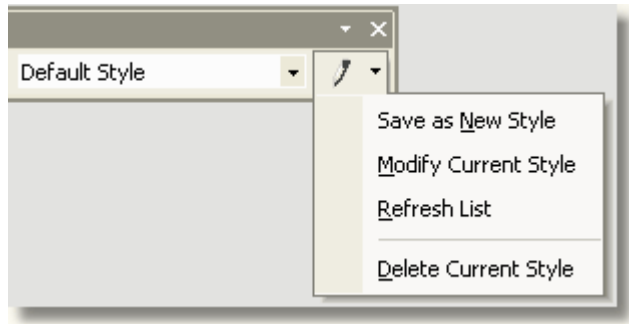


The *Format* toolbar is used to change the appearance of a specific element in the current diagram which does not effect any of the other elements of the same type throughout the model (to make a global change use the [Appearance dialog](#)). The *Format* toolbar provides quick access to the following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

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

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

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



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

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

4.5.9 Workspace Views



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

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

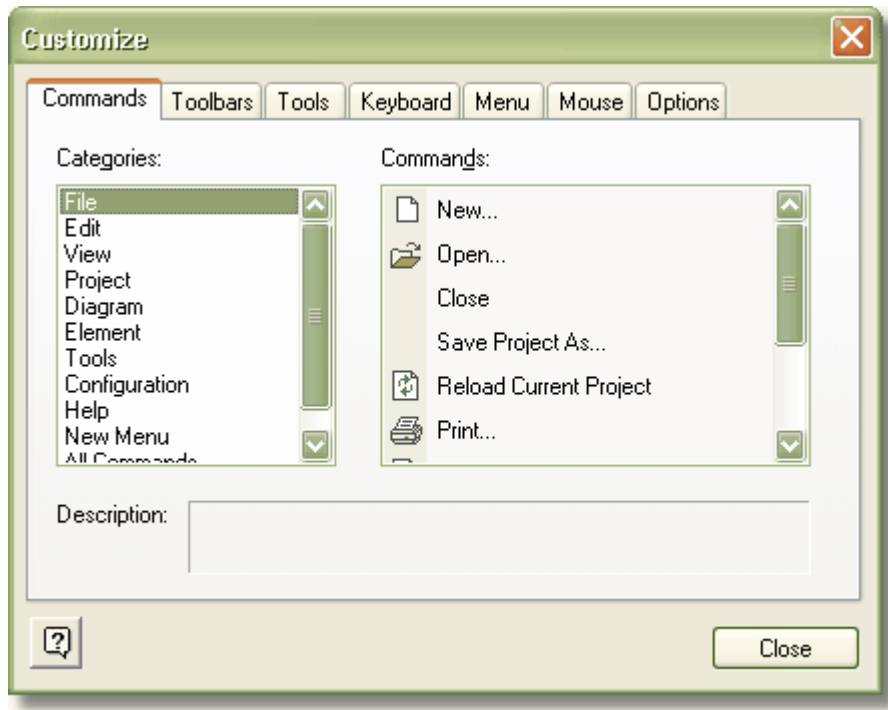
Click on any of these buttons to toggle the associated window on/off.

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

4.5.10 Customize Toolbars

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

as required.

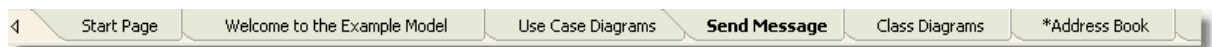


The Customize window allows you to customize the following:

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

4.5.11 Diagram Tabs

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

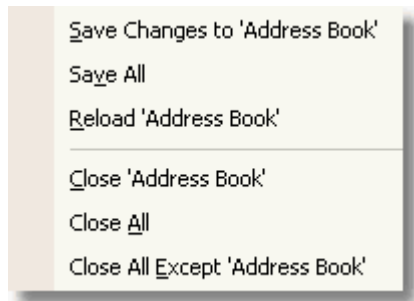


Notice that the Send Message tab is bold - this means that the current diagram is the Send Message diagram.

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

The Diagram Tabs Menu

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



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

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

4.5.12 Status Bar

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



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

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

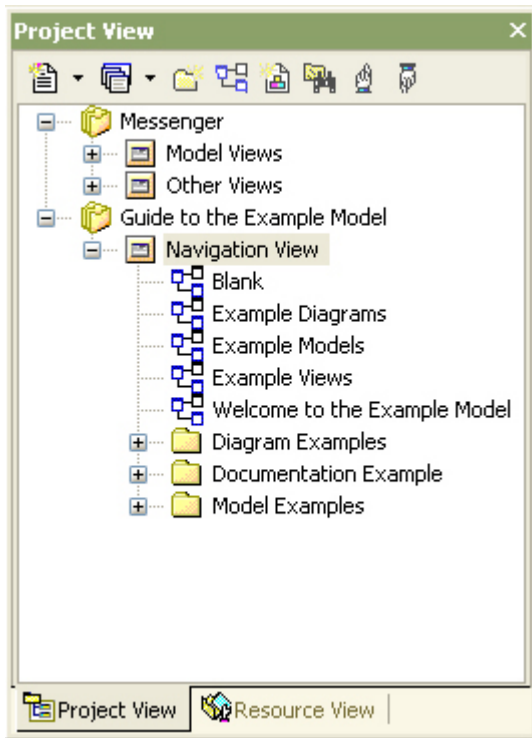
4.6 The Project Browser

The **Project Browser** allows you to navigate through the Enterprise Architect project space. It displays packages, diagrams, elements and element properties.

You can drag and drop elements between folders, or even drop elements from the Project Browser directly into

the current diagram.

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



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

Views

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

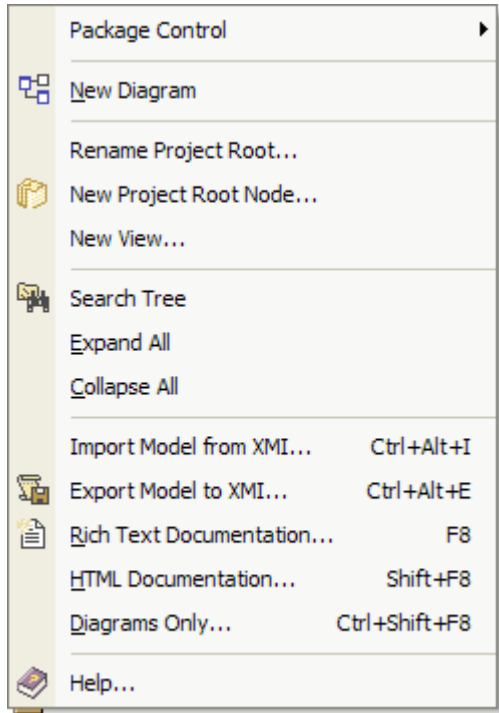
View	Description
Views	The root view and base of your project model.
Use Case View	The functional and early analysis view. Contains business process and use case models.
Dynamic View	Contains state charts, activity and interaction diagrams. The dynamics of your system.
Logical View	The class model and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

Note: You can hide and show the Project Browser by pressing the Hide/Show Project Browser button on the

toolbar, or use the Keyboard shortcut *Alt+0*.

4.6.1 Model Context Menu

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

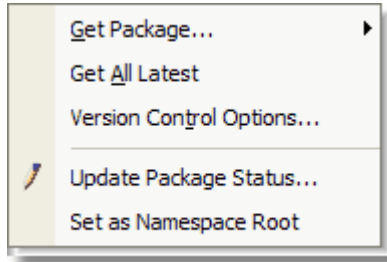


The Model context menu provides options to:

- Access Package Control Submenu
- Add a new diagram
- Rename the current model
- Create a new project root (model)
- Create a new View (package)
- Search the Project Browser
- Expand all items
- Close all items
- Import a model from an XMI file
- Export a model to XMI
- Produce RTF documentation for the model
- Produce HTML documentation for the model
- Produce a diagrams only report (in RTF) for the model

4.6.1.1 Root Node Package Control Sub-Menu

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

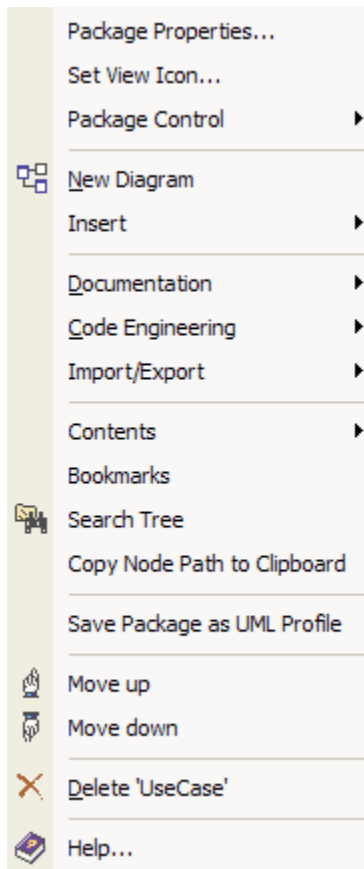


From this menu you can:

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

4.6.2 Package Context Menu

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

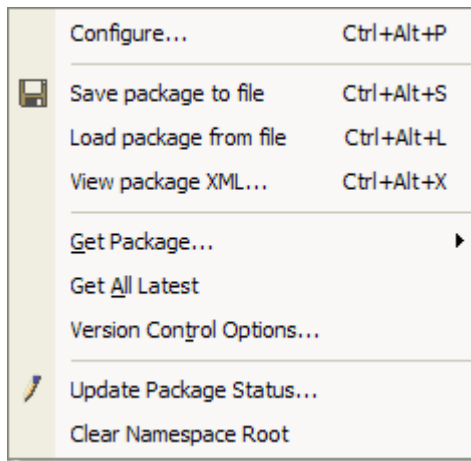


From this menu you can:

- Add new packages to the model
- Change the display icon for the selected package
- Create a new diagram within the View
- Add a New element to the View - the element may later be dragged onto a diagram
- For Use Case Views, produce a metrics report on the estimated model complexity
- Sort and order the package contents
- Search the View for specific elements
- Produce a variety of documentation in RTF format
- Open the View in the Relationship Matrix as Source or Target
- Control the package using XMI import and Export to text files
- Update the current status of elements in the package
- Perform Code Engineering functions
- Import and Export using XMI text files
- Move the View Up or Down in the list
- Get additional help

4.6.2.1 Package Control Sub-Menu

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

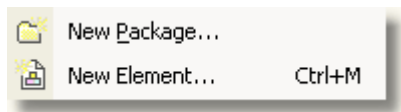


From this menu you can:

- Configure various settings for the package
- Save the package to file
- Load a package from a file
- View the XML for the selected package
- Get package for [version control](#)
- Configure [version control](#) options
- Update the package's status
- Clear the namespace root

4.6.2.2 Insert Sub-Menu

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

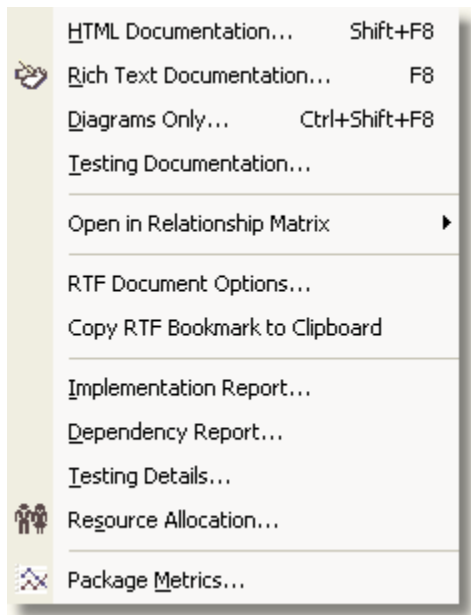


From this menu you can:

- Create a new package
- Create a new element

4.6.2.3 Documentation Sub-Menu

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

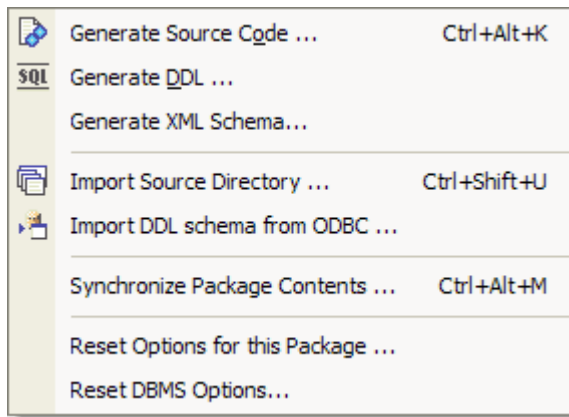


From this menu you can:

- Generate [HTML documentation](#)
- Generate [rich text documentation](#)
- Generate [diagrams only documentation](#)
- Generate [testing documentation](#)
- Open as a source or target in the [relationship matrix](#)
- Set [RTF documentation options](#)
- Copy a [bookmark](#) for linking RTF elements to the clipboard
- Generate an [implementation report](#)
- Generate a [dependency report](#)
- View [testing details](#)
- View [resource allocation](#)
- View [package metrics](#)

4.6.2.4 Code Engineering Sub-Menu

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

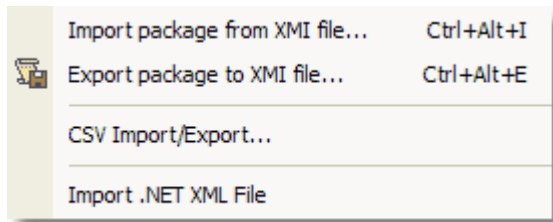


From this menu you can:

- [Generate source code](#)
- [Generate a DDL](#)
- [Generate XML schema](#)
- [Import a source directory](#)
- Import a DDL schema from a data source
- [Synchronize](#) the package contents
- [Reset the options](#) for this package

4.6.2.5 Import/Export Sub-Menu

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



From this menu you can:

- [Import a package](#) from an XMI file
- [Export a package](#) to an XMI file
- Perform a CSV [import](#) or [export](#)
- Import a .NET XML File

4.6.2.6 Contents Sub-Menu

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

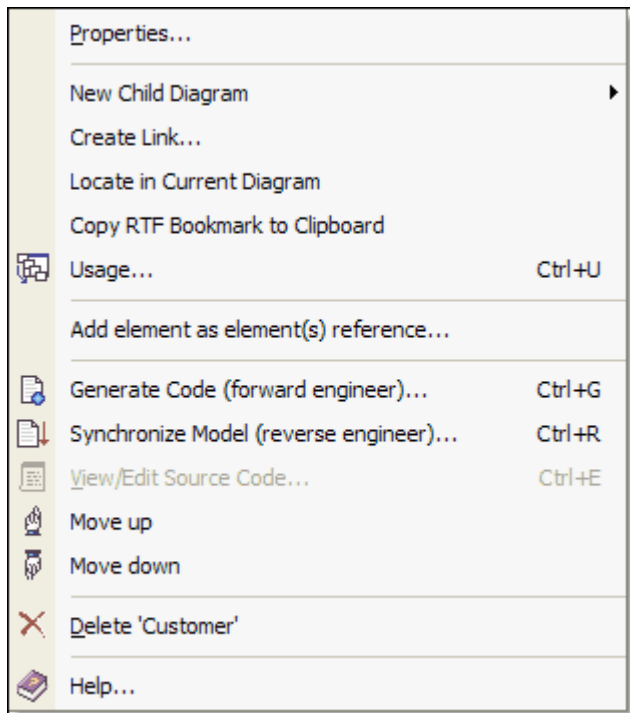


From this menu you can:

- Expand all of the items in the Project Browser
- Collapse all of the items in the Project Browser
- Reset the sort order

4.6.3 Element Context Menu - Project Browser

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



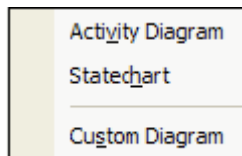
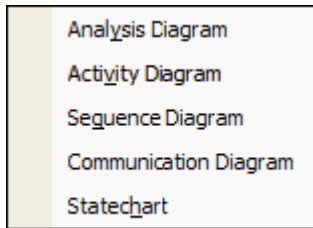
From this menu you can:

- View and modify the element's properties
- Create a [new child diagram](#)
- Create a [link](#)
- Locate the element in the diagram
- Copy a [bookmark](#) for linking RTF elements to the clipboard
- Show [Element Usage](#) within the model
- Add this element to others as a [cross reference](#)
- If supported, edit the list of element [attributes](#)
- If supported, edit the list of element [operations](#)
- Move the element up or down in the Project Browser list

- Delete the element from the model
- Get additional help

The New Child Diagram Sub-Menu

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

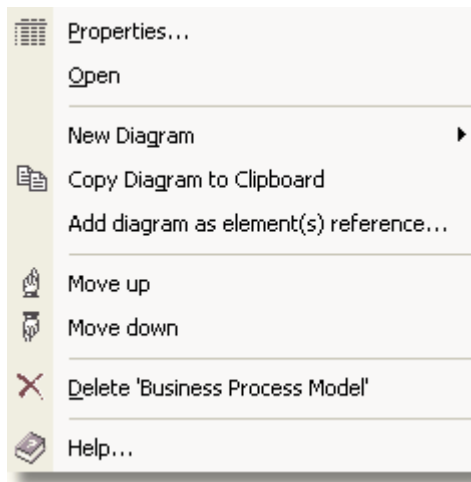


From these menu you can create:

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

4.6.4 Diagram Context Menu - Project Browser

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



From this menu you can:

- View and modify the diagram's properties
- Open the diagram in the View panel
- Create a [new diagram](#) within the current package
- Copy the diagram to the clipboard as an internal link only for pasting/copying to another location
- Copy a [bookmark](#) in RTF format to the clipboard
- Add this diagram as a [cross reference](#) to other elements
- Move the diagram up or down in the list of diagrams within this package
- Delete the diagram
- Get additional help

4.6.5 Order Package Contents

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

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

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

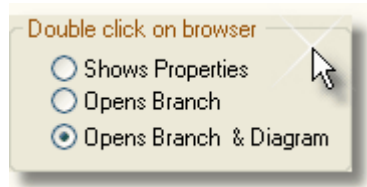
Also see: [Setting the Default Project Browser Behavior](#).

4.6.6 Setting Default Project Browser Behavior

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

Double Click Behavior

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



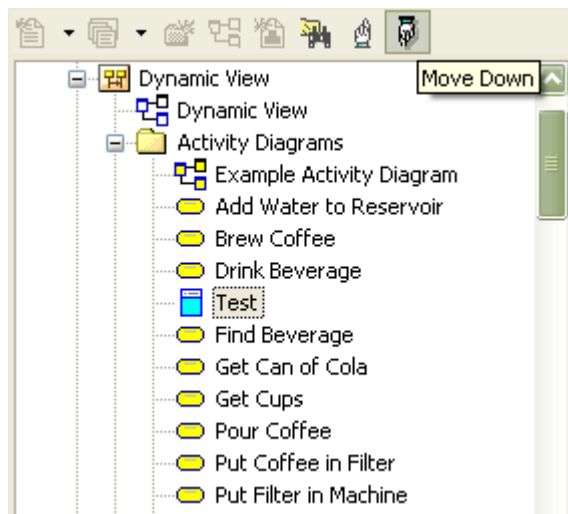
- *Shows Properties* - double clicking an item in the Project Browser will open a property dialog (if available) for that element.
- *Opens Branch* - double clicking an item in the Project Browser will expand the tree to show the item's children. If there are no children, nothing will happen.
- *Opens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

Allow Free Sorting

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

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

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



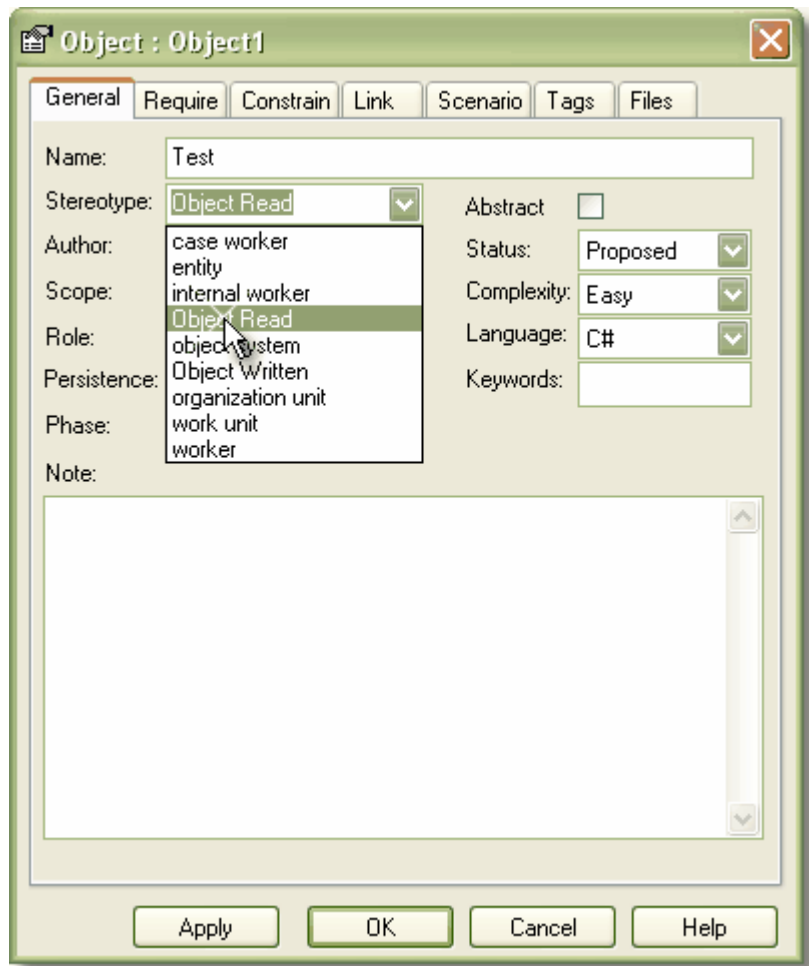
As the tooltip suggests - the *Move Down* icon moves the element further down the tree, and the *Move Up* icon moves the element further up the tree.

Show Stereotypes

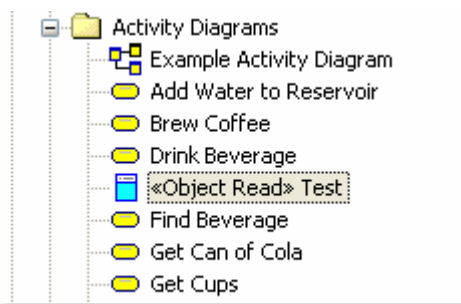
1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu. Navigate to the *General* page of the dialog.
2. Check the *Show Stereotypes* checkbox.
3. As prompted, shutdown and restart EA to enable this change to take effect.

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

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



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



4.7 Dockable Windows Available

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

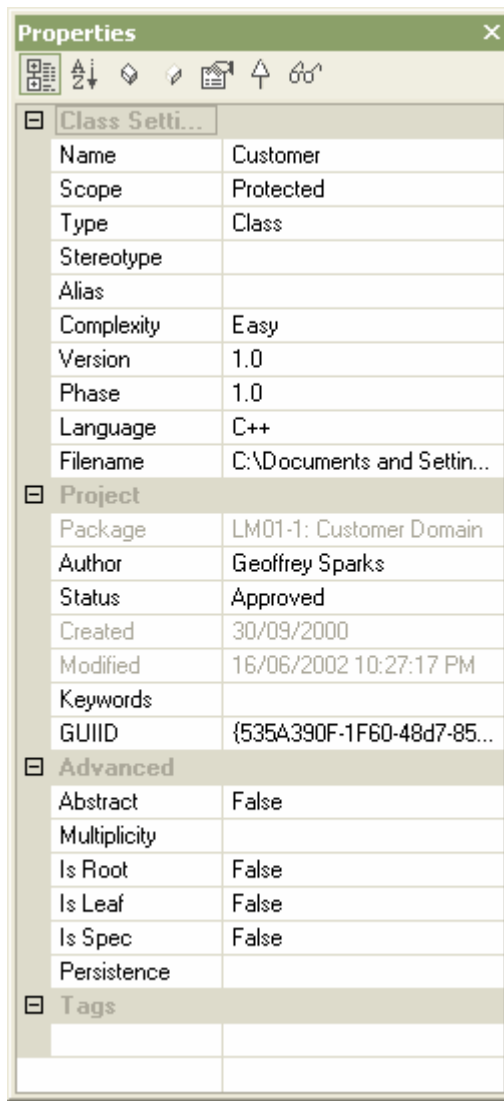
The dockable windows available include:

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

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

4.7.1 The Properties Window

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



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

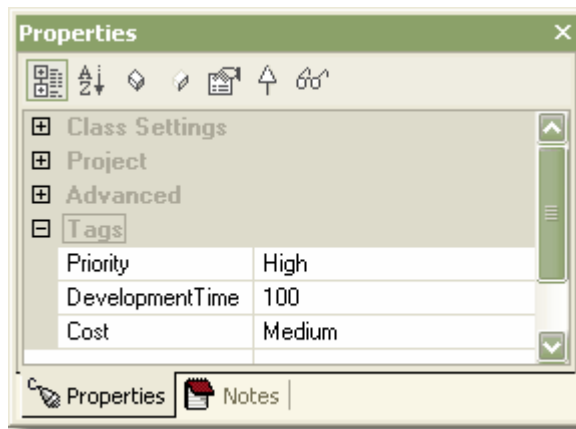
Properties Sections

The Properties tab is divided into three expandable sections:

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

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

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



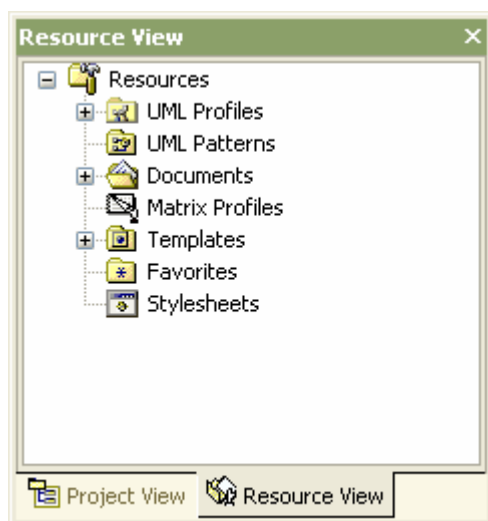
4.7.2 The Resource Window

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

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

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

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



See:

[UML Profiles](#)

[UML Patterns](#)

[MDG Technologies](#)

[Resource Documents](#)

[RTF Templates](#)
[Web Style Templates](#)
[Matrix Profiles](#)
[Favorites](#)

4.7.2.1 The Resource View

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



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

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

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

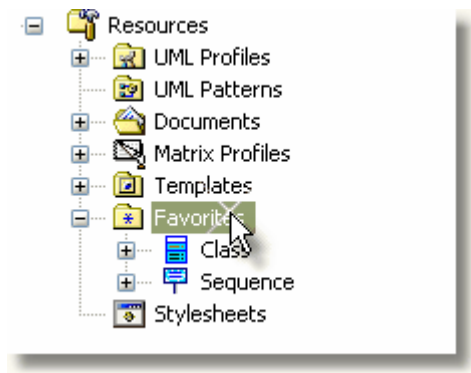
The Templates folder holds templates you can design yourself to apply to your documentation in either [RTF style](#) or [Web style](#).

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

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

4.7.2.2 Favorites

The resource view contains a *Favorites* folder. Here you can link to any UML element from the model as a whole, and conveniently drag and drop instances or links to this element into other diagrams. This is particularly useful where certain elements - eg. the list of Actors in a systems - are re-used again and again, and switching to the Actors folder is not convenient. In cases like this, using the Favorites folder makes managing and creating your model much easier.



Modifying the Favorites Folder

Add to the Favorites Folder

To add an element to the Favorites folder:

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

Delete from the Favorites Folder

To delete a favorite:

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

View Properties of a Favorite

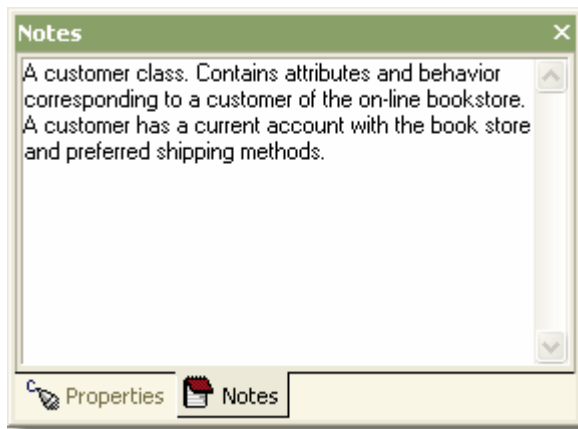
To view a favorite's properties from the Favorites folder:

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

4.7.3 The Notes Window

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

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



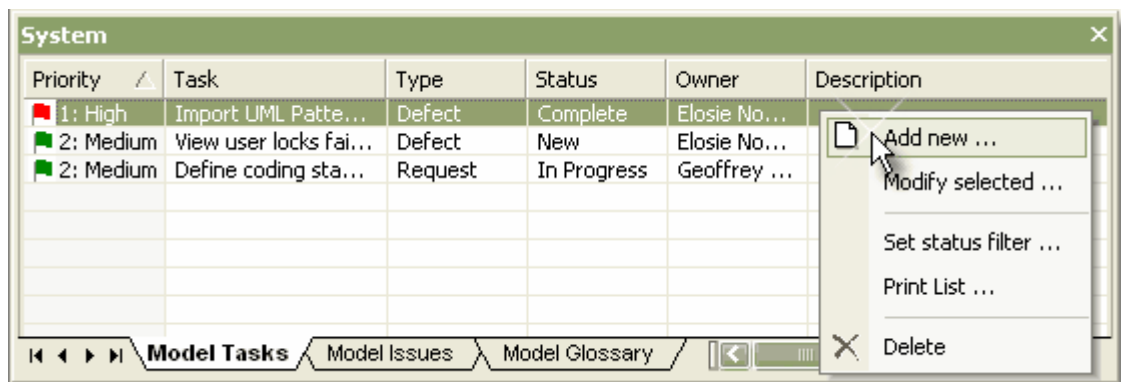
Tip: You can also edit notes by double clicking on an element in a diagram or in the Project Browser to open the property dialog.

4.7.4 The System Window

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

The System tab is divided into three sections:

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

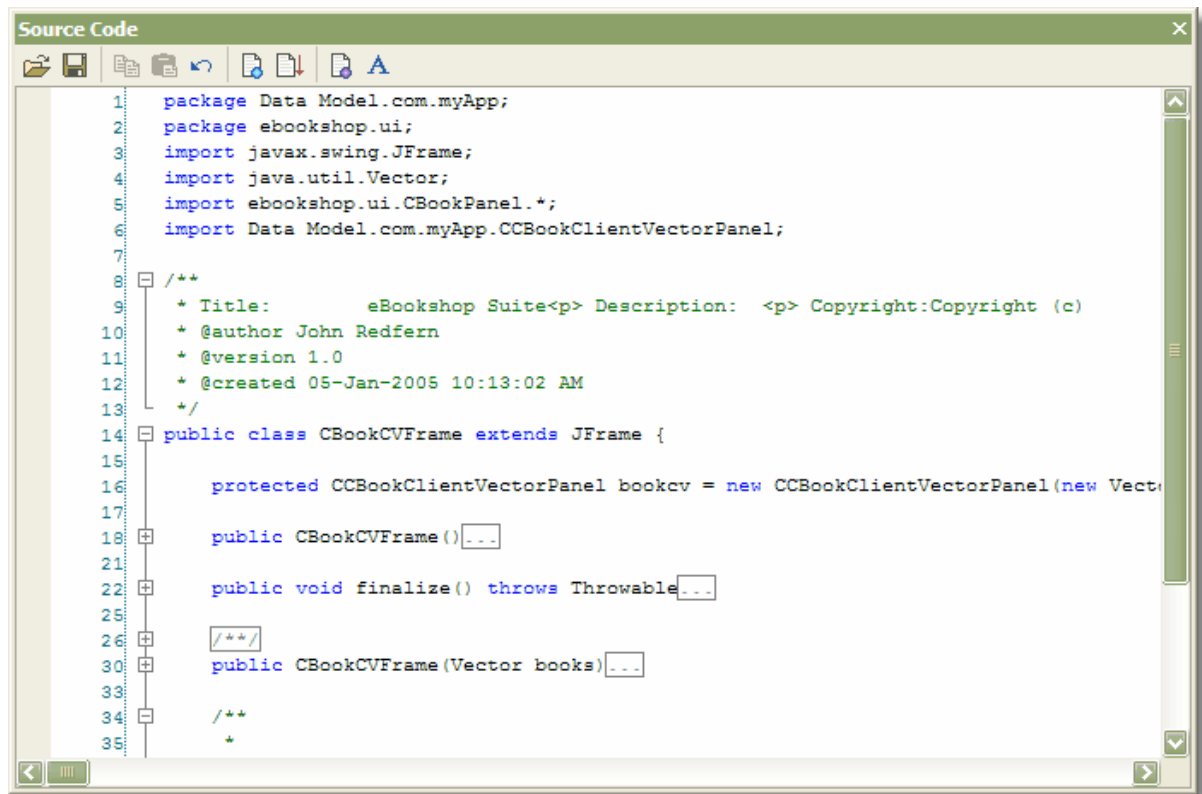


Tip: Right-clicking in the system window will bring up a context-sensitive menu, which has options for filtering tasks/issues by status, and glossary by term. You can also rearrange the sort-order by left-clicking in the title bar of the column which you wish the items to be indexed on.

4.7.5 The Source Code Viewer

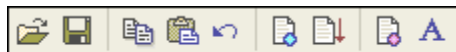
The **Source Code** viewer can be used to view any source code you wish to open. If a class is selected, it will show the source code for that class, provided it has already been generated. By default the source code viewer is set to display the line numbers and to have outlining enabled. To turn off either of these options go to










Tools | Options and from the *Local Options* dialog hierarchy select the *Code Editors* item and ensure that the *Show Line Numbers* or *Enable Outlining* checkboxes are not selected.



The Source Code Viewer window menu buttons

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

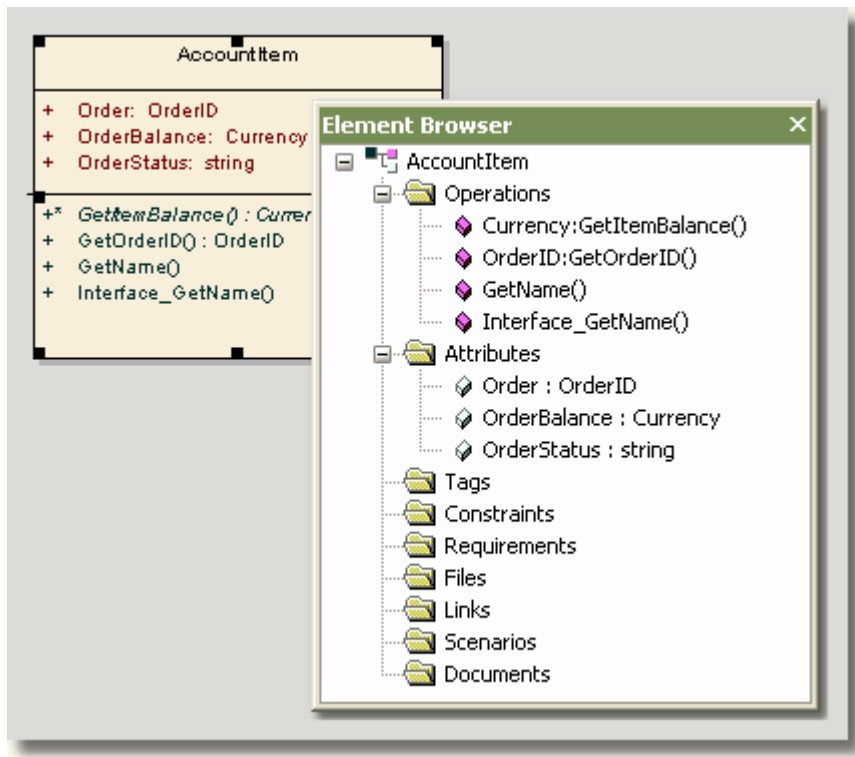


	The Open button opens the source code from an existing file.
	The Save Button saves the changes to the currently loaded source code.
	The Copy button copies the highlighted text.
	The Paste button pastes the text that is currently contained in the buffer to the source code viewer.
	The Undo but cancels the previous action
	The Generate and Reload button generates and reloads the current object source.
	The Save and Resynch button save the source code and re synchronizes the class.
	The Code Templates button access Code Templates Editor .
	The Set Font button sets the font for the text contained in the Source Viewer.

See also: [Generating Source Code](#)

4.7.6 The Element Browser

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



The following are displayed where available:

- Operations
- Attributes
- Tags
- Constraints
- Requirements
- Files
- Links
- Scenarios
- Documents

4.7.7 The Relationships Window

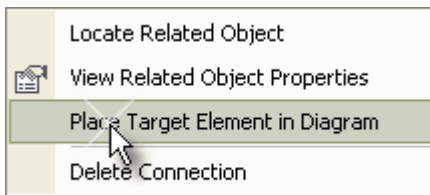
The *Relationships* window displays all connections between currently selected element and other elements. This provides a quick overview of an elements relations in the model.

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

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

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

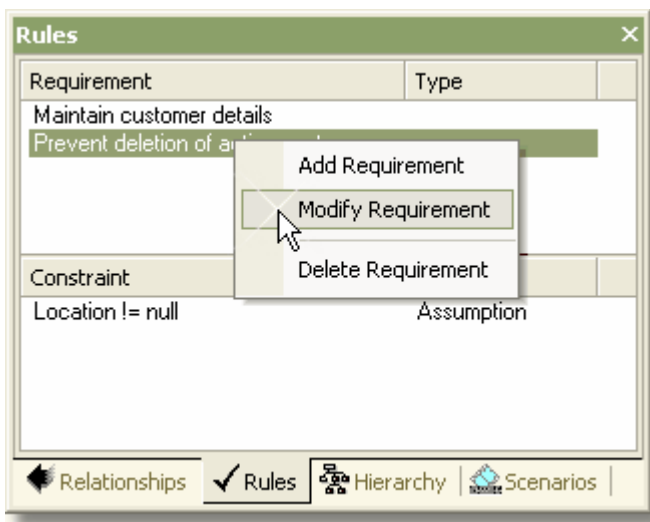
If an element is not visible in the current diagram, the context menu will have an option to place the selected element in the current diagram. This is useful when you are building a picture of what an element interacts with - especially when reverse engineering an existing code base.



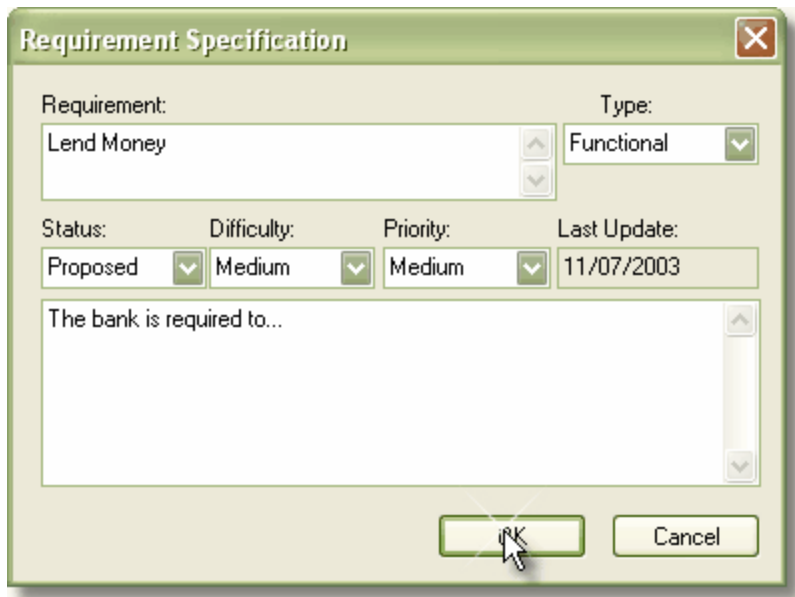
4.7.8 The Rules Window

The *Rules* window displays all *requirements* and *constraints* entered against an element. This is a quick view of the rules associated with an element.

To add, modify or delete requirements and constraints from this tab, right click with the mouse in the appropriate list area, and select an option from the context menu.



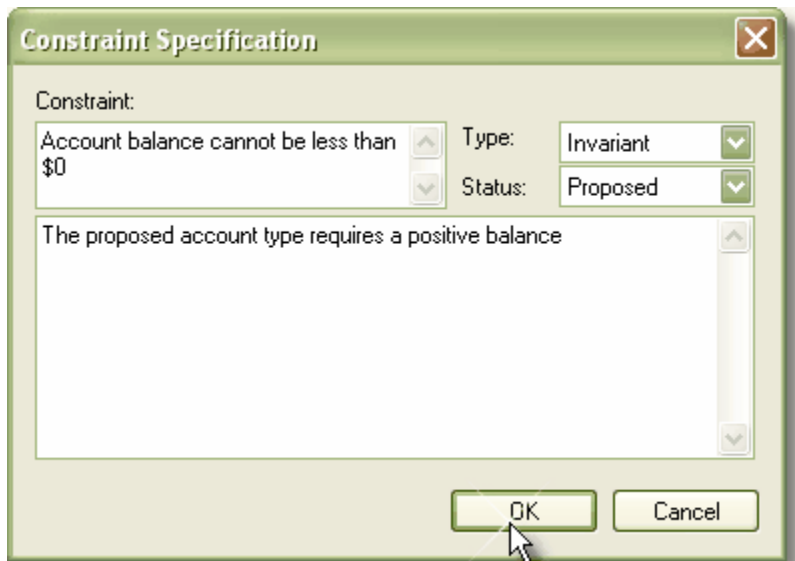
Select *Add Requirement* from the Requirement list context menu to open the *Requirement Specification* dialog. Enter the appropriate information and press *OK* to save.



The *Requirement Specification* dialog box is shown. It has a title bar with a close button (X). The dialog contains the following fields:

- Requirement:** A text box containing "Lend Money" with up and down arrow buttons.
- Type:** A dropdown menu set to "Functional".
- Status:** A dropdown menu set to "Proposed".
- Difficulty:** A dropdown menu set to "Medium".
- Priority:** A dropdown menu set to "Medium".
- Last Update:** A text box containing "11/07/2003".
- Description:** A large text area containing "The bank is required to...".
- Buttons:** "OK" and "Cancel" buttons at the bottom.

Select *Add Constraint* from the Constraint list context menu to open the *Constraint Specification* dialog. Enter the appropriate information and press *OK* to save.



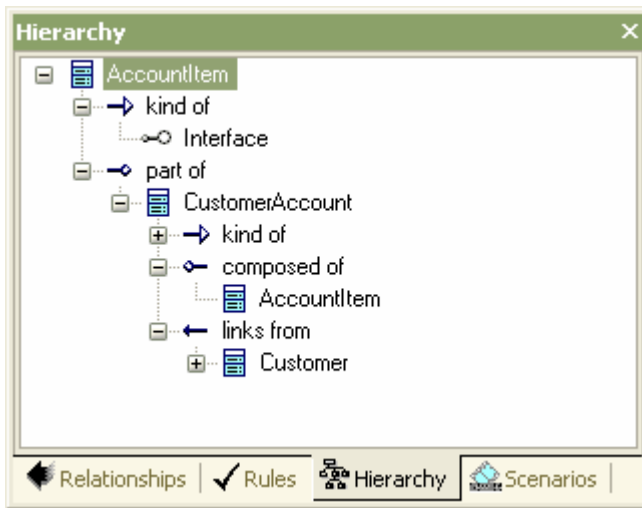
The *Constraint Specification* dialog box is shown. It has a title bar with a close button (X). The dialog contains the following fields:

- Constraint:** A text box containing "Account balance cannot be less than \$0" with up and down arrow buttons.
- Type:** A dropdown menu set to "Invariant".
- Status:** A dropdown menu set to "Proposed".
- Description:** A large text area containing "The proposed account type requires a positive balance".
- Buttons:** "OK" and "Cancel" buttons at the bottom.

4.7.9 The Hierarchy Window

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

This information is derived from Aggregation and Inheritance relationships and includes the relationships of child or related classes - this helps extend the picture of where an element exists in the model space.

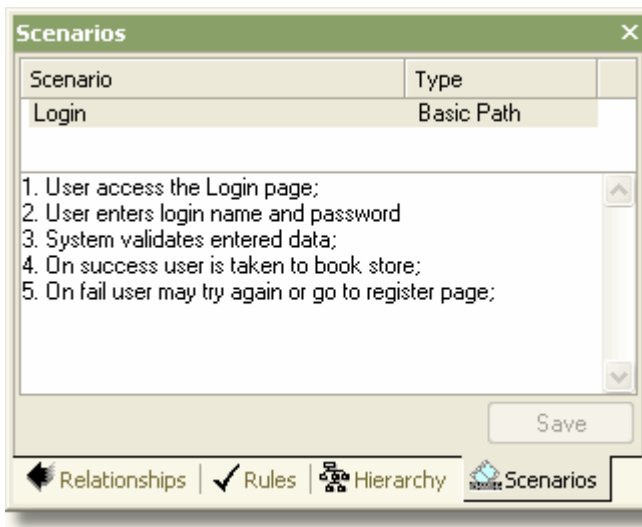


Tip: You can alter the maximum number and the initial number of levels that the hierarchy will open to on the **Tools | Options | General** tab (accessed from the main menu).

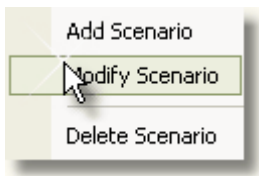
4.7.10 The Scenario Window

The **Scenario** window is used to view, modify and manage scenarios for the currently selected element - either in the current diagram or in the Project Browser.

This provides a convenient way to quickly view and edit scenarios when you are performing analysis work.



Access the context menu on the scenario list to add, modify and delete scenarios.



You can edit the text in the scenario tab note area and press **Save** to quickly save scenario notes. If you wish to edit other details or re-order the scenarios, then you can select **Modify** in the context menu or double click on a list item. The **Edit Scenarios** dialog is the same as that for the Object Properties dialog.

4.7.11 The Tagged Values Window

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

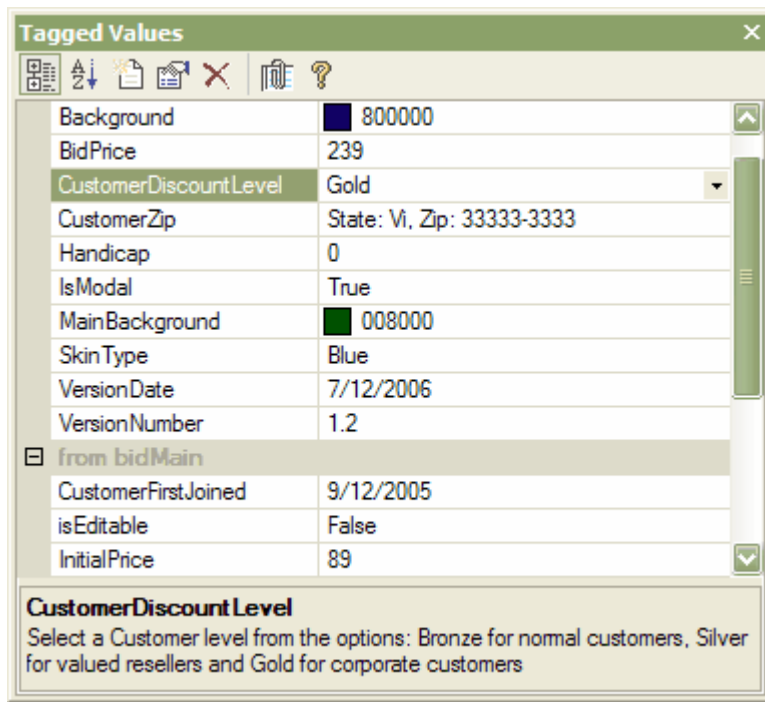
From the Tagged Values dockable window the user may perform the following actions:

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

Model Elements and features with tagged values

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

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

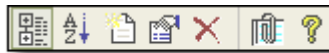









When over-riding an inherited property, EA copies the tag from the parent down to the child element and sets the new value, leaving the original tag unchanged.

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

Tagged Values window menu buttons

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



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

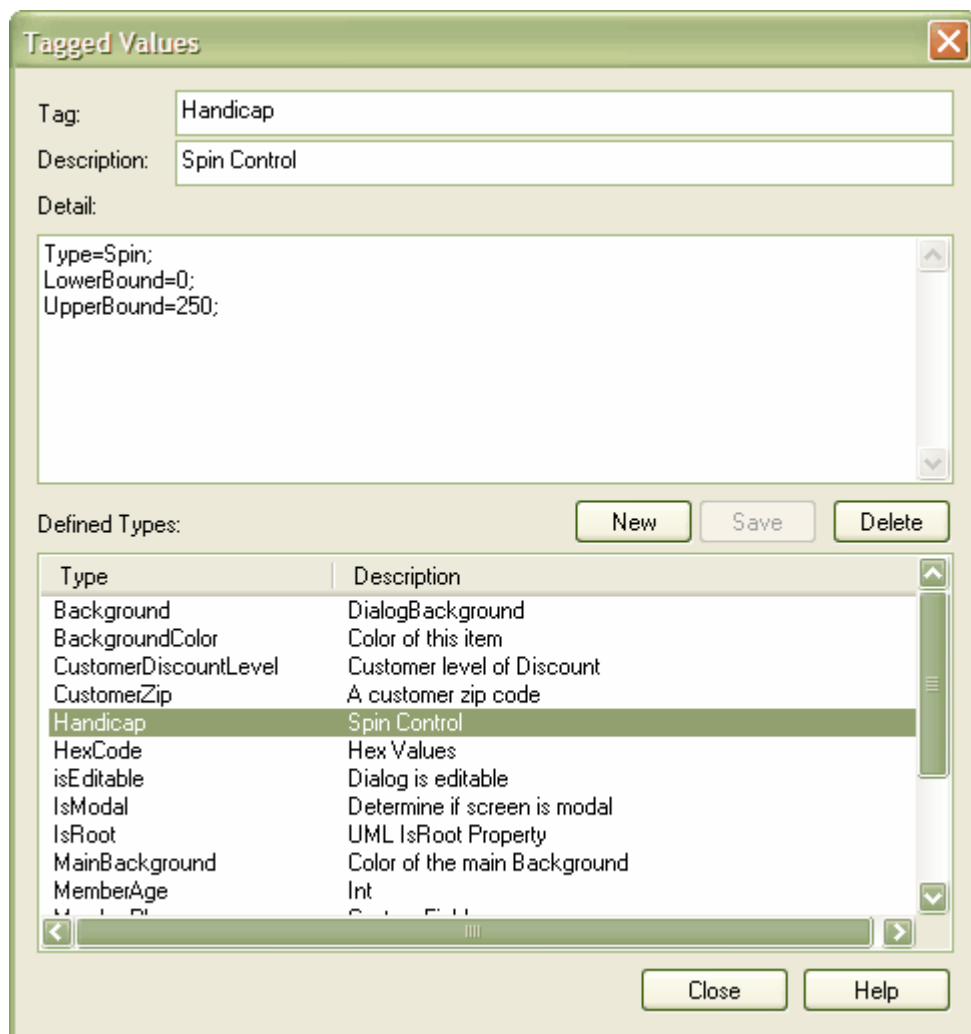
4.7.11.1 Predefined Tagged Value Types

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

Creating Structured Tags

To create a structured tag use the following instructions:

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



Types and Arguments

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

Integer	Type=Integer;	This predefined type allows the entry of an Integer value.
Float, Decimal, Double	Type=Float; Type=Decimal; Type=Double;	This predefined type allows the entry of a Float, decimal or a Double value. These types all map to the same type of data.
String	Type=String;	This predefined type allows the entry of a String value.
Enum	Type=Enum; Values=Val1, Val2, Val3; Default=Val2;	The Enum predefined type allows the user to define a comma separated list where Val1, Val2 and Val3 represent values in the list and Default represents the default value of the list.
Const	Type=Const; Default=Val;	The Const is a predefined type allows the user to create a read-only constant value.
Color	Type=Color;	Using the Color predefined type allows the user to input a color value from a color chooser menu.
Custom	Type= Custom;	Using the Custom predefined type allows the user to create their own template for predefined types, more information is provided in the Custom tagged vales type section .
DateTime	Type=DateTime;	Using the DateTime predefined type allow the user to input the date and time for the tagged value from a calendar menu.
Boolean	Type=Boolean;	Using the boolean predefined type allows the input of a true or false value.
Memo	Type=Memo;	The Memo predefined type allows the input of large and complex tag values.
Spin	Type=Spin; LowerBound=x; UpperBound=x;	The Spin predefined type allows the user to create a spin control with the value of LowerBound being the lowest value and UpperBound being the highest value.

4.7.11.2 Predefined Reference Data

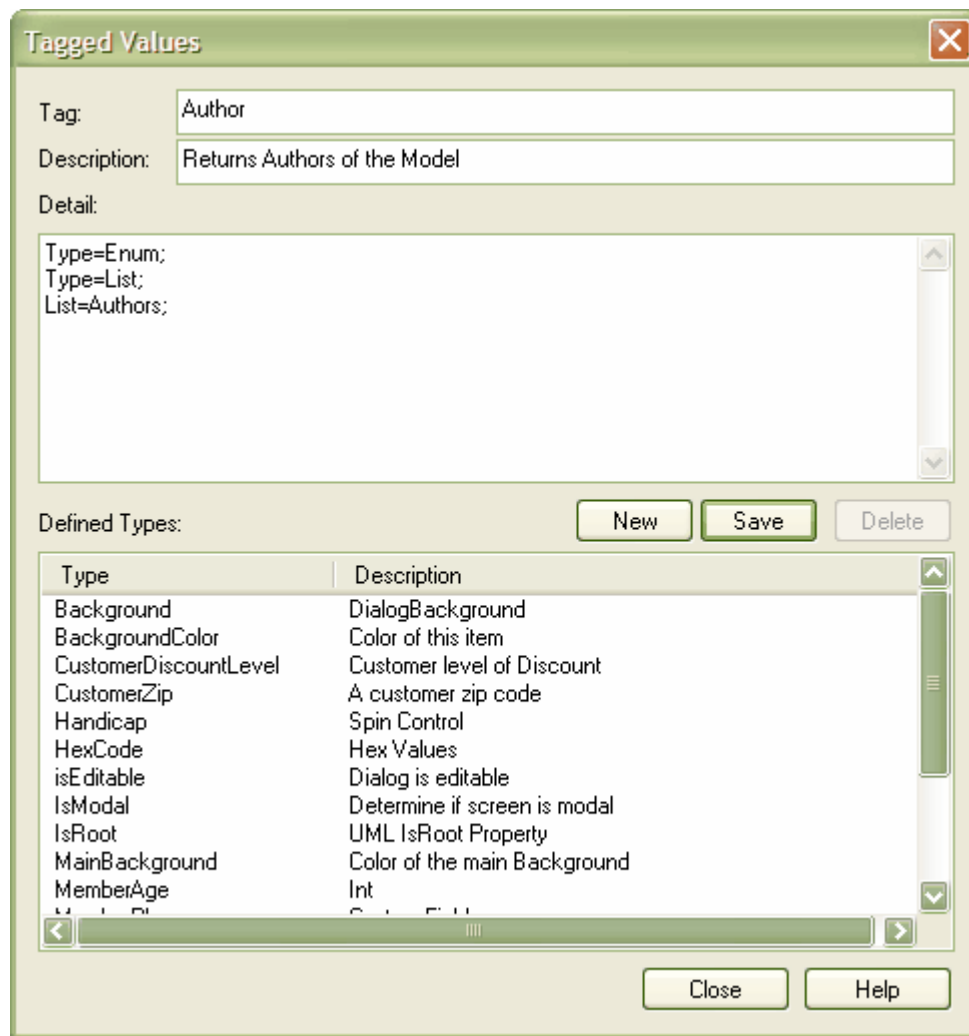
This table details the predefined tagged value types that is used to return the values held in a relevant table in EA along with the syntax required for their use:

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

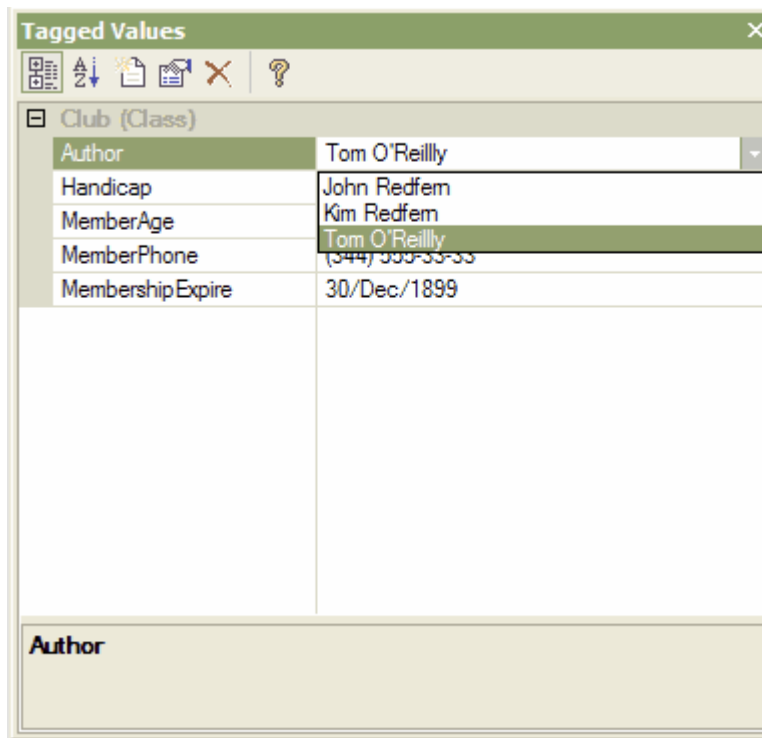
Creating Predefined Reference Data Tagged Value Types

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

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



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



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

4.7.11.3 Creating a Custom Tagged Value Type

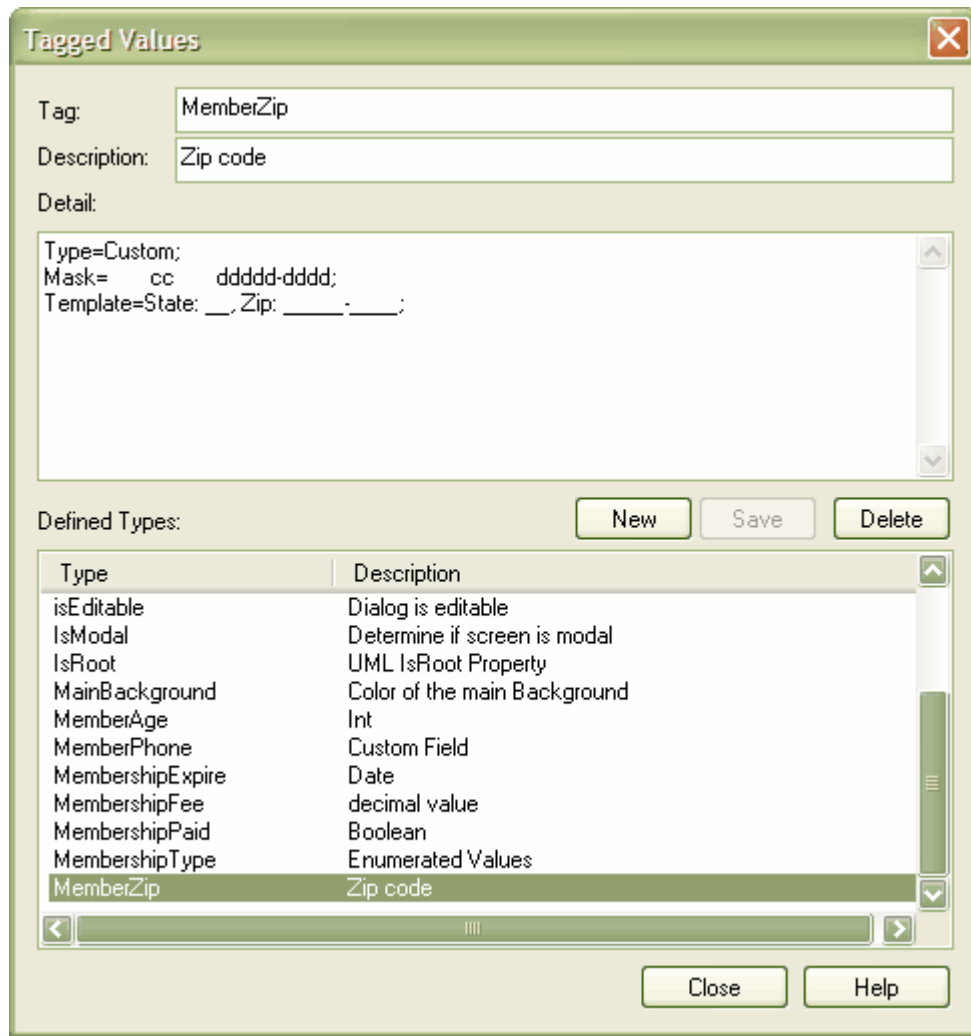
Creating a custom tagged value allows the user a great deal of flexibility for creating their own masked tagged values. To create a user defined masked tagged value follow the procedure detailed below:

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

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

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

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

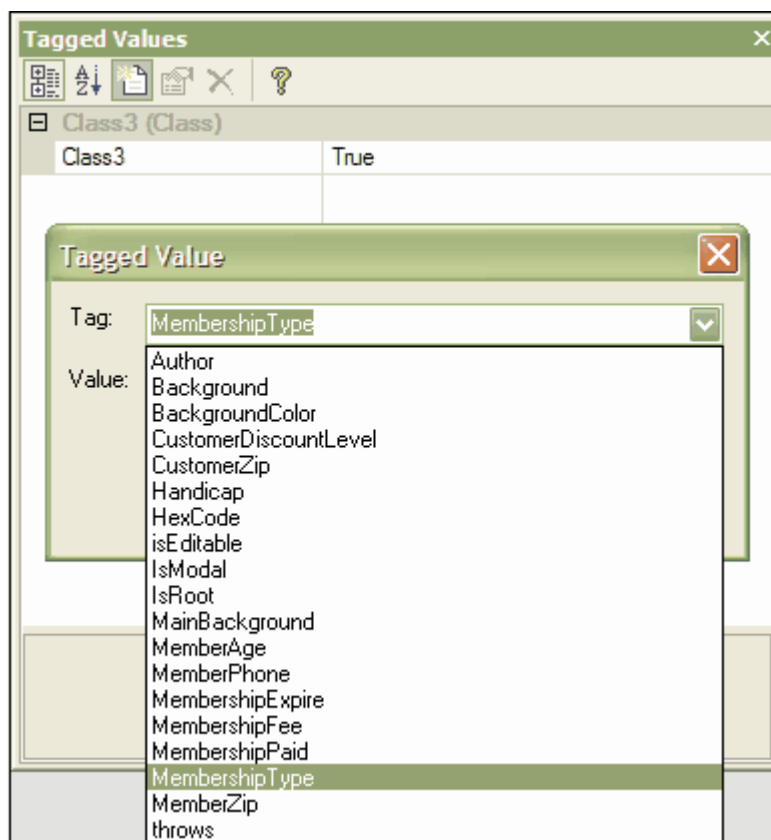


4.7.11.4 Assigning defined Tagged Value to an Item

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

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

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



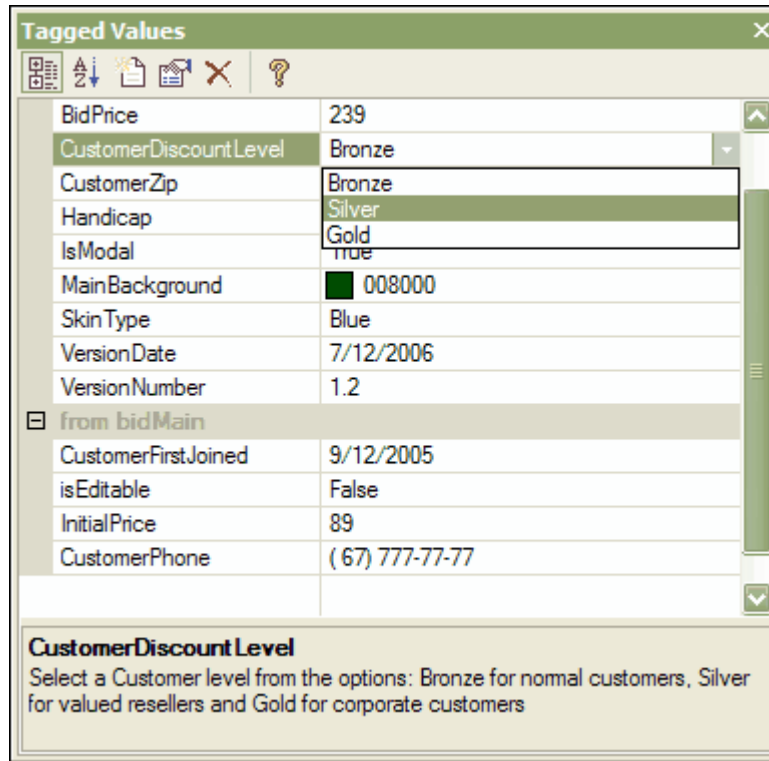
6. To confirm the selection of the tagged value, press the *OK* button.

Modifying Tagged Values with the Tagged Values Window

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

1. Ensure that the *Tagged Values* Window is visible by opening the *View* menu, mousing over the *Other Windows* item and selecting *Tagged Values* from the submenu. Alternatively press *Ctrl + Shift +6*.
2. Select the model feature that you wish to edit the tagged values, this will show all of the tagged values for the selected model feature.

3. Edit the fields as appropriate. The information entered may only reflect the masked value types that have been defined either as a predefined type or in the format defined by the creation of the Custom tagged type.
4. The example below shows a predefined Enum type having its value modified.

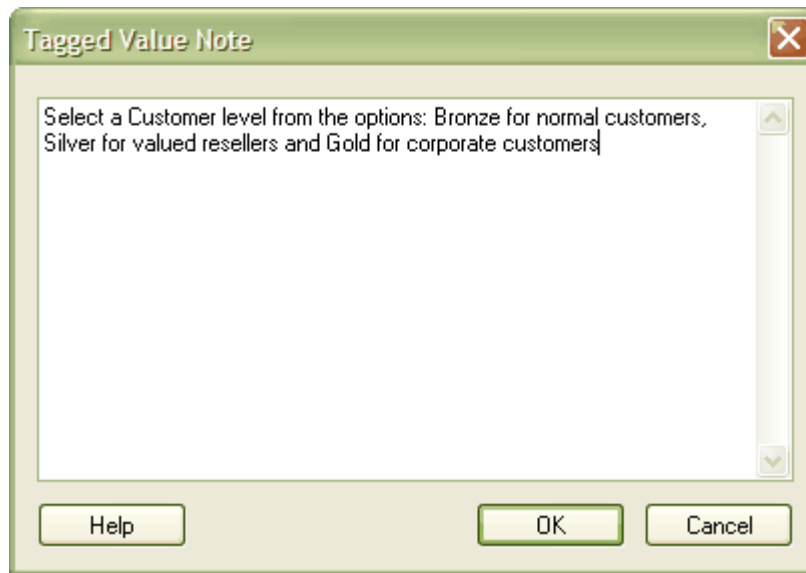


Note: To override a Tagged Value defined in a parent element, edit the value in the from [parentname] compartment of the Tagged Values dockable window, once this has been done the tag will be moved into the selected elements tagged values and this will not effect the tagged values defined in the parent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

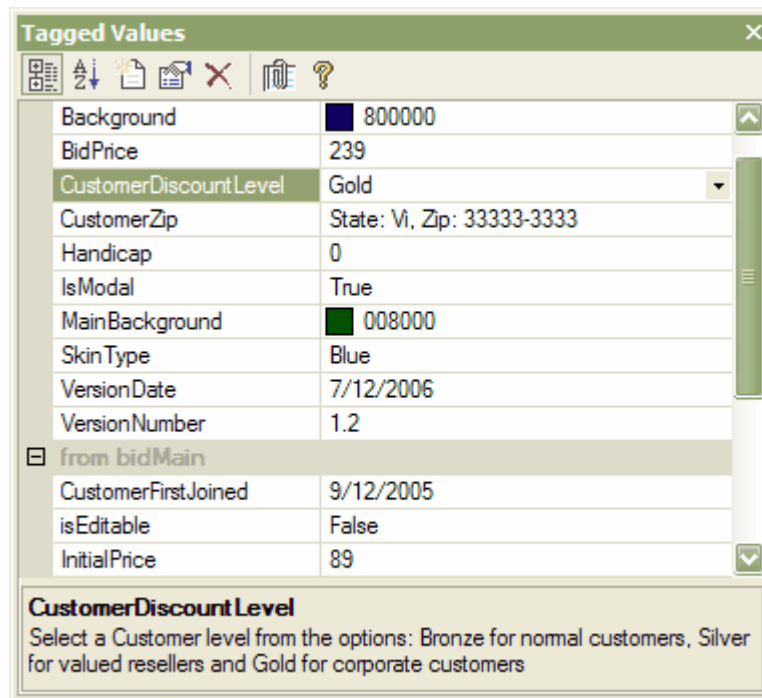
4.7.11.5 Assigning Information to a Tagged Value

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

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



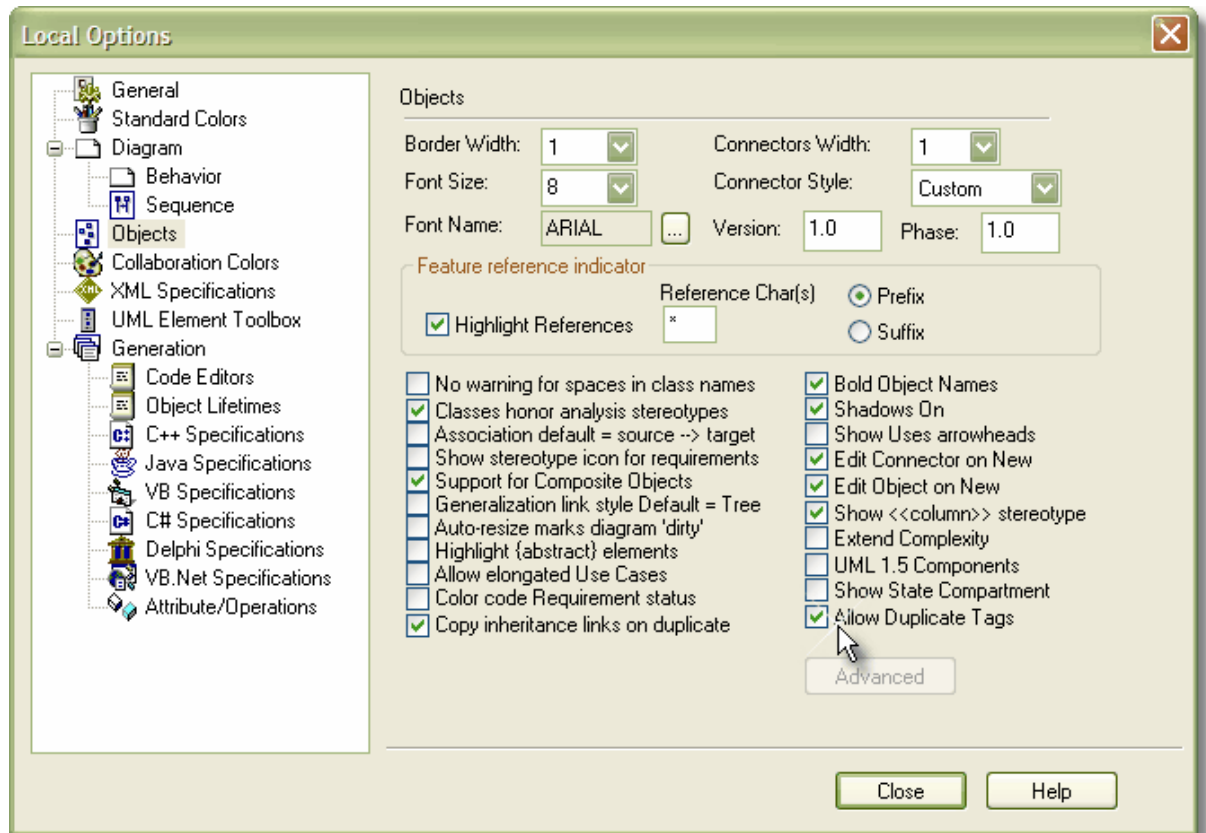
5. Enter the information relating to the tagged value into the *Note* field, this information will be displayed in the lower portion of the *Tagged Values* dockable window whenever the tagged value is selected.



4.7.11.6 Allow Duplicate Tags

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

1. Go to the *Tools* menu and select *Options*.
2. This will open the *Local Options* dialog, from the hierarchical tree select the *Objects* item.
3. Ensure that the *Allow Duplicate Tags* item is checked.



4.7.12 Project Management Window

The Project Management window allows for the input of the resources, effort, risk and metrics that may be added to elements contained in the model.

Open the *Project Management* window by selecting *Resourcing Metrics & Risk* from the *Element* menu or open the *Project Management* window from the *View | Other Windows | Project Management* (or press the *Ctrl + Shift + 7* hotkey combination).

Resource	Role	Time	%Done	Start	End	Description
John	Java Programmer	10.000000	75	5/01/2005	13/01/2005	Implement Login Use Case
Phil Smith	Graphic Artist	1.000000	100	5/01/2005	5/01/2005	Create Webpage logo
John	WebMaster	0.000000	0	5/01/2005		based Accounts s

Right click on the list to view the context menu which allows you to add, modify and delete list items. For more information see the [Adding, Modifying and Deleting Tasks](#) topic.

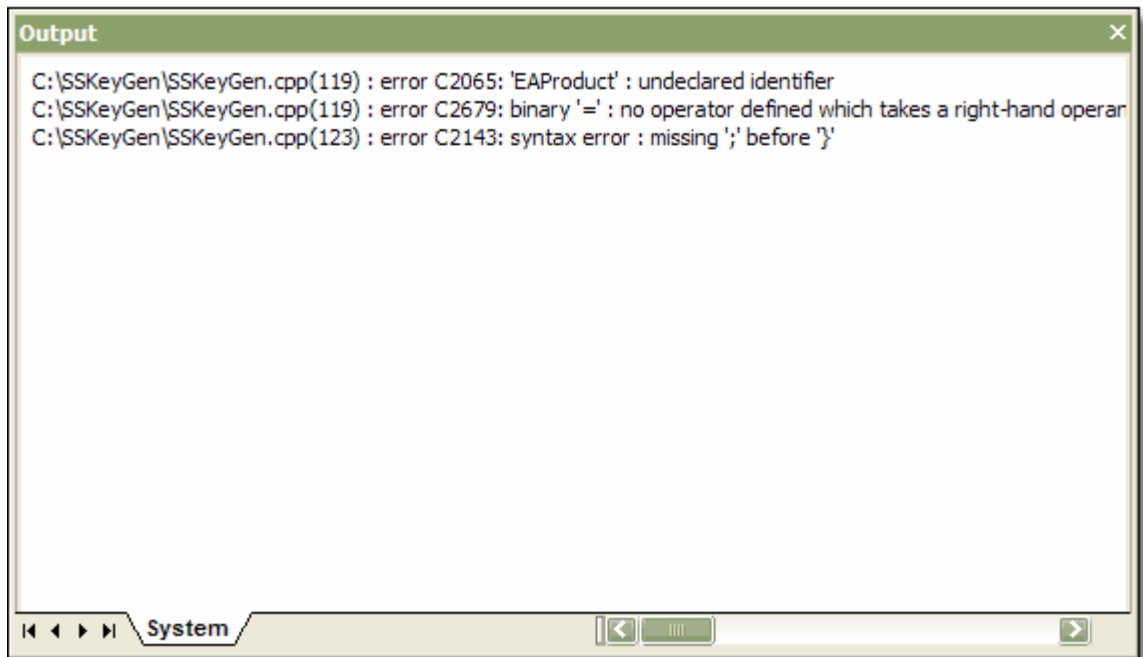
Tip: Select the *Print List* menu option to print out the currently displayed items.

See:

- [Resource Management](#)
- [Effort Management](#)
- [Risk Management](#)
- [Metrics](#)

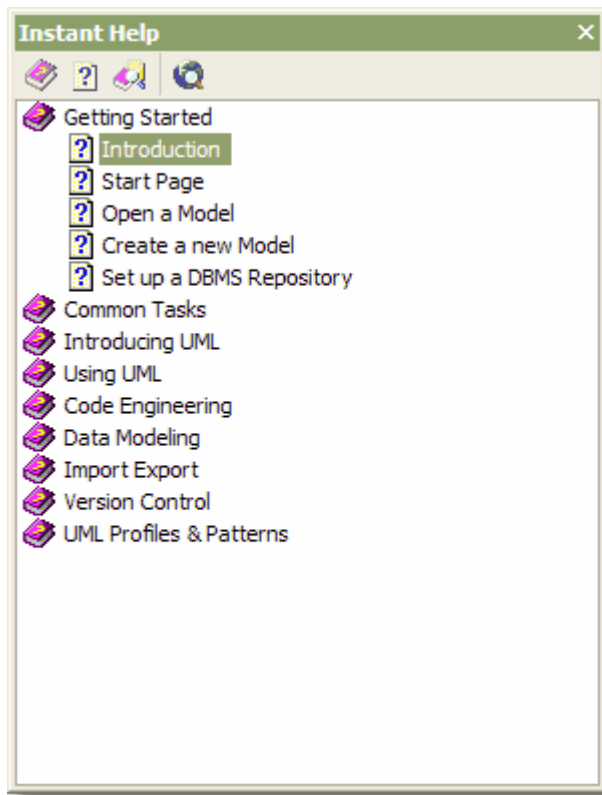
4.7.13 Output Window

The output window is used to display line items which may be either system generated or Add-In generated. For more information on using the Automation interface to allow Add-Ins make use the output window see the [Add-Ins | Repository](#) topic.



4.7.14 Instant Help Window





The Instant Help window allows access to the most popular help file reference topics. Double-clicking on a topic opens that topic in the main window of EA. Instant Help allows the user to interact with specific items in the EA help file, allowing access to the contents, search and index page of the EA help file. The Instant Help also allows the user to open an Internet search page within the Instant Help tab of the EA workspace.



Instant Help window menu buttons

The menu buttons in the Instant Help menu allow the user to interact with specific items in the EA help file and open a browser window from within EA. The table below details each button's functionality.



	The Open Help Contents button opens the help file in a separate window at the Introduction page.
	The Open Help Search button opens the help file in a separate window at the search page.
	The Help Index Page button opens the help file in a separate window at the Index page.
	The open Web Search button allows open a web search page within EA, the default page is www.google.com but the page may be altered to another web page from the Local Options dialog.

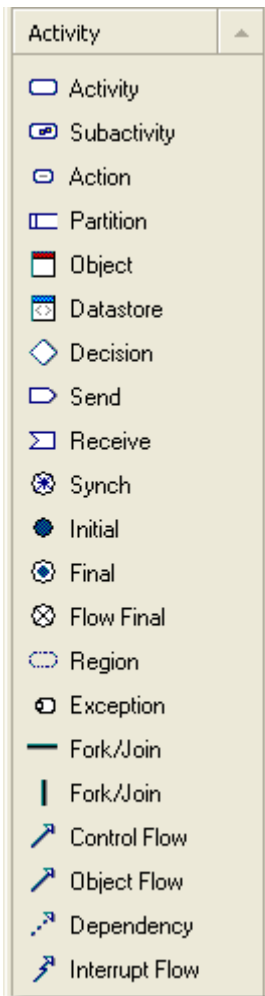
4.8 The UML Toolbox

The *UML Toolbox* is used to select element and connection types to be inserted into the current diagram. It is separated into tabbed categories, each with relevant UML elements. It is possible to add MDG technologies and Profiles to the UML toolbox to allow for custom stereotyped elements to be accessed from the UML

toolbox, for more information refer to the [Add MDG technologies to UML toolbox](#) and [Add Profiles to UML toolbox topics](#).

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

Tip: You may drag an element from the toolbox onto the diagram, or click on an element and place it using the mouse at a selected position.




To create a model element:

1. Select the appropriate category in the UML Toolbox, and select the desired element icon.
2. Click on the workspace to create and place a new element of that type in the current diagram (you may also drag the item to where you want it to appear in the diagram).
3. Enter a name when prompted.
4. Enter any other information, such as notes and version or phase number.
5. The elements size and position can be adjusted by highlighting the element, and either moving it or scaling it as desired.

Note: The toolbar contains several groups of elements. Each diagram has a default group which will be automatically selected when the diagram is opened. An element or connection from any group may be placed in any diagram, however - simply select the category you require. However, the functionality of some elements may not transfer over into a different diagram; a basic sense of UML should guide.

Note: The Package element when pasted into a diagram will create a new package in the Project Browser - and a default diagram of the same type as the one the package is pasted into.

Tip: You can hide and show the UML Toolbox using the  Hide/Show Toolbox button on the shortcut toolbar.

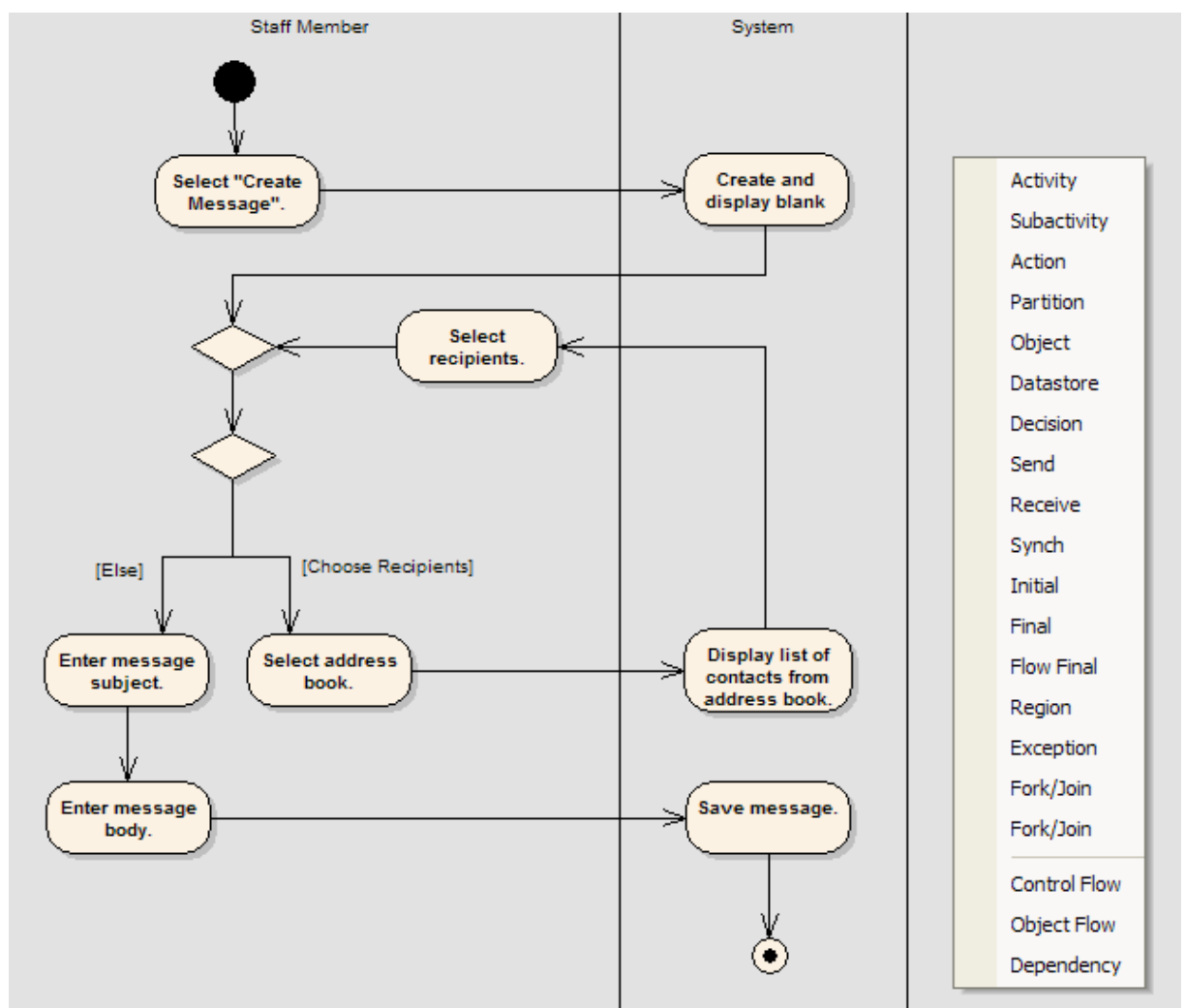
Warning: Make sure you have created and selected a current diagram first. You cannot create elements or connections until you create or select a current diagram from the Project Browser.

4.8.1 UML Toolbox Shortcut Menu

Instead of employing the full UML Toolbox users may access the UML Toolbox shortcut menu to add elements and links into a diagram. The items that are available for use with the Toolbox Shortcut menu are determined initially by the type of diagram being used or the current UML category type selected in the UML Toolbox. The advantage of using the UML Toolbox shortcut menu is that it allows for an increased amount of the workspace to be used for diagramming rather than being used to display menus.

To use the UML Toolbox Shortcut Menu use the following steps:

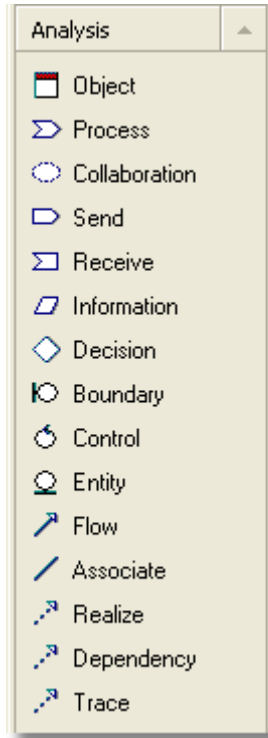
1. Open a diagram.
2. Hold down **Ctrl** key and right click the mouse.
3. Select the element or links from the menu that you wish to include in the diagram and click on the selection. The element or link will be added to the diagram.



4. If you require an element or link from a different UML category select a different element category on the UML Toolbox.

4.8.2 Analysis Group

Analysis type elements are used early in modeling to capture business processes, activities and general domain information. They are generally used in [Analysis diagrams](#).



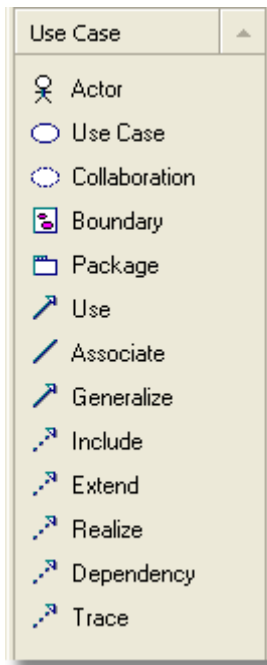
The elements and relationships in the Analysis group are used for early modeling of business processes, activities and collaborations. You can use stereotyped activities to model business processes, or stereotyped elements to capture standard UML business process modeling extensions such as *worker*, *case worker*, *entity*, *controller* and etc.

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

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

4.8.3 Use Case Group

Use Case elements are used to build [Use Case models](#). These describe the functionality of the system to be built, the requirements, the constraints and how the user will interact with the system. Often sequence diagrams are associate with Use Cases to capture work flow and system behavior.



The Use Case group is used to model the system functionality from the perspective of a system user. The user is called an Actor and drawn as a stick figure, although the user may be another computer system or similar. A use case is a discrete piece of functionality the system provides that allows the user to perform some piece of work or something of value using the system.

Examples of use cases are: login, open account, transfer funds, check balance and logout; each of these implies some purposeful and discrete functionality the system will provide a user.

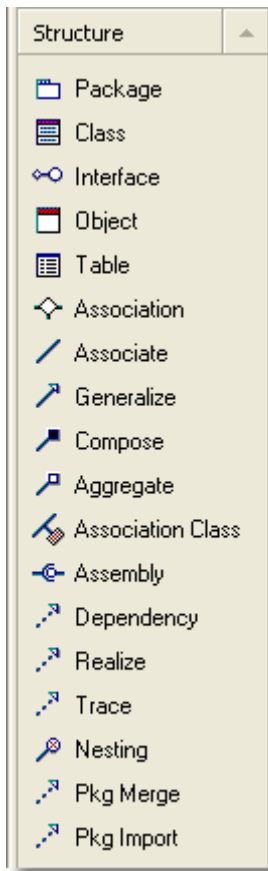
The connections available are: association (an actor uses a use case), extend (one use case can extend another), include (one use case can include another) and realize (this use case may realize some business requirement)

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

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

4.8.4 Structure Group

The *Structure* group can be used for [Package diagrams](#), [Class diagrams](#) and [Object diagrams](#) - those which usually display elements concerned with the logical structure of the system. These include objects, classes and interfaces. Logical models may include domain models (high level business driven object model) to strict development class models (define inheritance, attributes, operations, etc).



The Structure group is used for creating class models and database models. Class modeling is done using the Class and Interface elements, as well as occasional use of the Object element to model class instances. You can add association, inheritance or aggregation relationships. See the Class Diagram for an example of this.

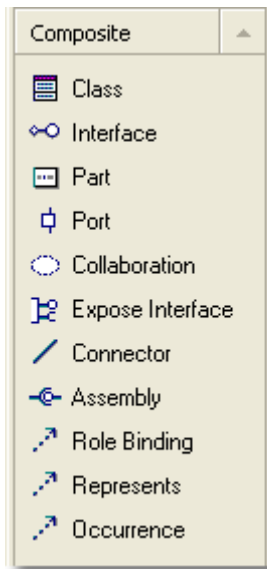
Use the Table element to insert a stereotyped class for use in database modeling. See the section on data modeling for more details.

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

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

4.8.5 Composite Group

The *Composite* group is used for [Composite Structure diagrams](#). Composite Structure diagrams reflect the internal collaboration of classes, interfaces, or components to describe a functionality, and to express run-time architectures, usage patterns, and the participating elements' relationships, which static diagrams may not show.

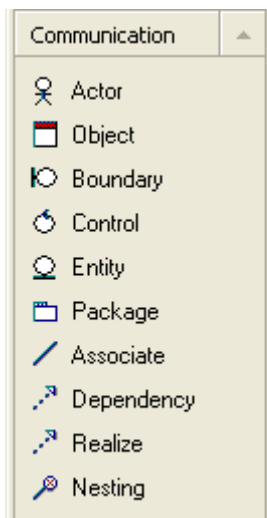


To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

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

4.8.6 Communication Group

The *Communication* group is used to model dynamic interactions between elements at run-time. The actor element models a user of the system, while the other elements model things within the system - including standard elements (rectangular element), user interface component (circle with left positioned vertical bar), controller (circle with arrow head in top most position) and entity (circle with bar at bottom).



Communication diagrams are used to model work flow and sequential passing of messages between elements in real time. They are often placed beneath Use Case elements to further expand on use case behavior over time.

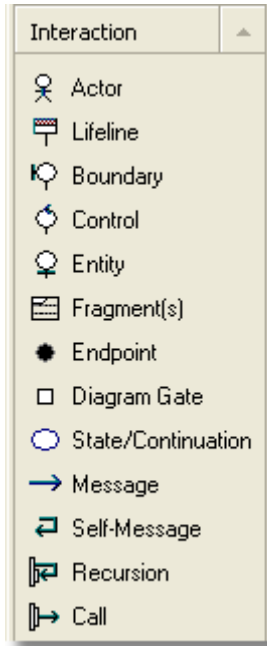
To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

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

Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

4.8.7 Interaction Group

The *Interaction* group is used for [Interaction diagrams](#), which are used to model work flow and sequential passing of messages between elements in real time. They are often placed beneath Use Case elements to further expand on use case behavior over time.



The Interaction group is used to model dynamic interactions between elements at run-time. The actor element models a user of the system, while the other elements model things within the system - including standard elements (rectangular element), user interface component (circle with left positioned vertical bar), controller (circle with arrow head in top most position) and entity (circle with bar at bottom). The meaning of these symbols is discussed further in the section on [Sequence diagrams](#). The message relation is used to model the flow of information and processing between elements.

The following model elements are supported: Actor (sequence element), element (sequence element), Boundary (sequence element), Control (sequence element), Entity (sequence element), Message and Iteration boundary.

Note: Messages may be simple or recursive calls.

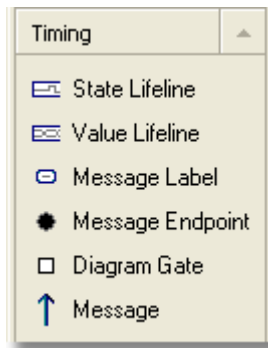
To add an element to the current diagram, left click the required icon, and drag it into position on the diagram. Set an element name and other properties as prompted.

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

4.8.8 Timing Group

The *Timing* group is used solely for [Timing diagrams](#), which use a time-scale to define the behavior of objects. The time-scale visualizes how the objects change state and interact over time. Timing diagrams can be used for defining hardware-driven or embedded software components, also time-driven business processes.

Timing diagrams can be used for defining hardware-driven or embedded software components, also time-driven business processes.



A lifeline is the path an object takes across a measure of time, indicated by the x-axis.

A [State Lifeline](#) follows discrete transitions between states, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations.

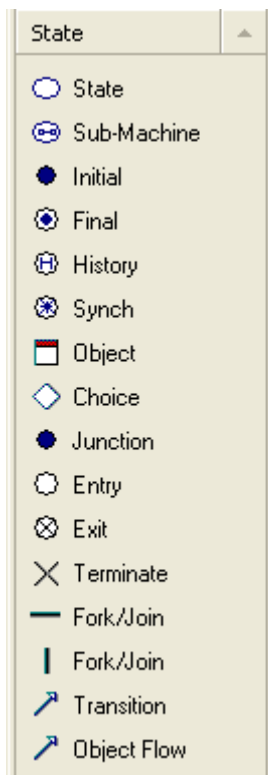
The [Value Lifeline](#) shows the lifeline's state across the diagram, within parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

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

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

4.8.9 State Group

The [State](#) group is used by [State Machine diagrams](#) to show the allowable states a class/element may be in and the transitions from one state to another. These diagrams are often placed under a class element in the Project Browser to illustrate how a particular element changes over time.



The State group provides elements common to State Machine diagrams - basically the State, start and end nodes and the Flow relation. State Machine diagrams are used to model the states or conditions that elements at runtime may be in - eg. active, inactive, idle, accelerating, braking, etc.

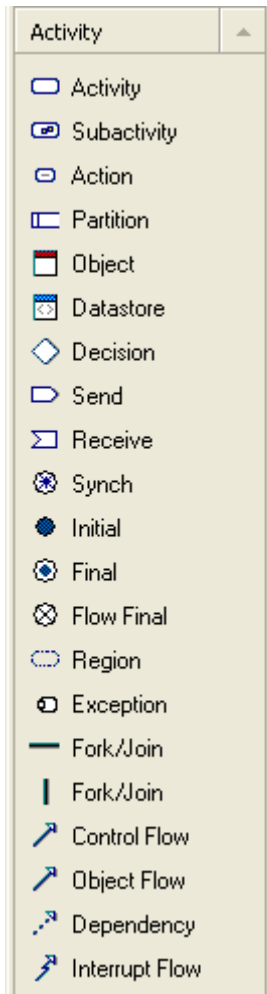
This group of icons allows you to add elements and relations to the current diagram. Click on the required element or connection to add to the current work. An element will be automatically inserted and ready for naming and properties to be set. A connection requires you to click on the start element and drag to the end element.

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

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

4.8.10 Activity Group

The **Activity** group is used to model system dynamics from a number of viewpoints in [Activity diagrams](#) and [Interaction Overview diagrams](#). An activity is some work that is carried out - it may overlap several use cases or form only a part of one use case. Send and receive events are included as triggers. A decision element marks a point where processing may split based on some outcome or value. The Flow relation models an active transition and synch points are used to split and rejoined periods of parallel processing.



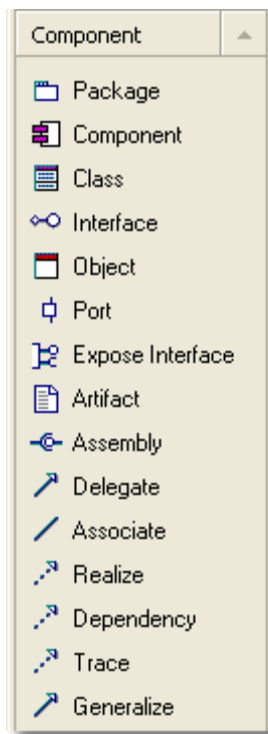
Activity elements allow you to describe the **dynamics** of the system from the point of view of **activities** and **flows** between them. Activities may be stereotypes as a "process" to display a business process icon.

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

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

4.8.11 Component Group

The **Component** group allow you to model the physical components of your system in a [Component diagram](#). A component is a piece of hardware or software that makes up the system, for example, a DLL or Web Server are examples of Components that may be deployed on a Windows 2000 Server (Node). See the Deployment Diagram for an example of this.



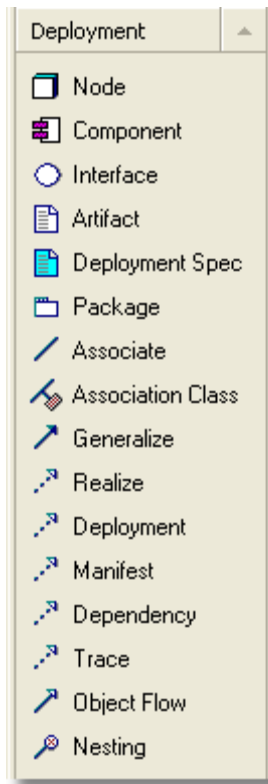
The Component group contains elements related to the actual building of the system – the components that will make up the system (e.g. ActiveX DLL's or Java beans), the Interfaces they expose and the dependencies between those elements.

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

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

4.8.12 Deployment Group

The **Deployment** group allow you to model the physical components and deployment structure of your system in a **Deployment diagram**. A component is a piece of hardware or software that makes up the system, and a Node is a physical platform on which the component will exist. For example, a DLL or Web Server are examples of Components that may be deployed on a Windows 2000 Server (Node). See the Deployment Diagram for an example of this



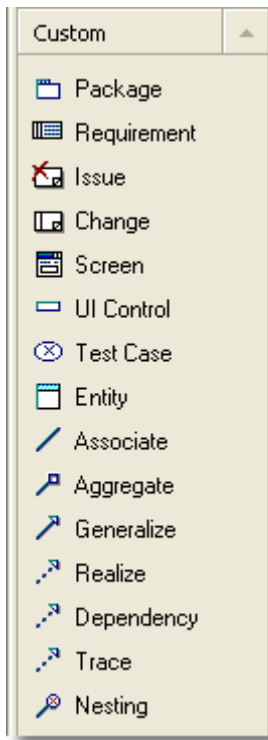
The Deployment group contains elements related to the actual building of the system – the components that will make up the system (e.g. ActiveX DLL's or Java beans) and the nodes those components will run on - including the physical connections between nodes.

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

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

4.8.13 Custom Group

The *Custom* group contains a few extended UML elements that may be of use in modeling or designing your system in a [Custom diagram](#).



An **Entity** is a stereotyped element that represents any general thing not captured by the element or class type elements (for example a trading partner).

A **Screen** provides a stereotyped class element that displays a GUI type screen – this can be used to express application GUI elements and flows between them.

A **UI control** likewise can be used to express GUI controls.

A **Requirement** is a custom element used to capture requirements outside of standard UML elements. A requirement expresses required system behavior that may cross several use cases. You may link requirements to other elements using the realize link to express the implementation of a requirement and hence the traceability from user needs to what is being built.

An **Issue** is a custom element used to capture model issues (such as defects and bugs).

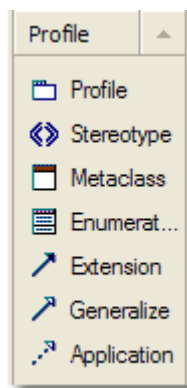
A **Change** is a custom element used to model change requirements

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

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

4.8.14 Profile Group

The **Profile** group contains some extended UML elements that may be of used to create and modify Profiles for the rapid creation of stereotyped and tagged values that may be applied to elements, attributes, methods, links, etc..



A **Profile** is used to provide a generic extension mechanism for building UML models in particular domains. They are based on additional Stereotypes and Tagged values that are applied to Elements, Attributes, Methods, Links, Link Ends, etc.

A **Stereotype** provides a mechanism for varying the behavior and type of a model element.

A **Metaclass** is used to create a class whose instances are classes, a metaclass is typically used to construct metamodels.

An **Enumeration** creates a class stereotyped as enumeration, which is used to provide a list of named values as the range of a particular type.

Note: The Profile section of the UML toolbox is not shown by default. To show this section, select **Tools | Options | UML Element Toolbox** on the main menu, tick **Profile** at the bottom of the **Visible Elements** list, then press the **Save Folder List** button. Close the **Local Options** window. The Profile section should now show in the UML toolbox as below

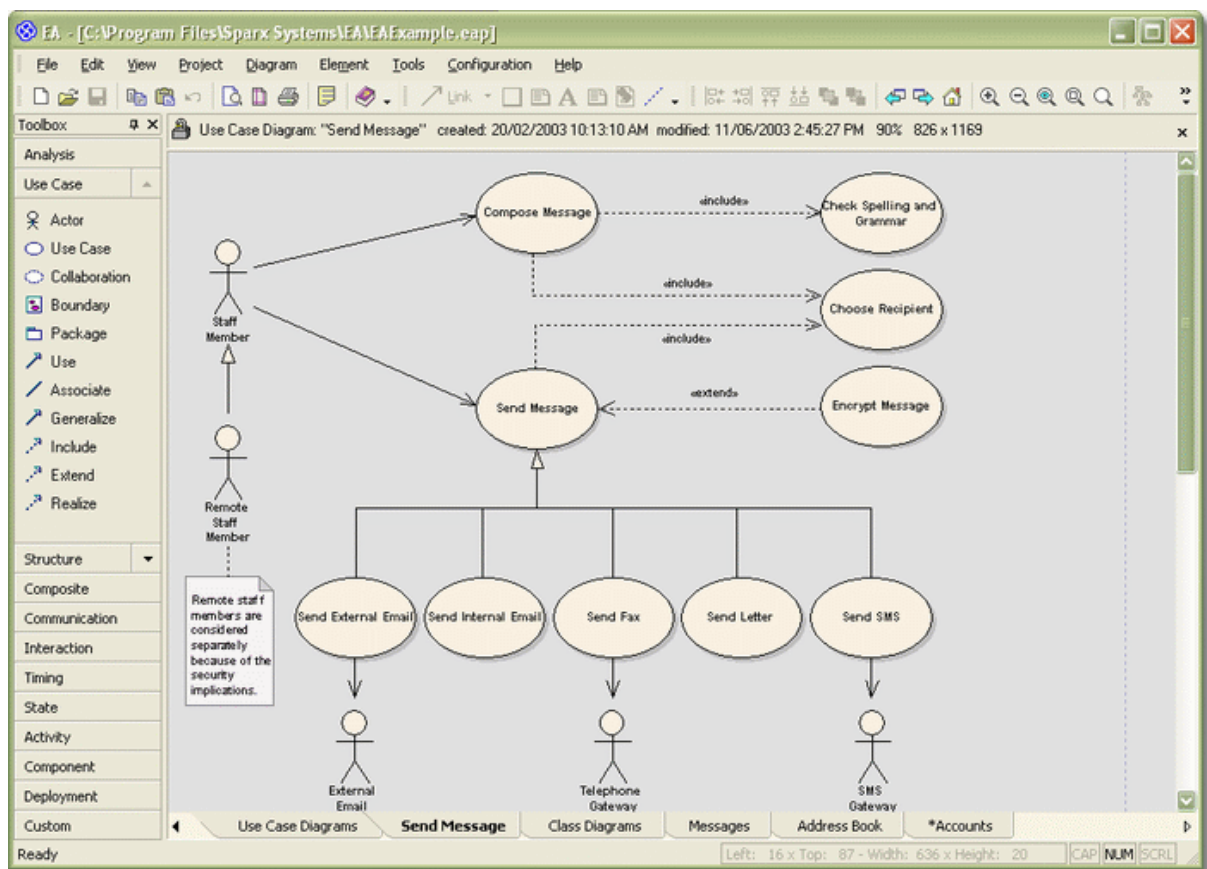
4.9 View Options

You view your model in two ways in the application workspace - either in [Diagram View](#) or [Report View](#).

4.9.1 Diagram View

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

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



Typical diagram activities include:


- Print and print preview diagrams
- Add new elements to the diagram using the UML Toolbox
- Add connections between elements using connections from the UML Toolbox
- Set diagram properties
- Save the diagram image to file
- Save the diagram image to the clipboard
- Copy elements in a diagram to link or copy elsewhere

- Zoom diagram to different magnifications
- Use the toolbar forward and back arrows to load previous and next diagrams
- Align and size multiple elements
- Delete elements from the diagram (but not the project)
- Double click with the left mouse button on the diagram background to open the diagram property dialog
- Right click on the diagram background to open the context menu

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














4.9.2 Report View

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

To access the report view, load a diagram, then press the report view button  on the shortcut toolbar.

The Report View may be sorted by any column (ascending or descending) by clicking in the column header. Columns may be moved left or right, however the order you place them in will not be retained in subsequent sessions.

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

Deployment Diagram: "Seattle Office Deployment Model" created: 24/02/2003 4:42:11 PM modified: 22/05/2003 5:31:32 PM								
Object	Type	Stereotype	Scope	Status	Phase	Created	Modified	
 SE0002	Node	pc server	Public	Proposed	1.0	19/02/2003 11:19:45 AM	5/06/2003 10:19:56 AM	
 Internet	Package		Public	Proposed	1.0	20/02/2003 1:25:20 PM	10/06/2003 8:08:47 PM	This package represents the Internet.
 LAN	Package	subsystem	Public	Proposed	1.0	20/02/2003 1:29:19 PM	10/06/2003 8:08:01 PM	This subsystem represents all the nodes that make up a local area network. It includes nodes such as hubs, routers, computers and network adaptors.
 SE0001	Node	pc server	Public	Proposed	1.0	24/02/2003 4:45:56 PM	4/06/2003 3:24:23 PM	
 192.168.0.1	Node		Public	Proposed	1.0	24/02/2003 4:52:19 PM	22/05/2003 5:25:13 PM	
 216.239.46.185	Node		Public	Proposed	1.0	24/02/2003 4:52:25 PM	22/05/2003 5:31:32 PM	
 192.168.0.2	Node		Public	Proposed	1.0	24/02/2003 4:54:51 PM	22/05/2003 5:29:00 PM	
 Seattle Client 01	Node		Public	Proposed	1.0	24/02/2003 5:02:54 PM	22/05/2003 5:29:00 PM	
 Seattle Client 02	Node		Public	Proposed	1.0	24/02/2003 5:04:12 PM	22/05/2003 5:29:00 PM	
 Seattle Remote Client 01	Node		Public	Proposed	1.0	24/02/2003 5:04:46 PM	22/05/2003 5:25:13 PM	
 Seattle Remote Client 02	Node		Public	Proposed	1.0	24/02/2003 5:05:29 PM	22/05/2003 5:25:13 PM	
 Note	Note		Public	Proposed	1.0	11/04/2003 1:30:49 PM	22/05/2003 5:25:13 PM	
 Note	Note		Public	Proposed	1.0	22/05/2003 5:29:00 PM	22/05/2003 5:31:32 PM	These are generic examples of remote clients.

The **Report View** is a convenient overview of all elements in a diagram in text form. Activate the **Report View** from the shortcut toolbar.

4.10 Package Tasks

A **Package** is a container of model elements, and is displayed in the Project Browser using the 'folder icon' familiar to Windows users. This section explores the tasks you can perform with packages, including:

- [Open a package](#)
- [Add a package](#)
- [Rename a package](#)
- [Drag a package onto a diagram](#)
- [Show or hide a package](#)
- [Delete a package](#)

See also [The Project Browser](#)

4.10.1 Open a Package

Open a Package

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

To open a package:

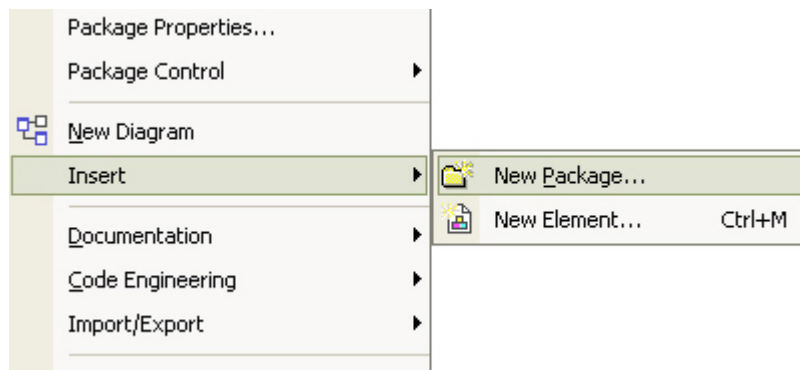
- Double clicking on a *package* will open it and display any contents in the [Project Browser](#) (or tree).
- Alternatively, you can click once on the + and - symbols next to the folder icon to open or close the package respectively.

Tip: Package contents are arranged alphabetically and elements may be dragged from one folder to another using the mouse.

4.10.2 Adding Packages

To add a new package

1. Select the package or view in the [Project Browser](#) under which you wish to add a new package.
2. Right click the mouse on the folder icon withing the Project Browser to open the context menu.
3. Select **Insert -> New package...**



4. In the dialog box, fill in the name of the new package and press **OK**.

The new package will be inserted into the tree at the current location.

Note: A package may also be added using the UML Toolbox - and pasting a new package element into a diagram. In that case the package is created under the diagram's owning package, and will be created with a default diagram of the same type as that the Package is created in.

Tip: Note that in a multi-user environment, other users will not see the change until they reload their project.

4.10.3 Renaming Packages

To rename a package

1. Select the package you wish to rename in the [Project Browser](#).
2. Right click to open the context menu.
3. Select the *Package Properties* option.
4. Enter the new name in the *Name* field.
5. Press *OK*.

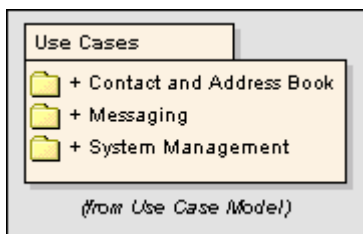
Alternatively, highlight the package you wish to rename, and press *F2*

Note: In a multi-user environment, other users will not see the change until they reload their project.

4.10.4 Drag a Package onto a Diagram

You can drag a package element from the Project Browser onto the current diagram. This will display the package and any contents within. This is a useful feature to help organise the display and documentation of models.

The illustration below displays how a package will be displayed in a diagram - note the Actor and child package icons.

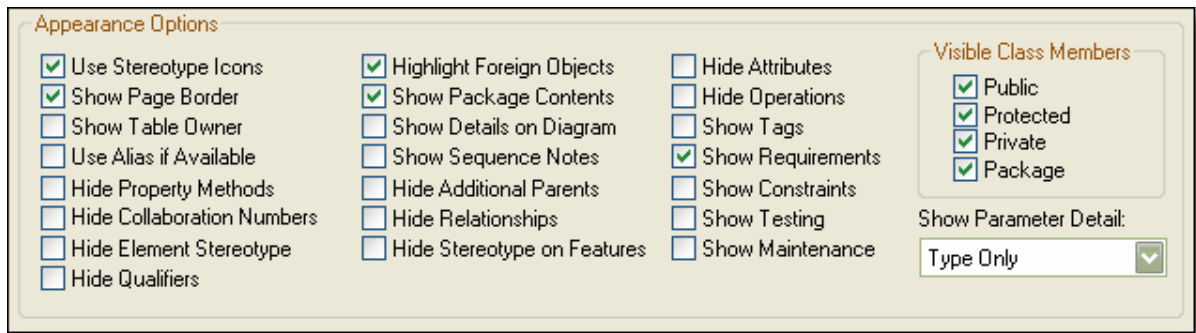


See also: [Show or Hide Package Contents](#)

4.10.5 Show or Hide Package Contents

You can show or hide the contents of packages in a diagram. To do so, follow these steps:

1. Load a diagram.
2. Double click in the background area to open the diagram properties window.
3. Check or clear the *Show Package Contents* box as required.
4. Press *OK*.



You can also configure the following settings in this window:

- Hide relationships
- Hide operation parameters
- Highlight linked elements

4.10.6 Delete a Package

You can delete a model element by highlighting it in the [Project Browser](#), then right clicking to open the context menu.

Select the **Delete** option. You will be asked to confirm your request – press **OK** to proceed.

Warning: *Deleting a package deletes all contents of the package as well, including sub-packages and elements. Make very sure that you really want to do this before proceeding.*

Note: *In a multi-user environment, other users will not see the change until they reload their project.*

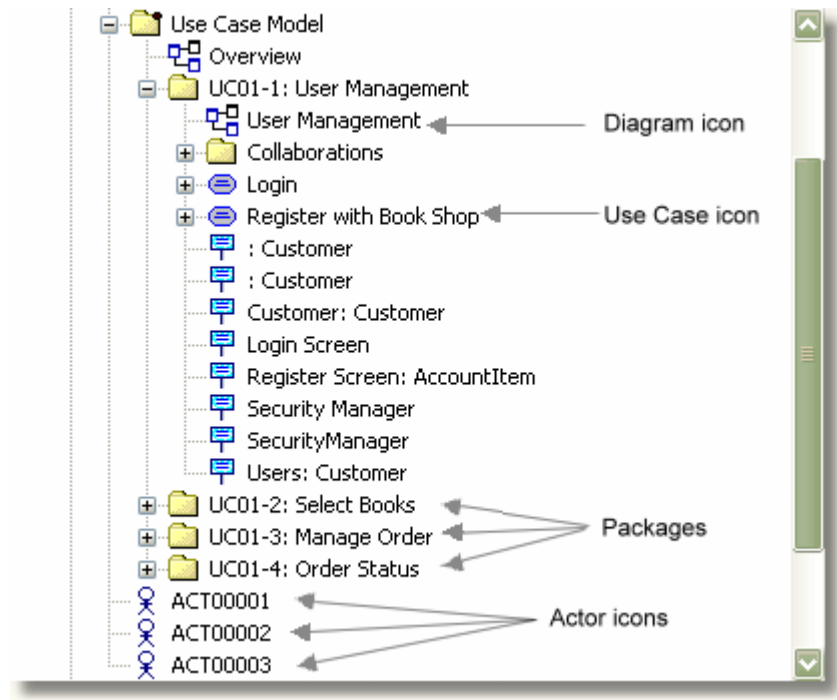
4.11 Diagram Tasks

This section explores the tasks you can perform with diagrams, including:

- [Select a diagram](#)
- [Add a diagram](#)
- [Zoom a diagram view](#)
- [View last and next diagram](#)
- [Set diagram page size](#)
- [Scale image to page size](#)
- [Lock diagram](#)
- [Show or hide attributes and operations](#)
- [Layout a Diagram](#)
- [Set appearance options](#)
- [Undo last action](#)

4.11.1 Select a Diagram

Diagrams are used to display model elements (elements, use cases, components, etc.).



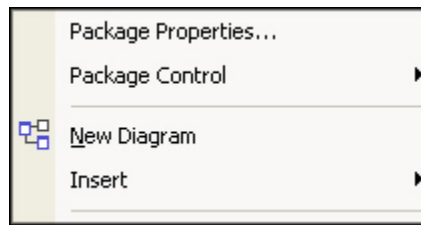
To open a diagram that already exists, follow these steps:

1. Search the *Project Browser* until you locate the diagram you require. You may need to explore several levels before you find your diagram.
2. Double click on the *diagram icon*.
3. Enterprise Architect will open the diagram and display any model elements in the diagram view.

4.11.2 Add a Diagram

To add a new diagram, follow these steps:

1. In the *Project Browser* window, select the appropriate package or element under which you wish to place the diagram.
2. Right click to open the context menu and select *New Diagram*.
3. Enter the name of the new diagram in the dialog provided and select the type of diagram you require.
4. Press *OK* to create your new diagram.



Alternatively, select *New Diagram* from the Diagram menu.

Note: The type of diagram type determines the default toolbar associated with the diagram and whether it can be set as a child of another element in the Project Browser (eg. a sequence diagram under a use case)

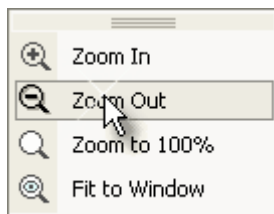
See also: [UML Diagrams](#).

4.11.3 Zoom a Diagram View


You can zoom into and out from a diagram view using the zoom buttons on the diagram toolbar, or by using the *View | Zoom* submenu.



Change the zoom level by 10% by pressing either the *Zoom In (+)* or *Zoom Out (-)* buttons. The same is achieved by selecting *Zoom In* or *Zoom Out* from the *View | Zoom* submenu.



There are two ways to return the diagram to 100%:

- Click the  button OR
- Select *Zoom to 100%* from the *View | Zoom* submenu.

Note: At high levels of zoom, element features will cease to display. This is because of the difficulty the Windows font mapper has in choosing a font for extreme conditions, and the result can look odd.

4.11.4 View Last and Next Diagram

Enterprise Architect retains a list of recently visited diagrams. This is useful for stepping backwards and forwards through the list of recently accessed diagrams.



To view the previous or next diagram use the *Forward* or *Next* buttons on the diagram toolbar.

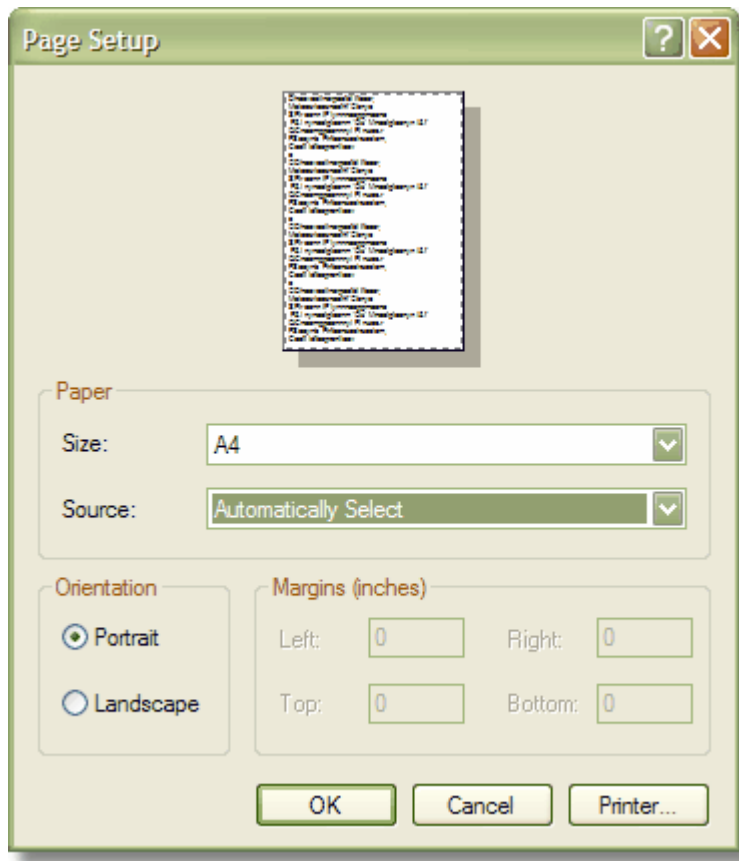
There is also a *Home* button which may be used to go to the [default project diagram](#) (if one has been specified).

4.11.5 Set Diagram Page Size

You can change the size of the diagram area (or scrollable/printable area) using the diagram properties window.

To set the page size, follow these steps:

1. Load a diagram.
2. Double click on the background to open the properties dialog.
3. In the Page Size section press the *Advanced* button.
4. Select an appropriate page size from the dropdown list and then choose the appropriate Orientation of the page that you intend to print.
5. Press *OK*.



The scrollable area for your diagram will be expanded or reduced accordingly. Note that when you print or print preview, the output is cropped to the bounding rectangle for the diagram.

Setting the Default Paper Size for New Diagrams

You can set the default paper size for new diagrams on the View/Options/Diagram options dialog. Once set there, all new diagrams will have that as the default size.

See [Diagram Settings](#).

4.11.6 Scale Image to Page Size

You may scale a diagram image to fit the currently selected page size, scale the diagram image to fit the current page size or define a custom scaling layout.

The default diagram image setting is to fit the size of the currently selected printer paper. The image will not be scaled up to fill the page, but will be scaled down if it exceeds the current page boundary. Also, the image will retain its current dimensions - that is it will be scaled down equally in the X and Y dimensions.

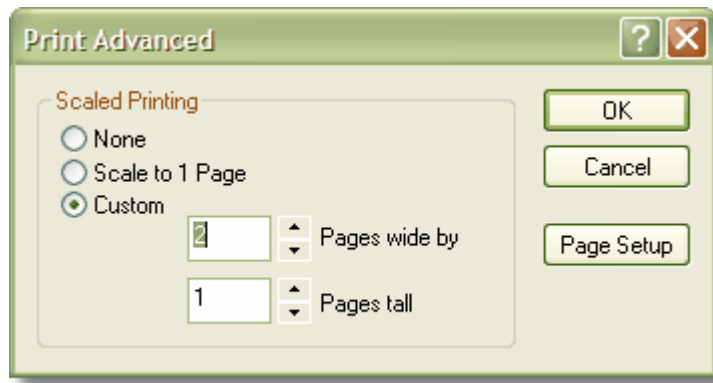
Scale image option

To access the image scaling options, follow these steps:

1. Select the diagram to scale.
2. Double click on the diagram background to get the properties dialog, or select the properties option from the context menu.



3. Under the *Page Size* Heading press the *Advanced* button.



4. From the Print Advanced Dialog the following options are available:
 - *None*, the none option prints on as many pages as the diagram image covers.
 - *Scale to 1 page*, this option scales the diagram image to fit on the currently selected page.
 - *Custom*, the custom setting allows the user to specify the width and height of the diagram images across a specified amount of pages
5. *Page Setup* allows the user to select the Page size and alignment.

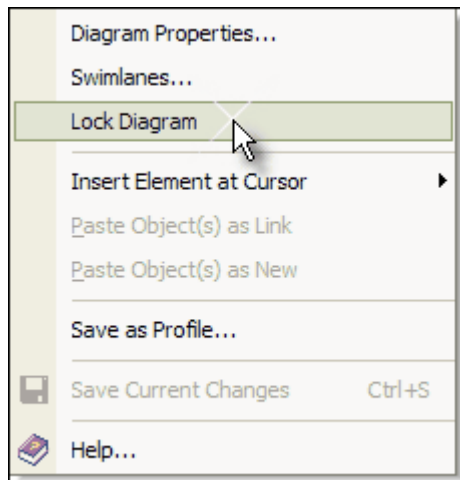
Note: Before printing, make sure you have selected the required Page Layout from the Page Setup button.

4.11.7 Lock Diagram

You can lock a diagram against inadvertent changes - moving or sizing elements, for example.

To lock a diagram, follow these steps:

1. Open the diagram you wish to lock.
2. Right click on the background to open the diagram context menu.
3. Click the *Lock Diagram* option to prevent further changes.



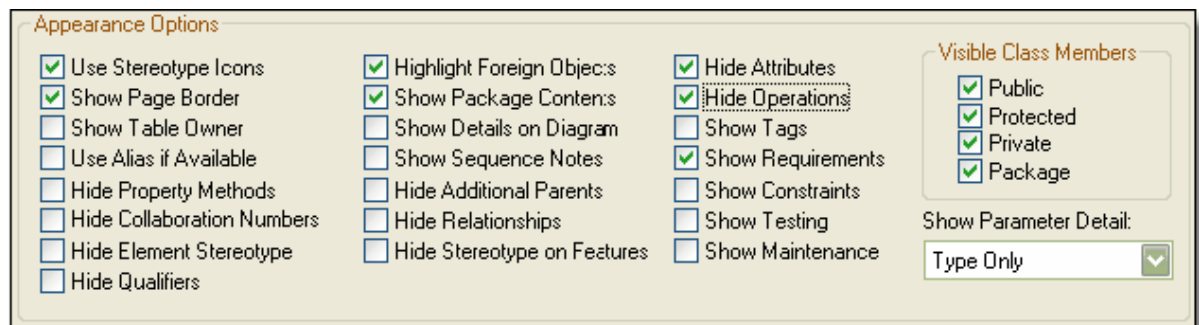
4. Press **OK**.

4.11.8 Show or Hide Attributes and Operations

You can show or hide attributes and operations within classes on a diagram.

To show or hide attributes and operations, follow these steps:

1. Open a diagram.
2. Double click on the diagram background to open the properties dialog.
3. In the **Appearance Options** section, check the **Hide Attributes** and/or **Hide Operations** as appropriate.



4. Press the **OK** button to confirm the selections.

4.11.9 Layout a Diagram

A facility is provided to layout diagrams automatically. This will attempt to create a reasonable tree based structure from the diagram elements and relationships in a diagram. Owing to the complexity of many diagrams, the results may need some manual 'tweaking'.

Layout a Diagram

To layout a diagram, follow the steps below:

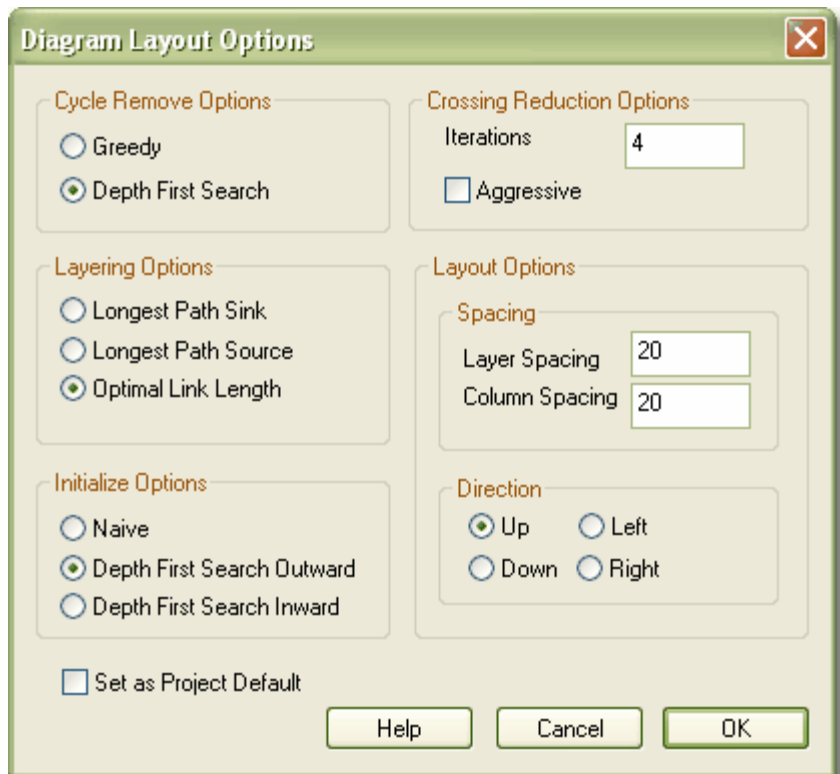
1. Select a diagram.
2. From the *Diagram* menu, select *Layout Diagram* -OR- use the *Auto Layout* button on the diagram toolbar .

Note: *Dynamic and analysis diagrams are NOT suited to this form of layout - please ensure that the diagram type you are laying out will benefit from the action first.*

Accessing the Diagram Layout Options Dialog

For a fine degree of control of the elements in your diagram, you can use the *Diagram Layout Options* dialog. Generally the default layout parameters provide adequate layouts for a wide range of diagrams, but there are times when more specific settings are required. To access the *Diagram Layout Options* dialog, follow the steps below:

1. Open the diagram's *Properties* dialog by double clicking on the background of the diagram.
2. Press *Set Layout Style* to open the *Diagram Layout Options* dialog.
3. When you have made any required changes, press *OK* to save.



You can alter any of these settings on the *Diagram Layout Options* dialog to refine your layout:

Cycle Remove Options

These settings determine the type of cycle removal to be used during layout.

Greedy - Uses the Greedy Cycle Removal algorithm.

Depth First Search - Uses the Depth First Search Cycle Removal algorithm.

Layering Options

These settings determine the type of layering to be used during layout.

Longest Path Sink - Uses the Longest Path Sink Layering algorithm.

Longest Path Source - Uses the Longest Path Source Layering algorithm.

Optimal Link Length - Uses the Optimal Link Length Layering algorithm.

Initialize Options

These settings determine the type of initialization of indices and columns to be used during layout.

Naive - Uses the Naive Initialize Indices algorithm.

Depth First Search Outward - Uses the Depth First Out Initialize Indices algorithm.

Depth First Search Inward - Uses the Depth First In Initialize Indices algorithm.

Crossing Reduction Options

Iterations - Enter the number of iterations to be used during cycle removal.

Aggressive - This option allows you to specify whether or not to use an aggressive (read time-consuming) crossing reduction step. If it is checked, the aggressive crossing reduction will be used. If it is unchecked, the aggressive crossing reduction will not be used.

Layout Options

Layer Spacing - Enter the default number of logical units between layers.

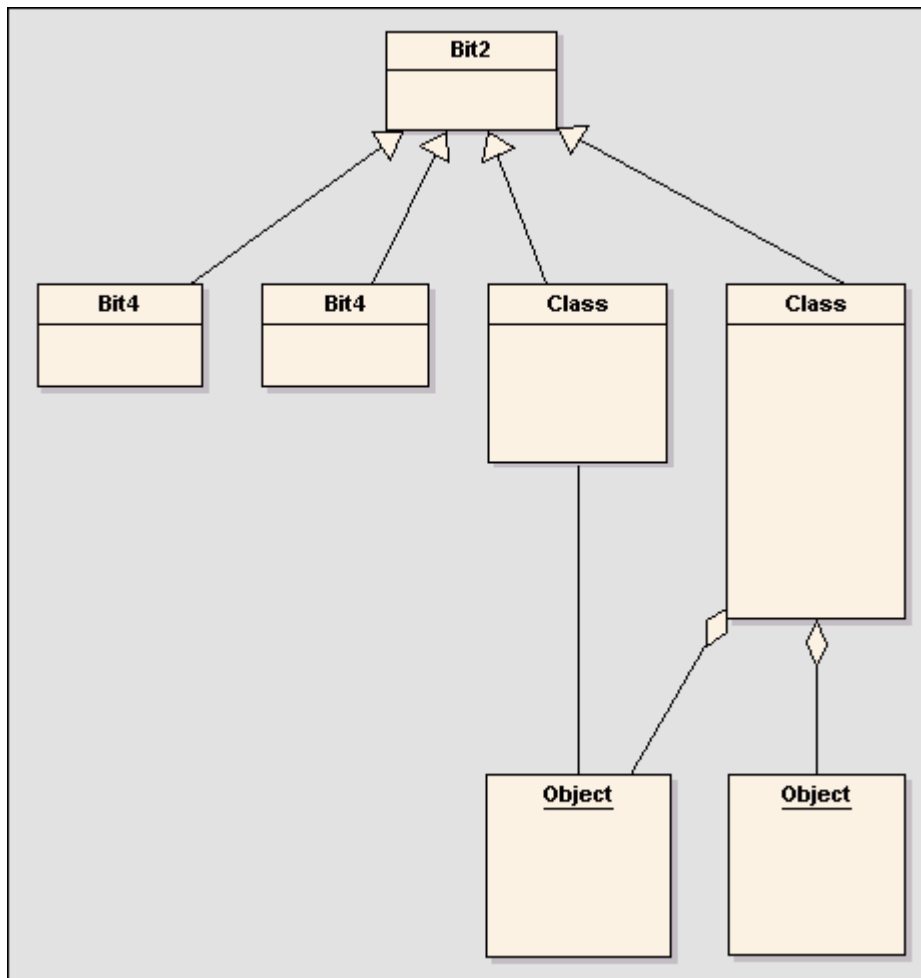
Column Spacing - Enter the default number of logical units between columns.

Direction - Set the direction that directed links should point - Up, Down, Right, Left.

Set as Project Default Checkbox

Check this to apply the Diagram Layout settings to all diagrams in the project. If you then check this box and press OK for a different diagram, the new settings will override the settings saved earlier.

An example of an automatically laid out diagram:



4.11.10 Set Appearance Options

You can customise the appearance of a diagram in a number of ways. The following options can be set on the diagram dialog:

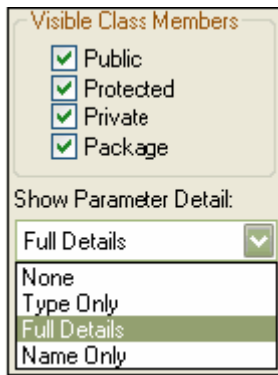
Appearance Options			
<input checked="" type="checkbox"/> Use Stereotype Icons	<input checked="" type="checkbox"/> Highlight Foreign Objects	<input type="checkbox"/> Hide Attributes	Visible Class Members <input checked="" type="checkbox"/> Public <input checked="" type="checkbox"/> Protected <input checked="" type="checkbox"/> Private <input checked="" type="checkbox"/> Package Show Parameter Detail: Type Only <input type="button" value="v"/>
<input checked="" type="checkbox"/> Show Page Border	<input checked="" type="checkbox"/> Show Package Contents	<input type="checkbox"/> Hide Operations	
<input type="checkbox"/> Show Table Owner	<input type="checkbox"/> Show Details on Diagram	<input type="checkbox"/> Show Tags	
<input type="checkbox"/> Use Alias if Available	<input type="checkbox"/> Show Sequence Notes	<input checked="" type="checkbox"/> Show Requirements	
<input type="checkbox"/> Hide Property Methods	<input type="checkbox"/> Hide Additional Parents	<input type="checkbox"/> Show Constraints	
<input type="checkbox"/> Hide Collaboration Numbers	<input type="checkbox"/> Hide Relationships	<input type="checkbox"/> Show Testing	
<input type="checkbox"/> Hide Element Stereotype	<input type="checkbox"/> Hide Stereotype on Features	<input type="checkbox"/> Show Maintenance	
<input type="checkbox"/> Hide Qualifiers			

The specific options available in the Appearance Options are:

Option	Description
Use Stereotype Icons	Toggle the display of stereotype icons in the current diagram. This only applies to stereotypes with icons internal to EA, i.e. Analysis stereotypes and Business Modeling stereotypes .
Show Page Border	Show a page border to align elements with.
Show Table Owner	Displays the Table Owner, more information relating to this Topic may be found in the Set Table Properties section.
Use Alias if Available	Use element alias as name if the alias is specified.
Hide Property Methods	Hide the property methods.
Hide Collaboration Numbers	Hide numbering in collaboration diagrams.
Hide Element Stereotype	Hide the element stereotype.
Hide Qualifiers	Hide the qualifiers from the current diagram.
Highlight Foreign Objects	Highlight objects from other packages.
Show Package Contents	Show icons for the contents of packages displayed in the current diagram.
Show Details on Diagram	Show some diagram details in a note on diagram.
Show Sequence Notes	Show the Sequence Notes on the current diagram.
Hide Additional Parents	Hide the explicit naming of parents in classes and interfaces.
Hide Relationships	Hide all relationships in the current diagram.
Hide Stereotype on Features	Hide all Stereotypes on features.
Hide Attributes	Hides the display of Attributes from the compartment enabled element, for more details see the Show or Hide Attributes and Operations topic.
Hide Operations	Hides the display of Operations from the compartment enabled element, for more details see the Show or Hide Attributes and Operations topic.
Show Tags	Show the tag compartment for elements that support it and have tags.
Show Requirements	Shows requirements in compartment enabled elements in a requirements compartment.
Show Constraints	Shows constraints in compartment enabled elements in a constraints compartment.
Show Testing	Shows test scripts in compartment enabled elements, for more details see the Show Testing Scripts topic.
Show Maintenance	Shows maintenance scripts in compartment enabled elements, for more details see the Show Maintenance Scripts topic.

4.11.10.1 Visible Class Members

The visible Class Members section of the diagram properties dialog Appearance Options is used to determine the visibility of class members and also determine the details shown for parameters. Use the checkboxes to select the visibility of class members. the options include Public, Protected, Private and Package.



Show Parameter Detail

The Show Parameter Detail dropdown allows the user to control the display of the Parameter details these are:

None	No details shown
Type Only	Shows only the type of parameter.
Full Details	Shows all of the details for parameters
Name Only	Shows the name of the parameter only.

4.11.11 Undo Last Action

When editing diagrams, Enterprise Architect supports multiple undo levels for moving, re-sizing and deletion of elements, and deletion of connectors.

There are three ways to undo the last action:

- Press **Ctrl-Z** OR
- Use the **Edit | Undo** menu option OR
- Use the **Undo** toolbar button (below)

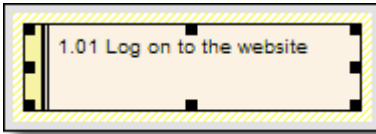


Warning: Currently the following cannot be undone:

- Element additions cannot be undone.
- Connector moves cannot be undone.

4.11.12 Highlight Context Element

You can show a hatched border around the selected element by checking the **Always Highlight Context Element** checkbox on the **Tools | Options | Diagram | Behavior** page. If you have this option checked, when you select an element it will appear like so:

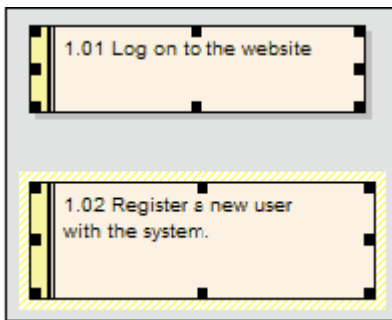


If you do not have this option checked, the selected element will not have a hatched border around it.

Multiple Selections

Whether you have the *Always Highlight Context Element* option selected or not, when you select multiple elements, one of the elements you select will have a hatched border. If you align the elements, this element will be the one used to align the other elements against.

For example, if the elements in the diagram below were to be aligned, the top element would align to the bottom element (the element showing a hatched border).



Changing the Element to Align Against

To change which element has a hatched border in a selected group - thus the element that will be aligned against - simply click on the element you want the other elements to align against.

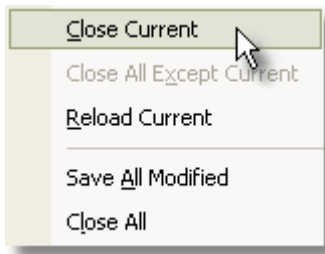
4.12 Diagram Views

This section explores optional available for viewing diagrams, including:

- [Close current diagram](#)
- [Closing all except the current diagram](#)
- [Reload current diagram](#)
- [Save all modified diagrams](#)
- [Closing all diagrams](#)
- [Zoom a diagram view](#)
- [View last and next diagram](#)

4.12.1 Close Current Diagram

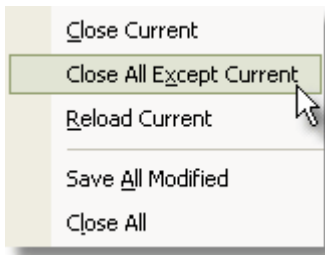
To close the view of the current diagram go to the *Diagram* menu and select *Diagram Views | Close Current*.



Alternatively right click on the context menu of the [diagram tabs menu](#) and select *Close <diagram name>*.

4.12.2 Close All Except Current Diagram

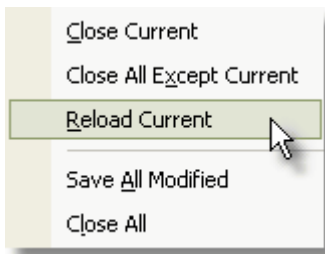
To close the view of all except the current diagram go to the *Diagram* menu and select *Diagram Views | Close All Except Current*.



Alternatively right click on the context menu of the [diagram tabs menu](#) and select *Close All Except <diagram name>*.

4.12.3 Reload Current Diagram

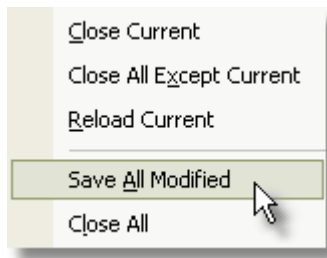
To reload the current diagram go to the *Diagram* menu and select *Diagram Views | Reload Current*.



Alternatively right click on the context menu of the [diagram tabs menu](#) and select *Reload <diagram name>*.

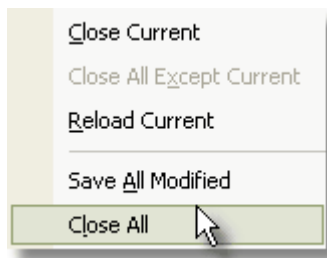
4.12.4 Save All Modified Diagrams

To save all of the diagrams that have been modified open the *Diagram* menu and select *Diagram Views | Save All Modified*.



4.12.5 Close All Diagrams

To close the view of all of the currently opened diagrams go to the *Diagram* menu and select *Diagram Views | Close All*.



Alternatively right click on the context menu of the [diagram tabs menu](#) and select *Close All*.

4.13 Element Tasks

This section explores the tasks you can perform with elements, including:

- [Add an element](#)
- [Connect elements](#)
- [Move elements and packages](#)
- [Auto element naming](#)
- [Search a project](#)
- [Default element template](#)

4.13.1 Add an Element

The UML Toolbox is used to select elements and connections to insert into the current diagram.

Click on an element type and drag it into the current diagram to create a new element of this type: you will be prompted for a name and any other settings you care to enter immediately.

Note: If there is no current diagram, no element will be inserted. You must first have created and opened a suitable diagram for your new model element.

The [UML Toolbox](#) contains a number of different element categories, press the named button for the group you need to select from. Each diagram has a default toolbar group and will automatically select that when loaded; you may however place an element or connection from any group in any diagram.

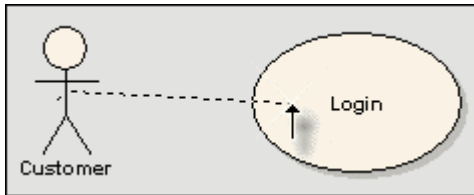
4.13.2 Connecting Elements

To connect classes, use cases, components or other elements, follow these steps:

1. Locate the type of association you want in the vertical toolbar - use case link, association, aggregation etc. Select (left click) the relationship.
2. Locate the start element in the diagram. Click on the element and hold down the left mouse button (the cursor should change to a vertical arrow).
3. Drag to the target element in the diagram and release the mouse button. While you do this a dotted line will indicate the relationship being created

Note: You may also drag a connection to position it: first click on the link, then hold the mouse button down and drag the link to where you want it to appear. Note that there are some limitations on how far or to where you can drag a connection.

Tip: Press the **F3** key to repeat the last connection you used.



Double click on an association to change its properties, or right click to change the connection type and direction if necessary.

In summary:

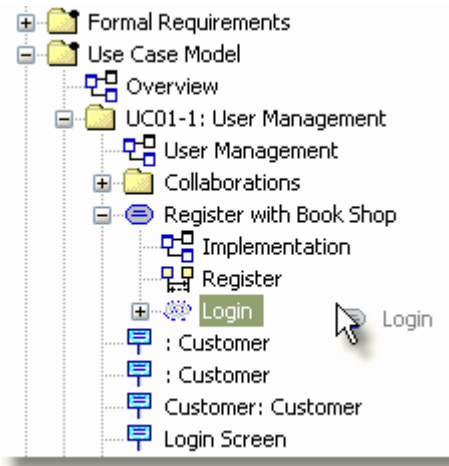
- select a link
- click on a start element
- holding the mouse button down, drag to the destination and release.

4.13.3 Moving Elements and Packages

Elements and packages can be moved from one package to another by dragging and dropping the source element to the target destination in the Project Browser. Note that if you move a package, ALL the child packages and their contents will be moved to the new location also.

To move a model element between packages, follow these steps:

1. Left click the model element in the Project Browser.
2. Holding down the mouse button, drag the element to the target package.
3. Ensure you have positioned the mouse cursor over a package icon.



4. Release the mouse button. The element will automatically be shifted.

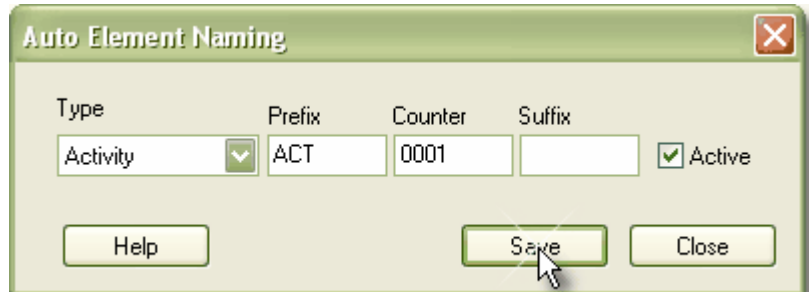
Warning: In a multi-user scenario, if one person moves or updates the Project Browser structure, other users will need to reload their project to see the latest changes in the Project Browser. Although this is true of any addition or modification to the tree, it is most important when big changes are made, such as dragging a package to a different location.

4.13.4 Auto Counters

The *Auto Element Naming* dialog allows you to configure automatic naming for any element type.

To set up auto naming, follow these steps:

1. Select the *Configuration | Auto Counters* menu option. This will open the *Auto Element Naming* window.



2. Select the element *Type* (eg. Activity) from the drop down menu.
3. Set the (optional) *Prefix* the new name will have.
4. Set the *Counter* value - use as many 0's as required to pad the name.
5. Enter an optional *Suffix*.
6. Tick the *Active* check box to turn auto naming on for this element type.
7. Press *Save*.

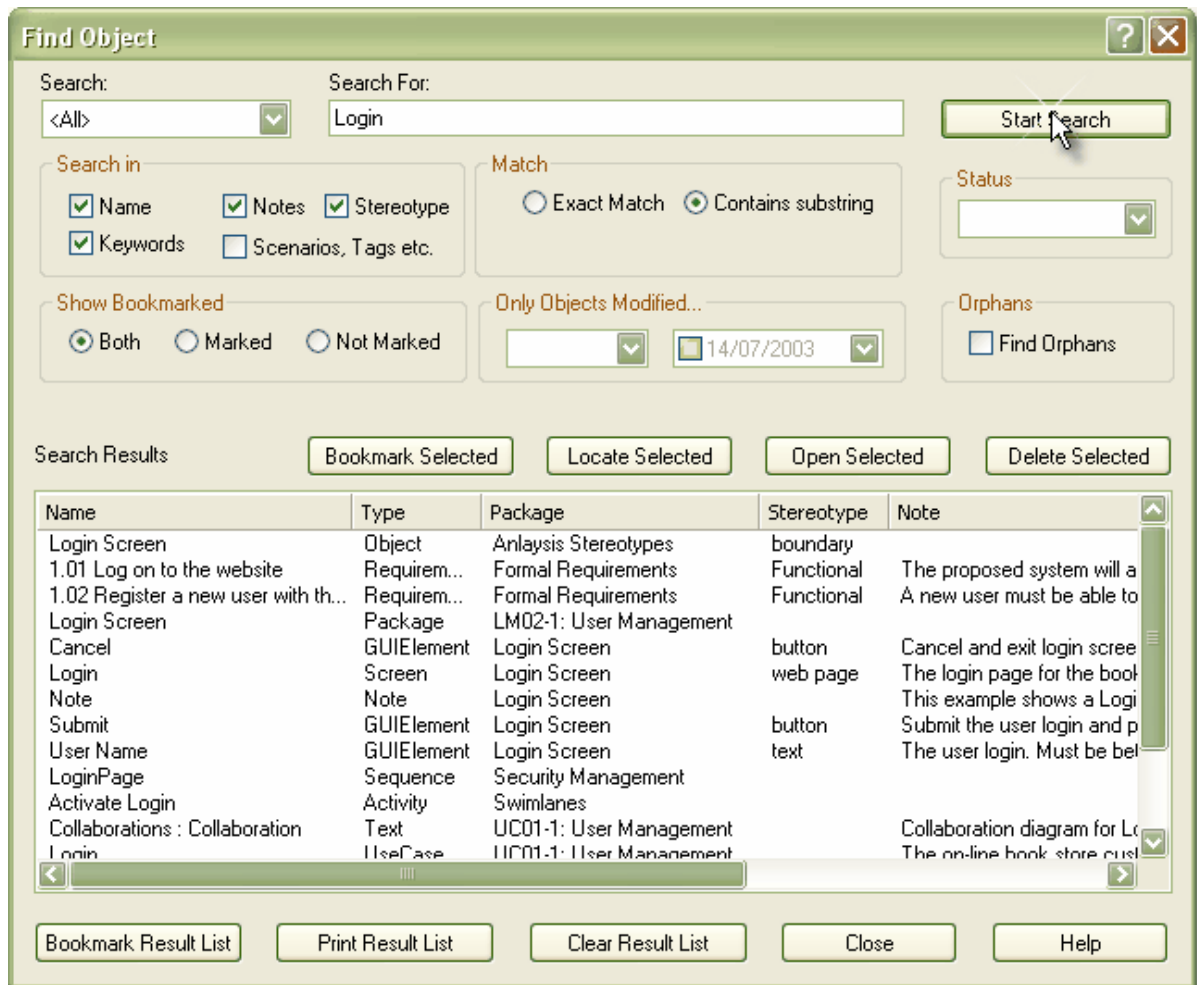
New elements of this type will now have an automatically generated name with an incrementing counter value.

4.13.5 Searching a Project

You may search your project for particular phrases or words. The search facility only checks elements - it does not search diagrams, requirements or other non-element text.

In addition to a standard search, you may also use *Find Orphans* option. This will locate all elements in the model that do not appear in any diagram. This is useful to find and utilize or delete elements that currently aren't displayed. When you check the *Find Orphans* checkbox, the other options are grayed out.

To open the *Find Object* dialog, go to *Edit | Find* on the main menu.



Once the search is complete you can:

- **Go to** a found element
- **Delete** a found element
- **Clear** the results
- **Get** element information

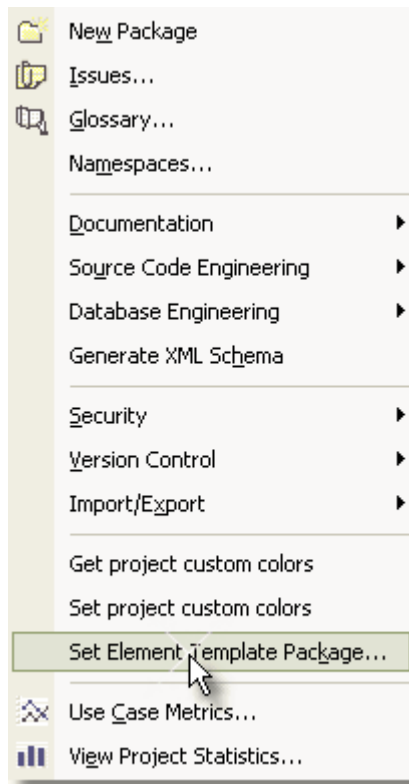
Element	Description
Search	What kinds of element to search.
Search For	The search string to find in the elements.
Search (button)	Start the search.
Search In	Which element fields to search (eg. name, stereotype, notes, tag).
Match	Select exact or like match.
Orphans	Find orphans only - these are element which do not appear in the Project Browser or any diagram.
Show Tagged	Show tagged elements only.
Open Element	Open the selected element from the result list.
Delete Element	Delete the selected element (from the project!).
Locate Element	Locate the selected element in the Project Browser.
Clear results	Clear the result list.

4.13.6 Default Element Templates

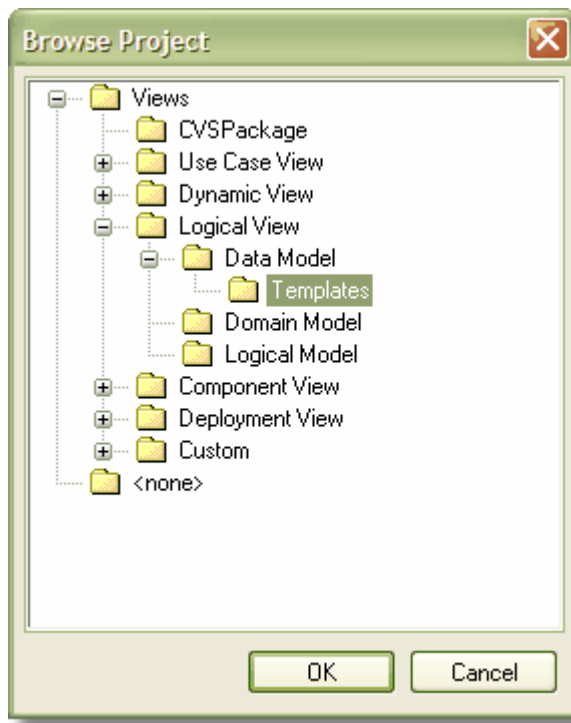
For control over the appearance of elements, the default element template can be set. This functionality could be used, for example, to denote different stages of a project. A template with different color fill could be created for each stage, for example, so that elements added in each stage are instantly identifiable as belonging to that stage. Element templates do not support the definition of stereotypes within the template. This means that when a stereotype is selected the stereotype settings will cause the template to revert to the default setting.

To configure the default element template, follow these steps:

1. First create a new package - this could be named "Templates" for example.
2. Within the Templates package create new diagrams - one for each type of diagram you wish to template. Name them so that they are easy to recognise eg. ClassTemplate for the template for class diagrams.
3. Add new elements to the template diagrams from the UML Toolbox and configure the size, appearance, notes, version, etc.
4. To set the templates as the default element templates, select the *Project | Set Element Template Package* option.



The *Browse Project* window is displayed.



5. Navigate to the *Templates* package. Select it and press *OK* to set it as the default element template.

Now each new element you add to your project will be created with the settings in the template diagrams. When you create elements, EA will check the templates directory first and if a template is found, it will copy the settings from there.

Note: *If you decide you do not wish to use the default element template, set the default element template to <none> in the *Browse Project* window.*

4.14 Element Inplace Editing Options

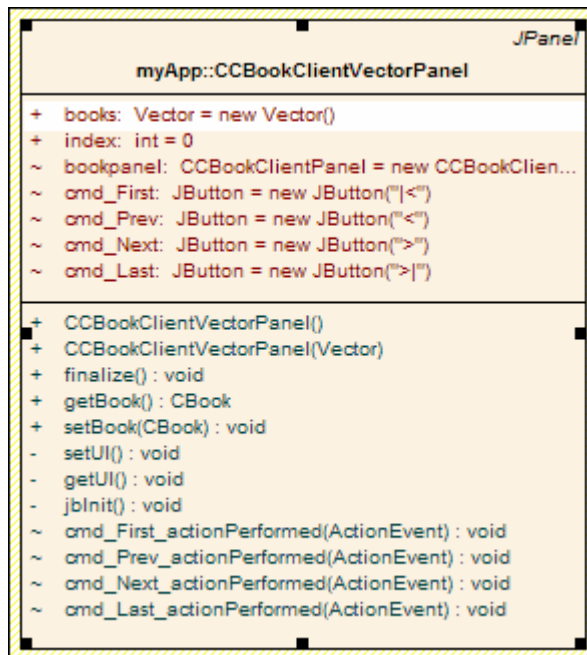
This section explores the tasks that can be performed using hotkeys and in place editing of elements. The tasks include:

- [Edit Name](#)
- [Edit Stereotype](#)
- [Edit Scope](#)
- [Edit Attribute keyword](#)
- [Edit Operation Parameter keyword](#)
- [Insert Operation Parameter](#)
- [Edit Parameter kind](#)
- [View Properties](#)
- [Insert New after Selected](#)
- [Add Maintenance Item](#)
- [Add Test Item](#)
- [Delete Selected from Model](#)

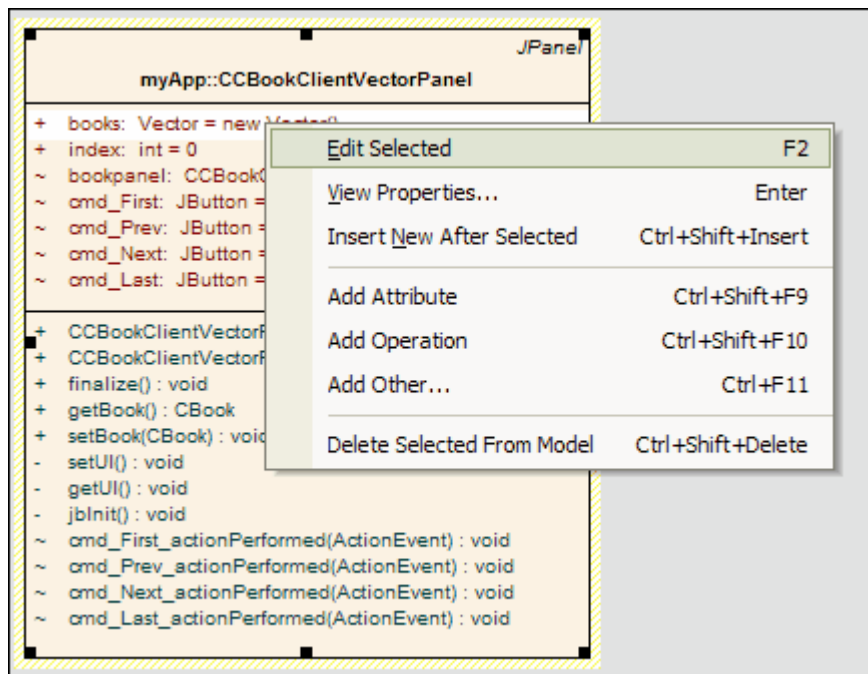
4.14.1 Inplace Element Item Tasks

To use the options that are available through the inline editing menu use the following steps:

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



- The items in the element can now be directly edited and manipulated, this may be achieved by either using the appropriate hotkey or by using right clicking on the highlighted item and choosing a task from the Element Items menu.



The following commands are available:

Menu Option	Hotkey	Description
Edit Selected	<i>F2</i>	Attach a note or attach a constraint to the connection.
View Properties	<i>Enter</i>	Opens up the dialog window containing details of the element.
Insert New After Selected	<i>Ctrl + Shift + Insert</i>	Inserts a new item to the element.
Add Attribute	<i>Ctrl + Shift + F9</i>	Adds an attribute to the element.
Add Operation	<i>Ctrl + Shift + F10</i>	Adds an operation to the element.
Add Other	<i>Ctrl + F11</i>	Allows the user to insert a feature on the specific element item, such as Maintenance features and Testing features.
Delete Selected from Model	<i>Ctrl + Shift + Delete</i>	Deletes the selected item from the model.
	<i>Ctrl + Shift + Arrow key</i>	Navigate Diagram Selection, this option allows users to navigate the diagram between elements without needing to use the mouse.
	<i>Shift + Enter</i>	Toggle element highlight option on and off

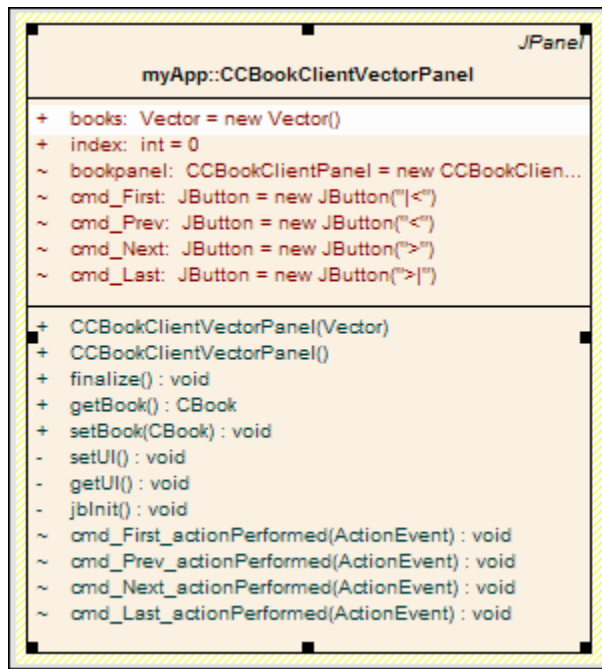
Other options that are available while in editing elements mode in a diagram (when an attribute or operation is highlighted):

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

4.14.2 Edit Element Name

By using the inline editing feature it is possible to change the name of an operation or attribute directly from the diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).

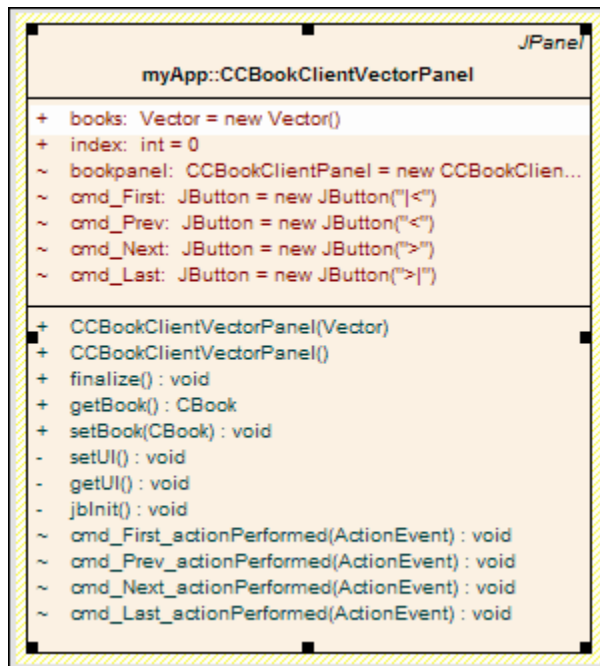


3. Right click the mouse and select *Edit Selected* from the menu or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Delete or type over the previous name to change the name of the attribute or operation. Then press the *Enter* key to accept the change or press the *Esc* key to cancel the change.

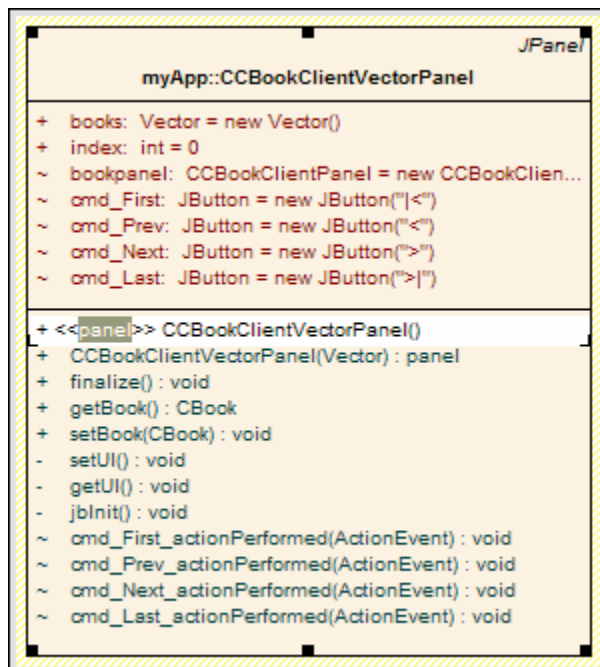
4.14.3 Edit Attribute or Operation Stereotype

By using the inline editing feature it is possible to change the Stereotype of an operation or attribute directly from the diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Move the cursor to the position before the name or within the existing attribute or operation Stereotype (denoted by << stereotype name >>).



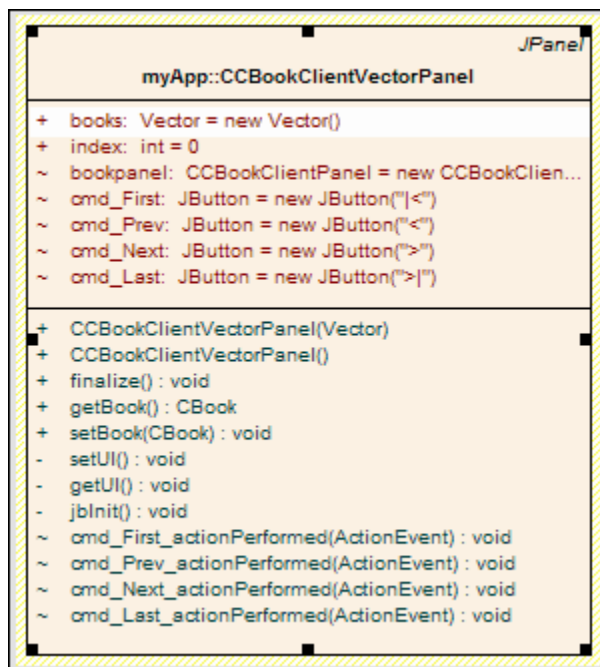
6. Delete or type over the previous Stereotype to change the Stereotype name of the attribute or operation.

Then press the *Enter* key to accept the change or press the *Esc* key to cancel the change.

4.14.4 Edit Attribute and Operation Scope

The inline editing feature allows the user to rapidly change the scope of an attribute or operation directly from the diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).

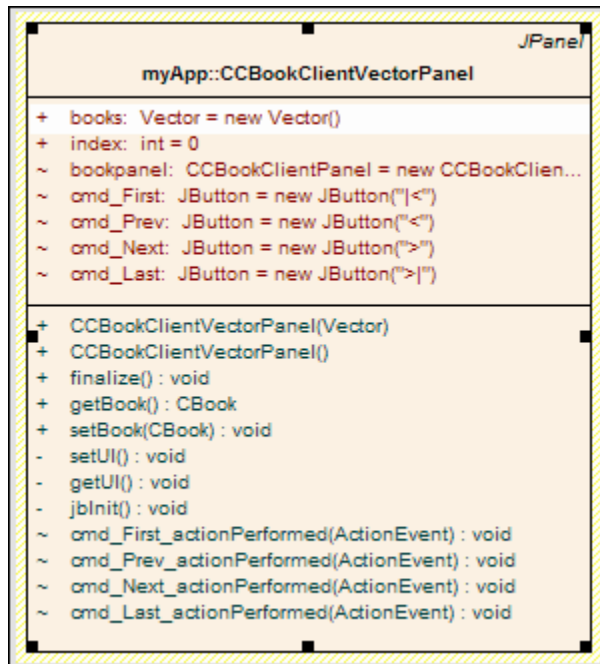


3. Right click the mouse and select **Edit Selected** or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Move the cursor to the attribute or scope of the item and delete the previous entry then reassign the entry by typing in the following symbols:
 - + indicates that the scope is Public.
 - - indicates that the scope is Private.
 - ~ indicate that the scope is Package.
 - # indicates that the scope is Protected.
6. Then press the *Enter* key to accept the change or press the *Esc* key to cancel the change. Once this procedure is completed the diagram will be updated to reflect the changes.

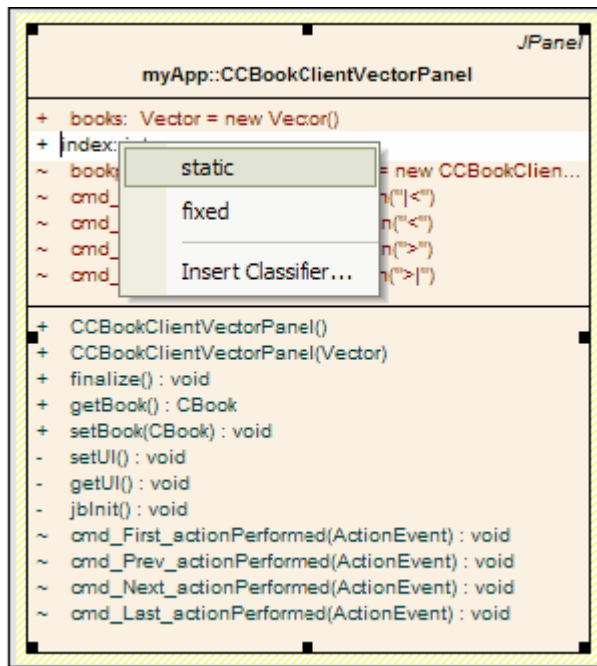
4.14.5 Edit Attribute Keyword

Features such as element keywords and classifiers may be directly added to an element by using the Element Keywords and Classifiers menu. This allows the user rapidly assign details on a per element item basis directly from a diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



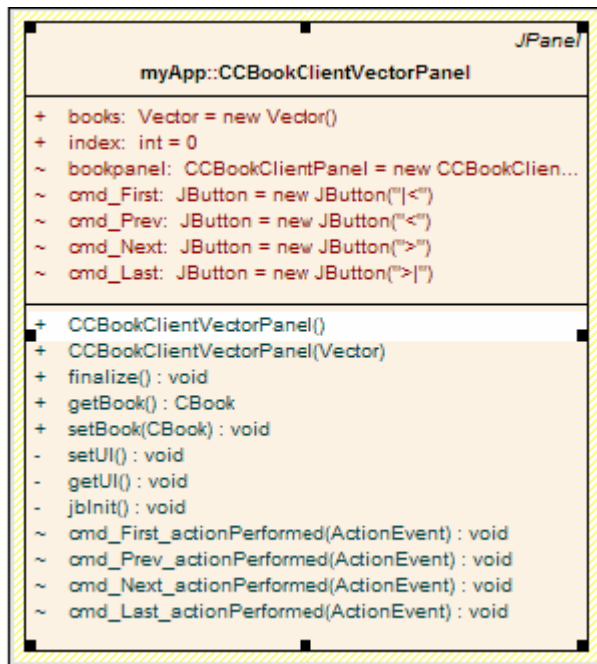
3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. If the item selected is an attribute it is possible to change the attribute classifier to static or fixed by moving the cursor to the beginning of the attribute name then right clicking the highlighted attribute and selecting the static or fixed options as needed the diagram will then update to reflect the changes.



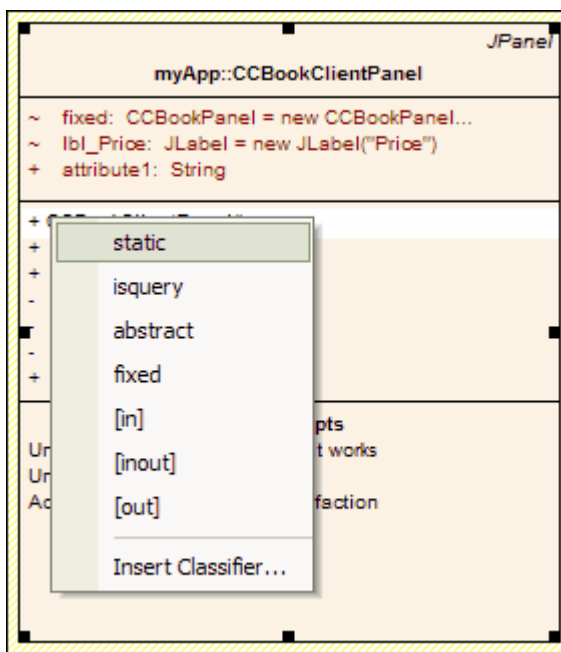
4.14.6 Edit Operation Parameter Keyword

Operation classifiers may be directly edited element by using the functions of the Inplace editing menu. This allows the user rapidly assign parameter keywords. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the operation that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



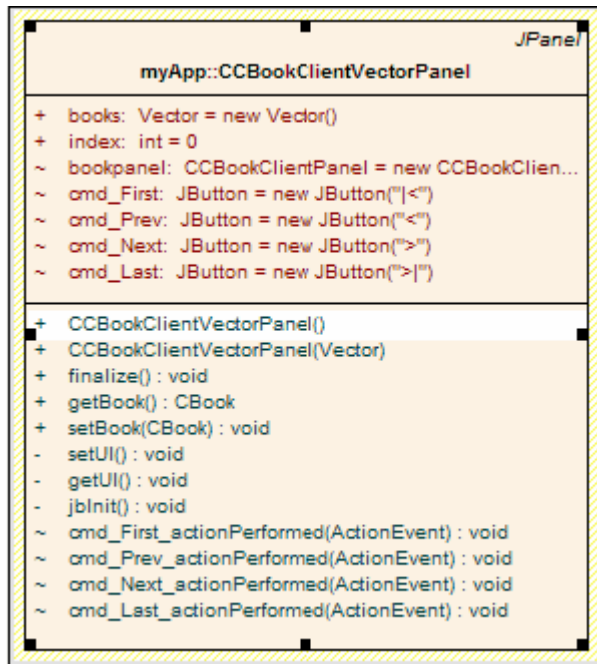
3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the operation to be edited directly from the diagram.
4. This will highlight the name of the operation.
5. If the item selected is an operation it is possible to change the operation classifier to static, isquery, abstract or fixed by moving the cursor to the beginning of the operation name then right clicking the highlighted attribute and selecting the appropriate options as needed the diagram will then update to reflect the changes.



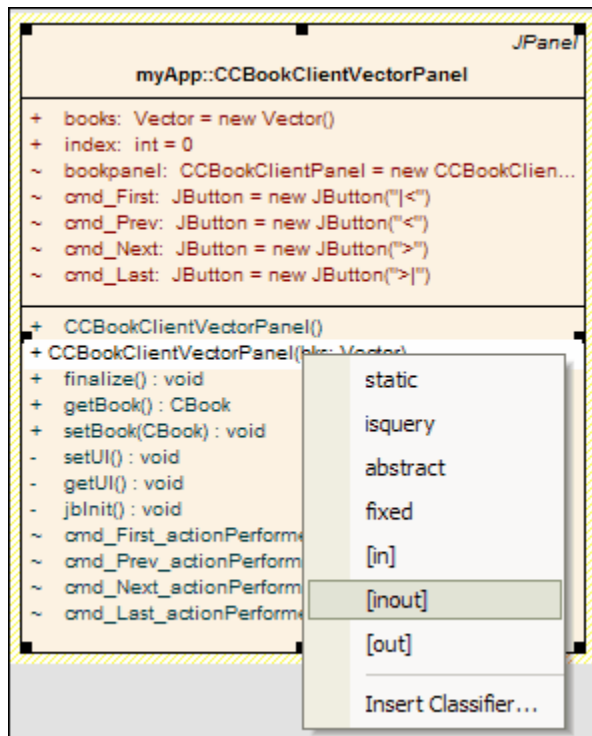
4.14.7 Edit Parameter Kind

The operation parameter kinds such as [in], [inout] and [out] directly edited an element by using the Element Keywords and Classifiers menu. This allows the user rapidly assign the parameter directly from a diagram. To use this feature use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



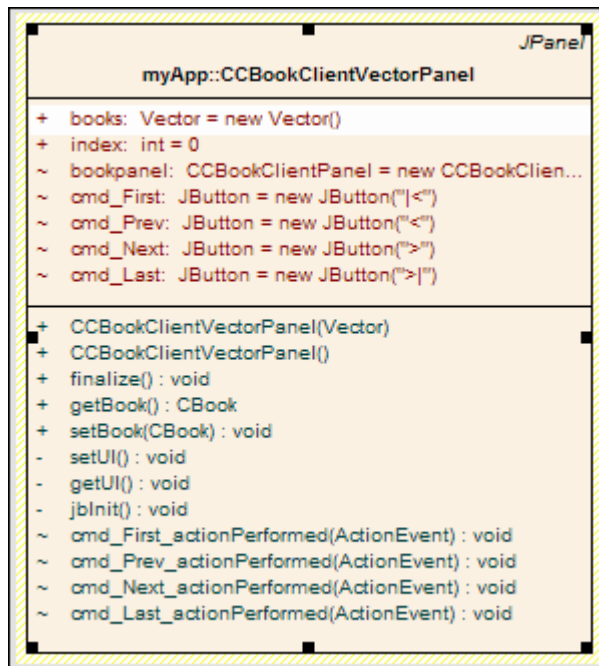
3. Right click the mouse and select *Edit Selected* or use the *F2* hotkey to allow the attribute or operation to be edited directly from the diagram.
4. This will highlight the name of the attribute or operation.
5. Operations also allow the user to set the parameter kind value to [in], [inout] and [out] by setting the cursor to the beginning of the parameter and then right clicking the highlighted operation and selecting the appropriate parameter kind value.



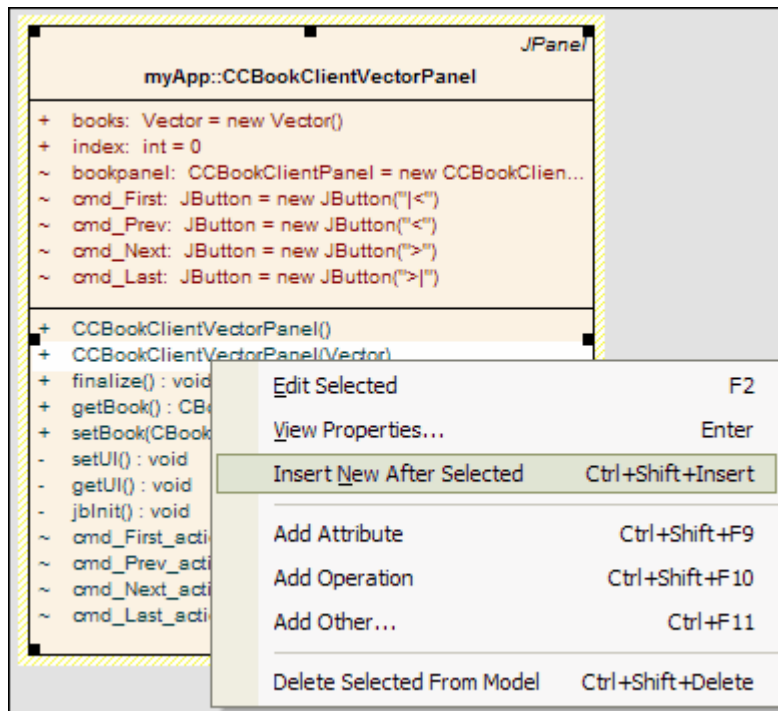
4.14.8 Insert new Attribute or Operation

Attributes and operations may be added to an element through the inline editing options by using a series of hotkey commands. To add attributes and operations to a class diagram element use the following instructions.

1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. Highlight the element that is to have either an attribute or operation added, then press the **Ctrl + Shift + Insert** hotkey combination or right click on the selected element item and choose the **Insert New After Selected** menu option.

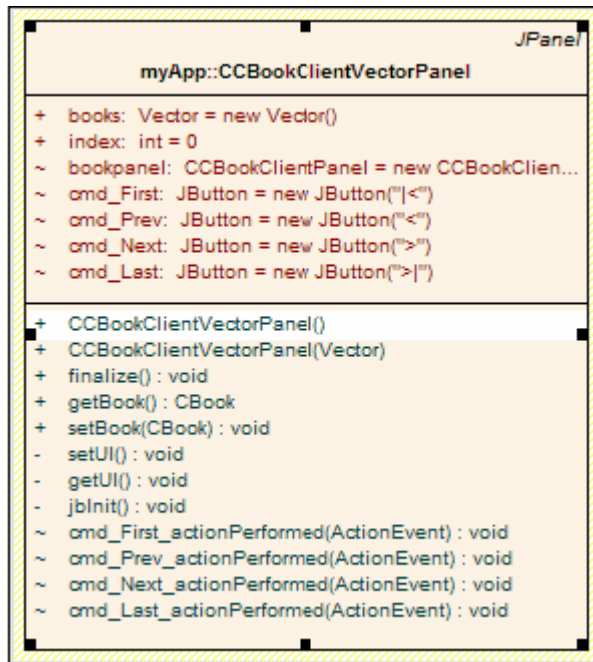


4. Type in the relevant information relating to the attribute or operation then press the **Enter** key to accept the change or press the **Esc** key to cancel the change. Once this procedure is completed the diagram will be updated to reflect the changes.

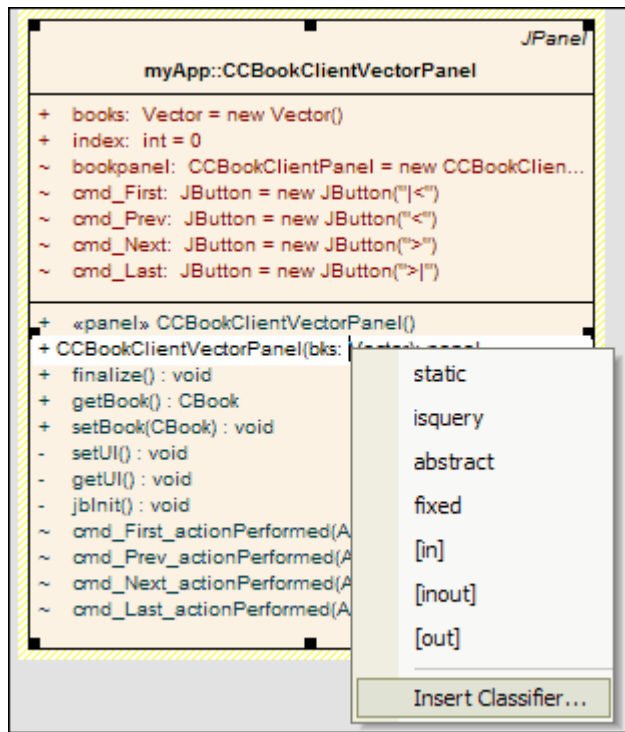
4.14.9 Insert Operation Parameter

Operations parameters may be added to an operation through the inline editing options by using a series of hotkey commands or menu shortcuts. To add parameters to operations in a class diagram element use the following instructions.

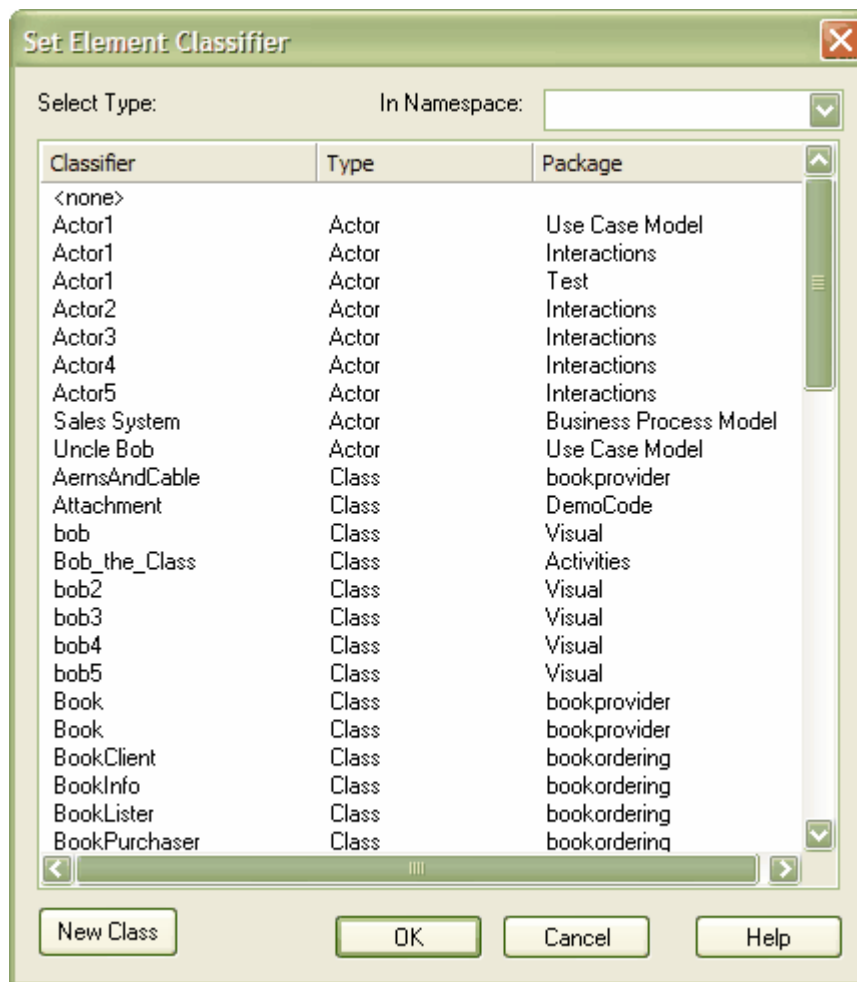
1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



3. Highlight the operation that is to have the parameter added or changed, then press **F2** or right click the selected item and select the **Edit Selected** menu item.
4. Move the cursor to the inside of the parameter brackets the choose reference to the parameter(i.e. bks: for a vector containing books) then either type the name of the parameter or place the cursor after the reference then right click the mouse to bring up the inline editing options menu and select the **Insert Classifier..** option.



5. The Set Element Classifier dialog will then appear allowing the selection of an appropriate parameter from the list of available items. Select the appropriate one then press the **OK** button to confirm your choice.

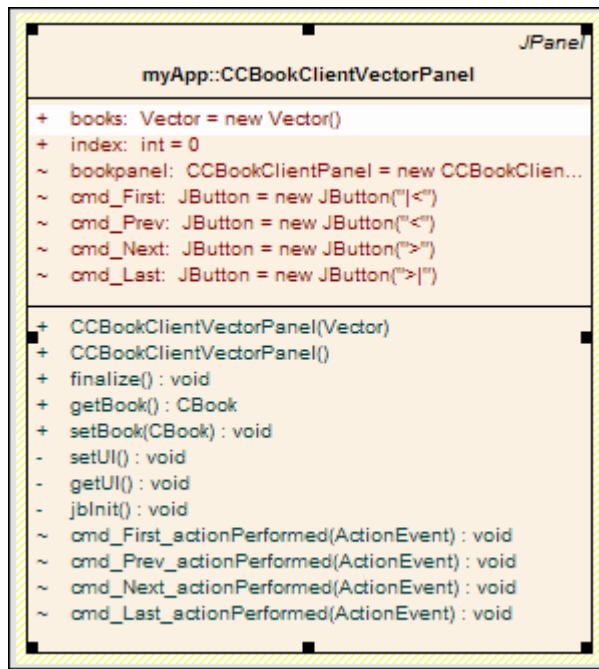


6. Press the **Enter** key to accept the change or press the **Esc** key to cancel the change. Once this procedure is completed the diagram will be updated to reflect the changes.

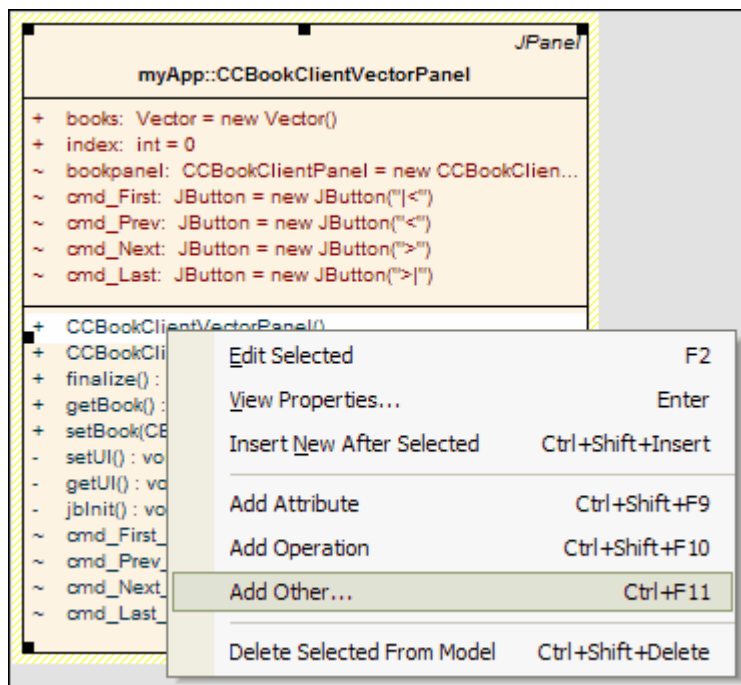
4.14.10 Insert Maintenance Feature

Maintenance items such as Defects, Changes, Issues and Tasks can be directly added to an element by using the Element Items menu. This allows the user rapidly assign maintenance details to an element item directly from a diagram. To use this feature use the following instructions.

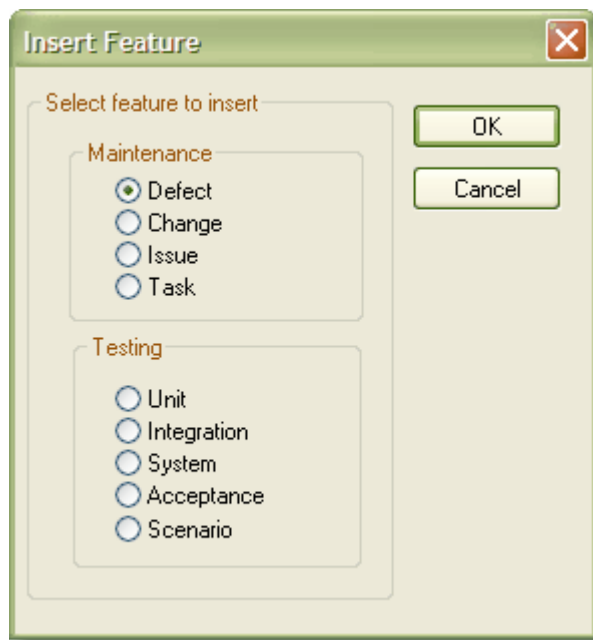
1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



- The items in the element can have a feature associated with it by using the **Ctrl + F11** hotkey combination or by right clicking the highlighted item and selecting the **Add Other** option.



- This will open up the Insert Feature dialog which allows the user to associate testing or maintenance features to the element item by selecting the appropriate radio button option and then pressing the **OK** button



5. This will open the appropriate Maintenance Dialog for the selected element, fill in the details and then save the details by pressing the **Save** button.
6. When all of the maintenance item/s for this maintenance type for the element have been completed press the **OK** button.

Defect details for Class: CCBookClientPanel!

Details

Name:

Date: Version:

Reported by: Priority:

Description:

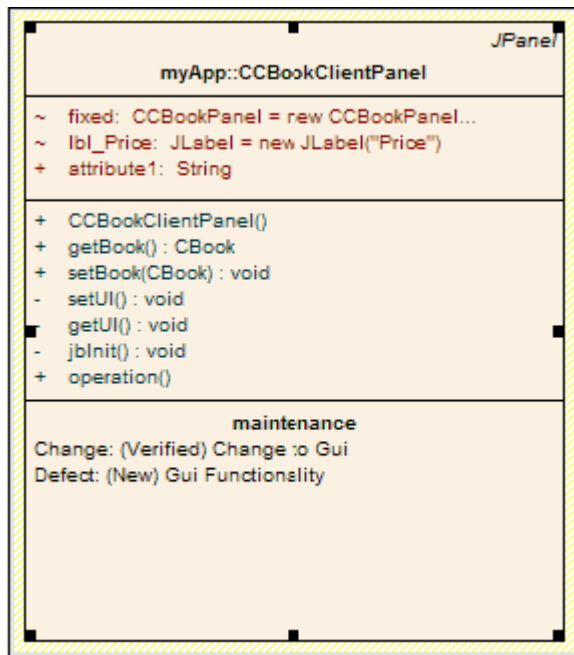
History

Status: Resolved by:

Date Resolved:

History:

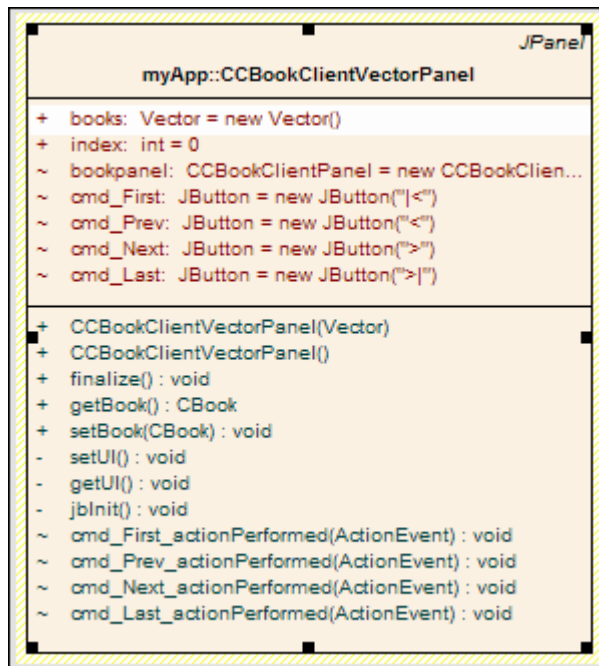
7. The maintenance details will now be added to the element. If the diagram properties appearance options have had the *Show Maintenance* option checked the maintenance item/s will be visible in the diagram element, as shown in the example below. For more information relating to diagram appearance options is found in the [Show Maintenance Scripts in Compartments](#) section.



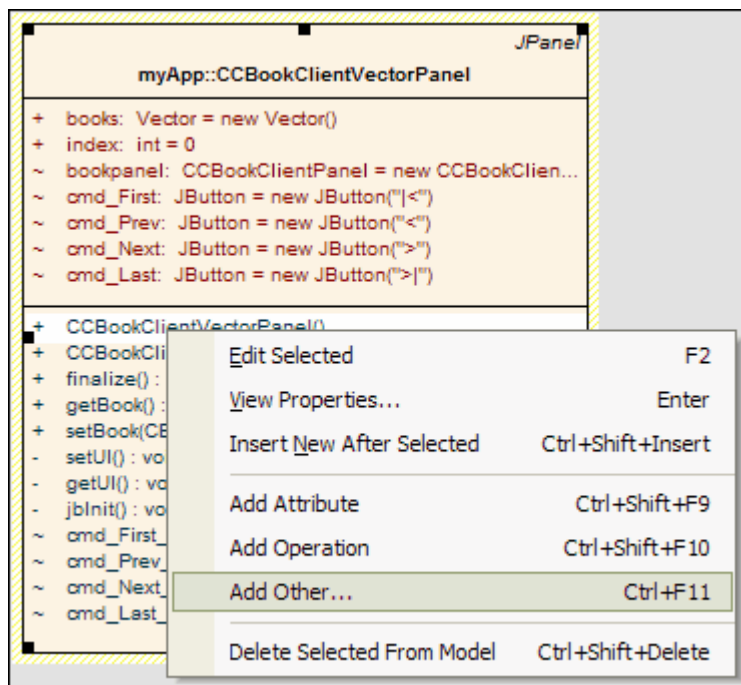
4.14.11 Insert Testing Feature

Testing features such as Unit, Integration, System, Acceptance and Scenario tests can be directly added to an element by using the Element Items menu. This allows the user rapidly assign tests to an element item directly from a diagram. To use this feature use the following instructions.

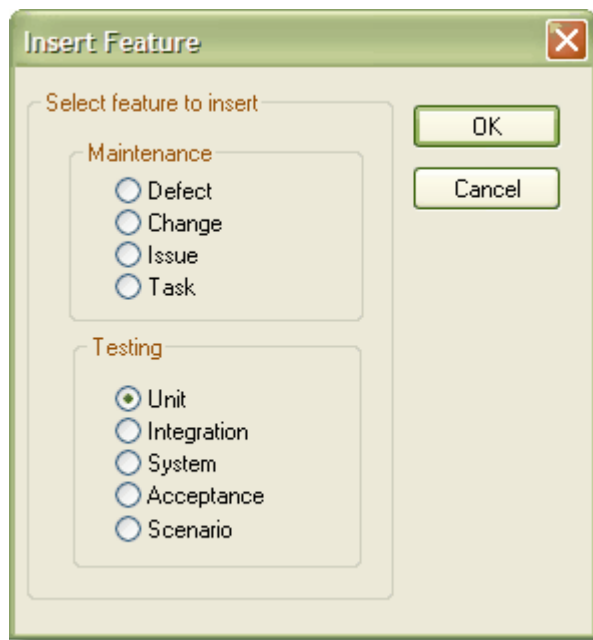
1. Open the diagram containing the element.
2. Select the element then select the item that you wish to manipulate within the element, this will indicate that it has been selected by changing the background of the selected element to a shade lighter than that of the elements background (the default is white).



- The items in the element can have a feature associated with it by using the **Ctrl + F11** hotkey combination or by right clicking the highlighted item and selecting the **Add Other** option.



- This will open up the Insert Feature dialog which allows the user to associate testing or maintenance features to the element item by selecting the appropriate radio button option and then pressing the **OK** button



5. This will open the appropriate Test Dialog for the selected element, fill in the details and then save the details by pressing the **Save** button.
6. when all of the test items for this type of test for the element have been completed press the **OK** button.

Test details for Class: CCBookClientPanel

Test Details and Execution Status

Test: Type: ▾

Description: ▾

Input: ▾

Acceptance Criteria: ▾

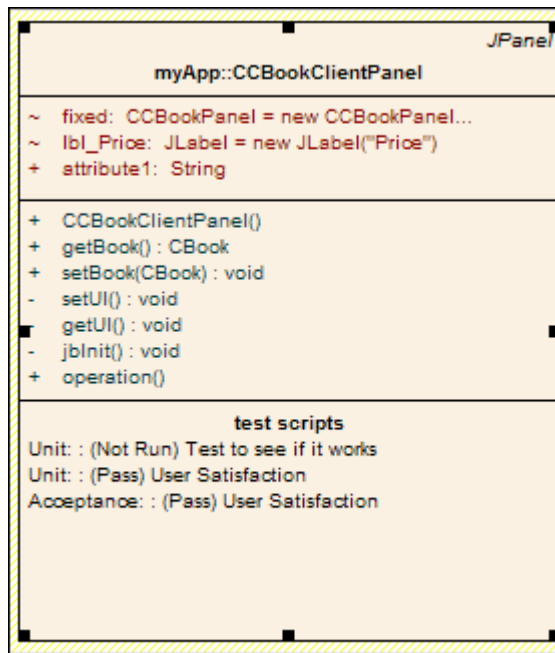
Execution

Status: ▾ Last Run Date: ▾

Run By: ▾ Checked By: ▾

Results: ▾

- The test details will now be added to the element. If the diagram properties appearance options have had the *Show Testing* option checked the test item/s will be visible in the diagram element, as shown in the example below. For more information relating to diagram appearance options is found in the [Show Test Scripts in Compartments](#) section section.



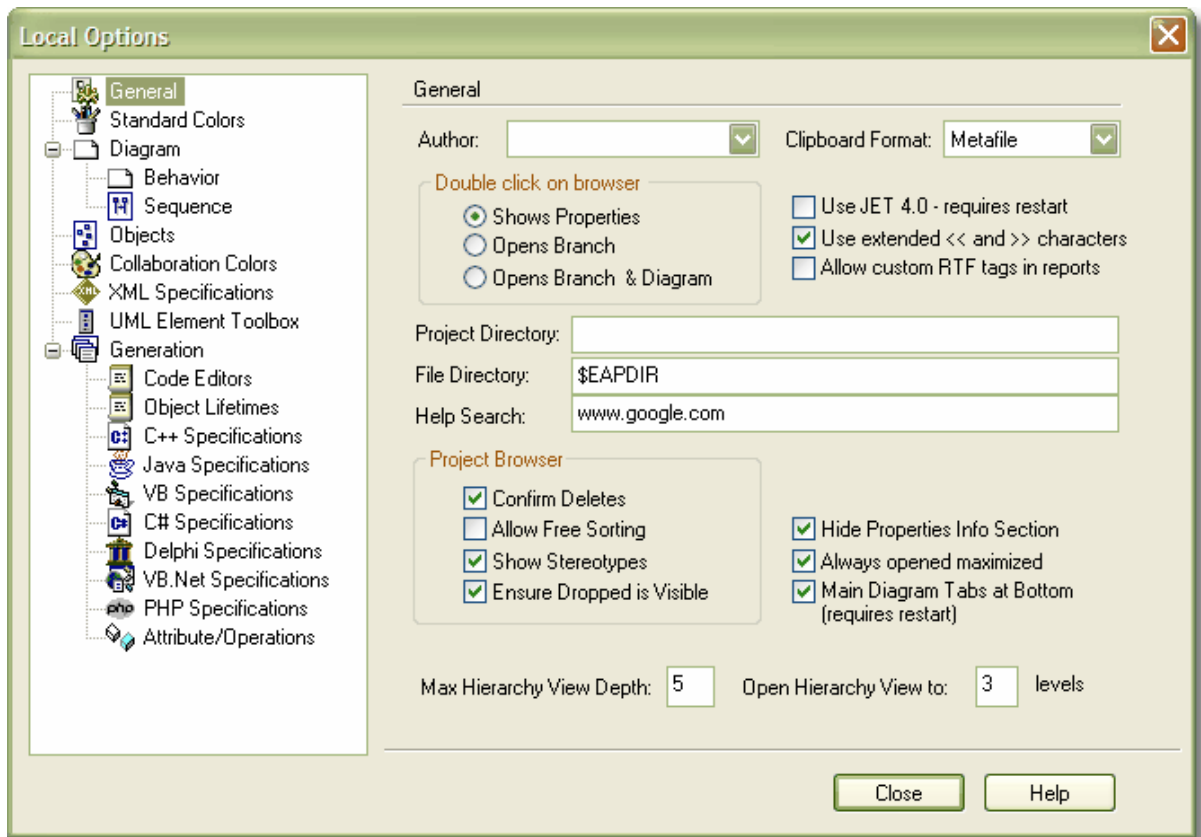
4.15 Defaults and User Settings

You can configure various settings using the [Local Options dialog](#) (open by selecting *Options* from the *Tools* menu). In addition, there are several options to change the [overall look and feel of EA](#) in the *View | Visual Style* submenu. Those settings and options are explored in the following section.

4.15.1 Configure Local Options

There are a large number of options to customise how EA displays and works with models and model elements. This section describes those settings that are local to a particular user and machine.

Open the *Local Options* dialog by selecting *Options* from the *Tools* menu.

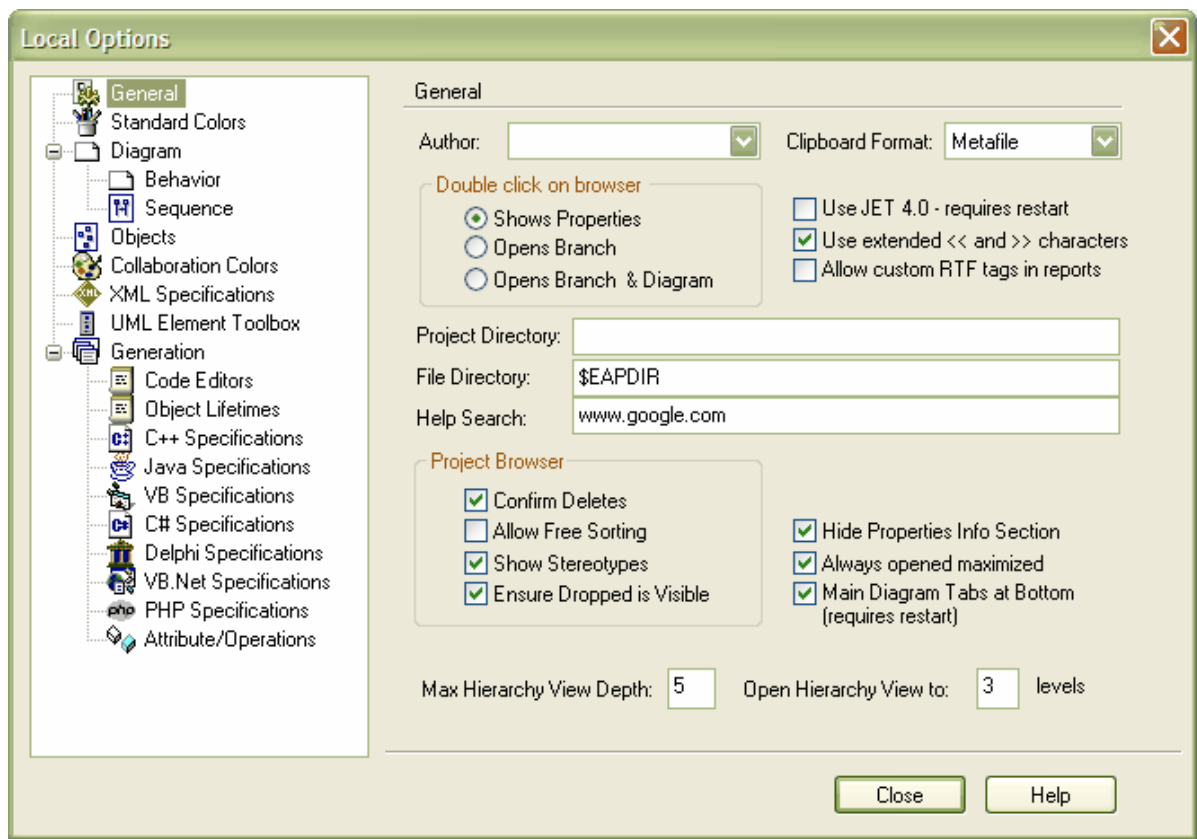


Most of these settings are stored in the user's registry so they are set for *that user only*. For a networked workplace, registry settings can be copied down to any network workstation a user logs in to. Otherwise, the settings are valid for the current machine only.

Note: Additional defaults and settings are discussed under the various code generation and import/export topics in this help file.

4.15.1.1 General

The **General** section of the **Local Options** dialog is shown below:



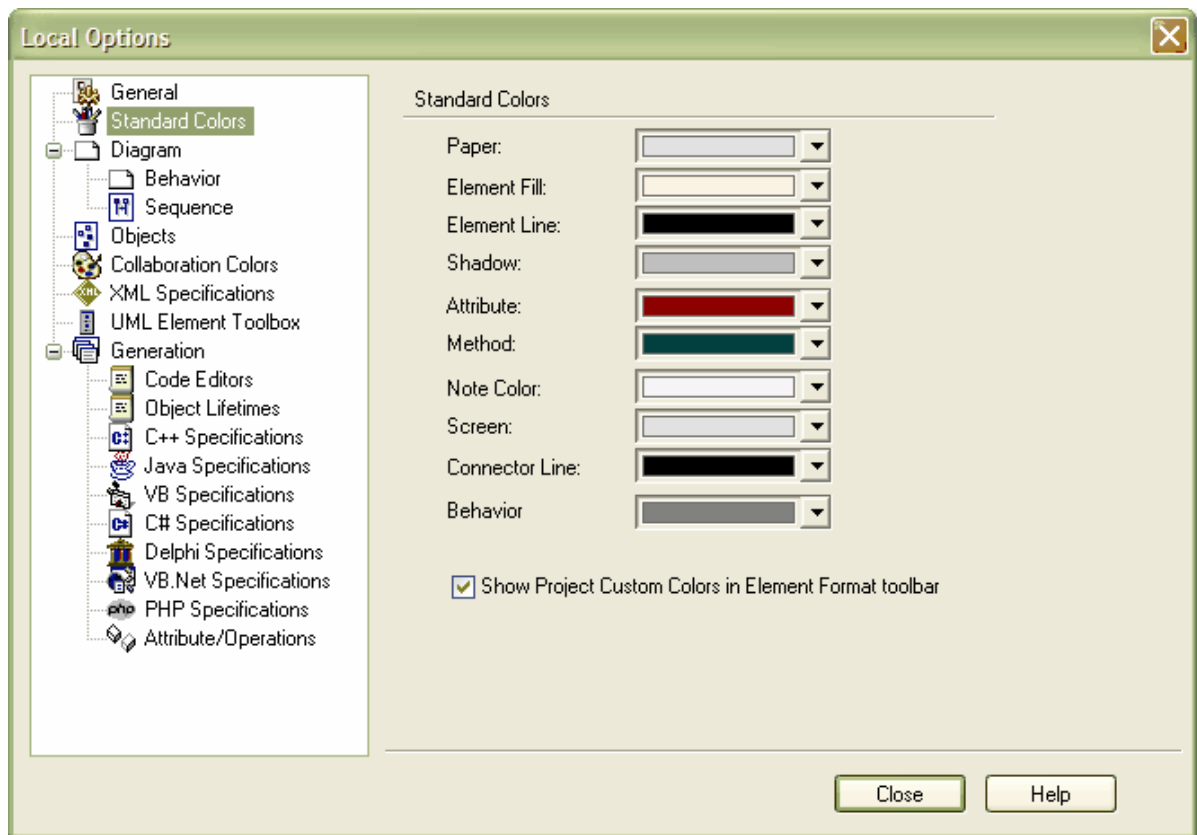
General Settings:

- *Author* - This will be used as the default author when new elements are created and modifications made.
- *Clipboard Format* - Graphic format in which to save image to clipboard - Metafile has the best detail.
- *Double Click on Browser* - Configure [Project Browser behavior](#).
- *Use JET 4.0* - Use JET 4.0 as database engine, this ensures compatibility with .EAP files compatible with versions of MS Access later than Access 97.
- *Use extended << and >> characters* - For some double byte character sets. It is best to check this box.
- *Allow custom RTF tags in reports* - Allow the use of custom RTF tags in reports.
- *Project Directory* - Default location of EA projects.
- *File Directory* - Default location for files.
- *Help Search* - Specifies the default web page to open when pressing the *open Web Search* button is pressed in the [Instant Help Window](#).
- *Confirm Deletes* (formerly "Confirm when objects are deleted from browser") - Clear to bypass confirmation dialog - only for experienced users!
- *Allow Free Sorting* - Allows "free sorting" of elements in the Project Browser regardless of type.
- *Show Stereotypes* - Shows element and feature stereotypes in the Project Browser.
- *Ensure Dropped is Visible* - If this option is checked, when moving an element in the Project Browser to another folder, the destination folder will open when you drop the element. If this option is unchecked, the destination folder will stay closed when you drop the element.
- *Hide Properties Info Section* - When checked, hides the properties information status bar on the Properties tab.
- *Always open maximized* - When checked, EA will always start up in a maximized window.
- *Main Diagram Tabs at Bottom (requires restart)* - When checked (by default) the diagram tabs will appear at the bottom of the mainview un-checking this item will result in the tabs being located at the top of the mainview.
- *Max Hierarchy View Depth* - Sets the maximum number of levels that the hierarchy will open to on the [Hierarchy tab](#).

- *Open Hierarchy View to* - Sets the initial number of levels that the hierarchy will open to on the [Hierarchy tab](#).

4.15.1.2 Standard Colors

The *Standard Colors* section of the *Local Options* dialog is shown below:



Standard Colors Settings:

- *Paper* - Paper color in diagrams.
- *Element Fill* - Fill color of elements.
- *Element Line* - Line color of elements.
- *Shadow* - Shadow color.
- *Attribute* - Attribute color.
- *Method* - Method color.
- *Note Color* - Note background color.
- *Screen* - Screen (element) color.
- *Connector Line* - Connector line color.
- *Behavior* - Color for behaviors in Activity diagrams.
- The show Project Custom Colors in Element Format toolbar - This checkbox is used allow the showing of project custom colors for more information relating to setting and getting the custom colors see the [Get and Set Project Custom Colors](#) topic.

4.15.1.3 Diagram

The *Diagram Settings* section of the *Local Options* dialog shown below allows you to configure overall options for new diagrams and general diagram behavior.

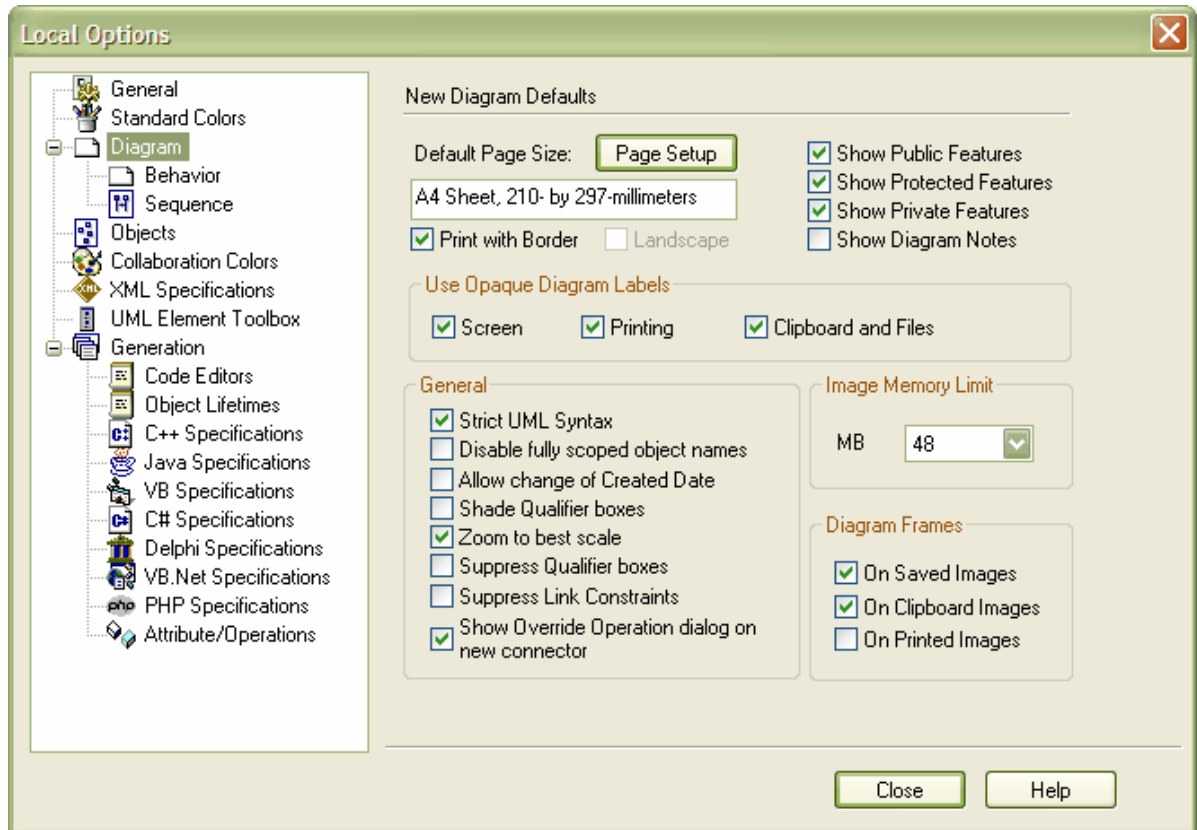


Diagram Settings:

- *Default Page Size* - Default page size for new diagrams. Change this using the *Page Setup* dialog
- *Print with Border* - Select to have pages print with a border.
- *Landscape* - If checked, pages print in landscape orientation. This checkbox is controlled from the *Page Setup* dialog.
- *Show Public/Protected/Private Features* - Set the default visibility of class features.
- *Show Diagram Notes* - Set default visibility of details on new diagrams - details include diagram name, package, version and author.
- *Opaque Diagram Labels* - Use opaque diagram labels - Screen and Printing are best, Clipboard and images may not be desirable.
- *Disable fully scoped object names* - Use this when an element is in a diagram but not in its home package. A scoped name may look like "MyClasses::foo", the "::" character indicating the class is within another namespace.
- *Allow change of Created Date* - Select to allow the creation date on the diagram properties dialog to be altered.
- *Shade Qualifier boxes* - Qualifier boxes are lightly shaded if this option is checked.
- *Zoom to best scale* - If this box is checked, the diagrams will fit neatly to screen.
- *Image Memory Limit* - Used when generating images for RTF or HTML and when saving images to file. It is important when you have very large diagrams, as it affects the point at which EA will start to scale down the image - a low memory setting means it will scale the image sooner.
- *Show Override Operation dialog on new connector* - Use this when adding generalizations and realizations between classes and interfaces. When ticked, the dialog will show automatically if the target

element has overridable features.

- *Strict UML Syntax* - Enforces when adding new connections etc.
- *Suppress Qualifier boxes* - If this option is on, qualifiers will not appear in a box, thus if it is off, qualifiers will appear in a box.
- *Suppress Link Constraints* - If this option is on, links constraints will not appear in the digram, thus if it is off, qualifiers link constraints will be shown.

4.15.1.3.1 Behavior

The *Diagram Behavior* section of the *Local Options* dialog is shown below:

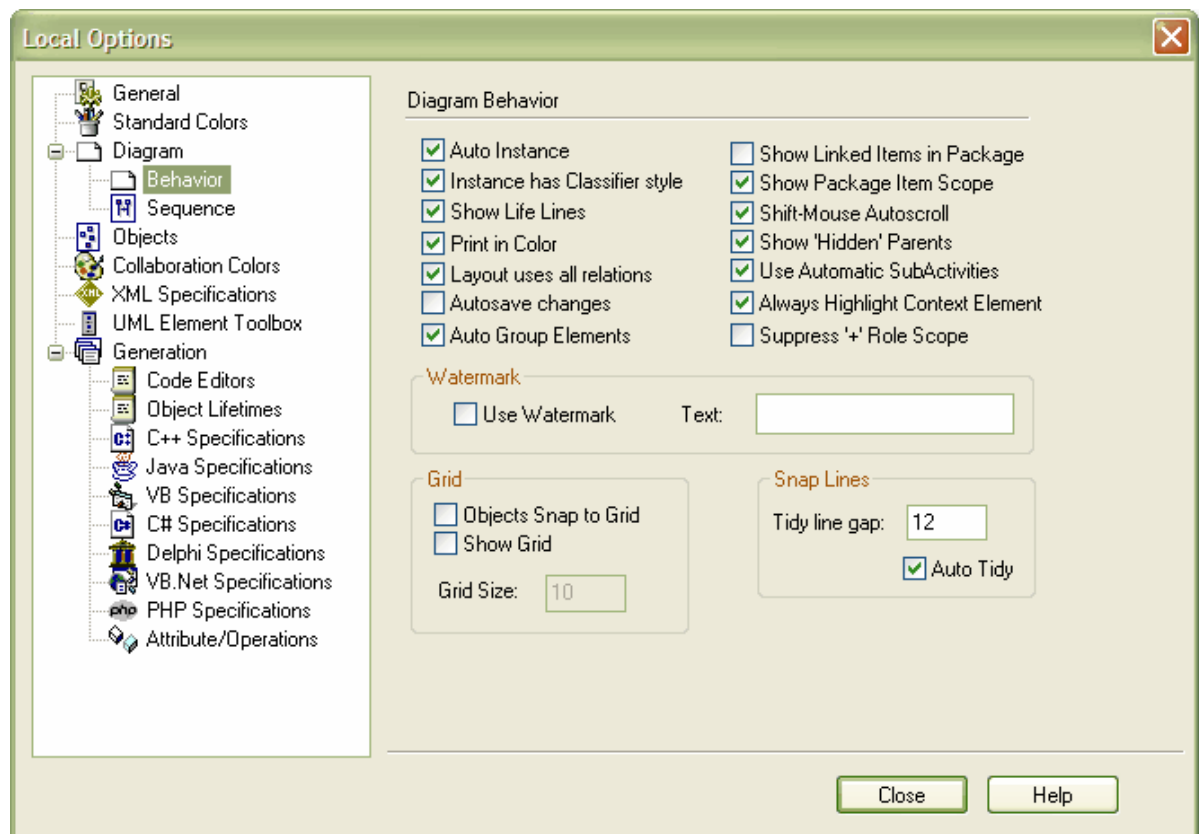
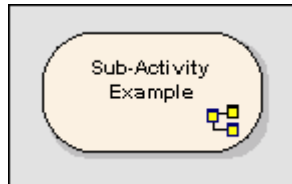


Diagram Behavior Settings:

- *Auto Instance* - When Auto Instancing is turned on, some element types - such as Class and Component - will create object instances with the dragged object as the classifier.
- *Instance has Classifier style* - When an instance is created it will have the classifier style of the element it was instantiated from.
- *Show Life Lines* - Toggle whether to show life lines for sequence elements in NON-SEQUENCE diagrams.
- *Print in Color* - Elect whether you want your diagrams printed in color or black and white.
- *Layout uses all relations* - If unchecked, only generalizations and associations will be shown.
- *Autosave changes* - If checked, EA will automatically save your changes as you work, without prompting to do so.
- *Auto Group Elements* - If checked, elements will automatically be grouped so the items can be moved using the "container" object.

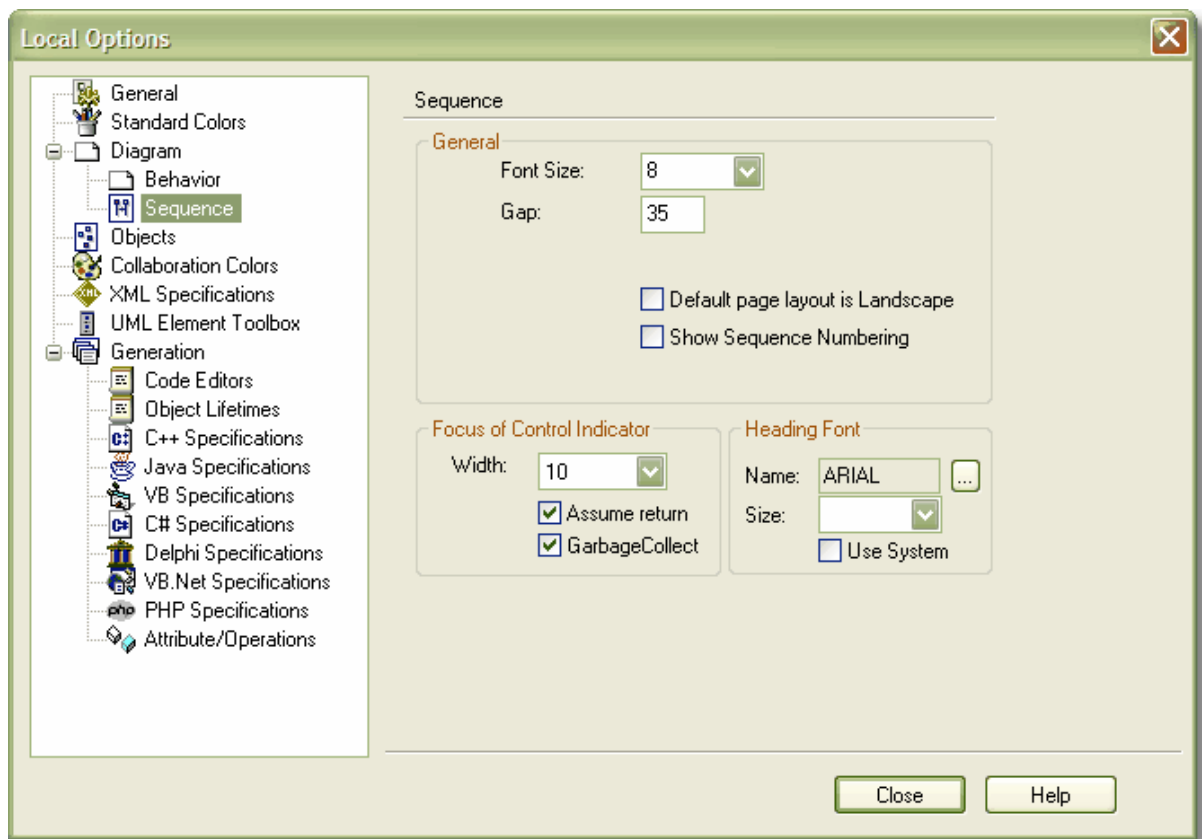
- *Show Linked Items in Package* - If checked, packages will display any linked items.
- *Show Package Item Scope* - If checked, the + and - representing the scope of the items will be display.
- *Shift-Mouse Autoscroll* - If checked, when you hold down shift you can use the mouse to autoscroll around diagrams.
- *Show 'Hidden' Parents* - If checked, any parents of elements in the diagram which aren't part of the diagram will be displayed.
- *Use Automatic Subactivities* - If checked, subactivities will be shown like so:



- *Always Highlight Context Element* - Toggle whether to show a hatch border around the selected element.
- *Use Watermark* - You can use a watermark in any diagrams you print.
- *Text* - Enter the watermark text if a watermark is to be used.
- *Objects Snap to Grid* - If checked, all elements will snap to the grid lines.
- *Show Grid* - If checked, the grid will be displayed.
- *Grid Size* - If *Objects Snap to Grid* is checked, you can set the grid size here.
- *Tidy line gap* - Set the amount EA will allow you to move a line away from horizontal and vertical when you are [tidying lines](#) for custom connectors. (See *Auto Tidy* below).
- *Auto tidy* - Automatically [tidy line angles](#) for custom connectors. This 'nudges' the custom line into horizontal and vertical increments.
- *Suppress ' + ' Role Scope* - ensure that the role and scope are not displayed on the diagram.

4.15.1.3.2 Diagrams

The *Sequence* section of the *Local Options* dialog shown below allows you to configure various font settings and focus of the control indicator.

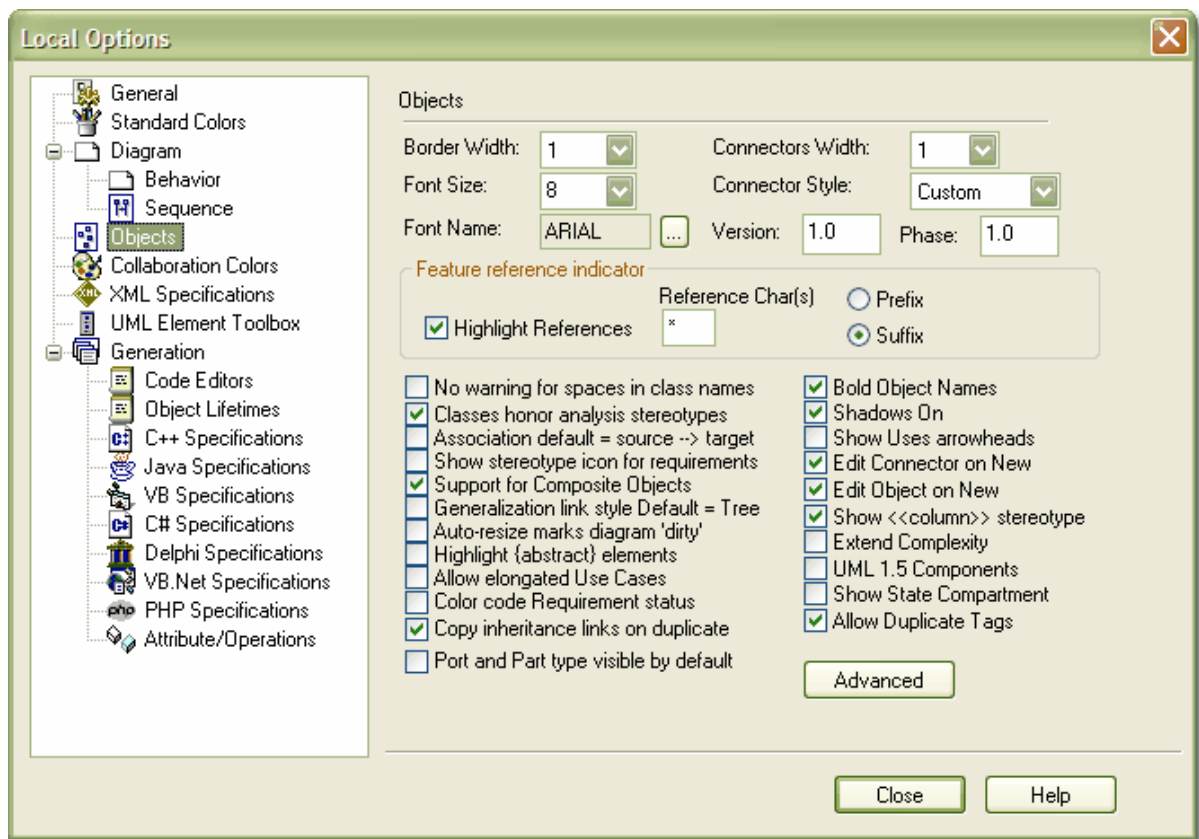


Sequence Settings:

- *Font Size* - For messages in sequence diagrams.
- *Gap* - Set the vertical gap between sequence messages (may be overridden manually by dragging a message up or down).
- *Default page layout is Landscape* - Set the default orientation of Sequence diagrams to landscape.
- *Show Sequence Numbering* - Select whether to show sequence numbers on sequence messages
- *Width* - The width of the 'focus of control' rectangle (thick part of lifeline).
- *Assume return* - Assume implicit returns when none are explicitly drawn (recommended).
- *Garbage Collect* - Automatically truncate lifelines for created elements after the last message (ie. assume garbage collect rather than explicit delete).
- *Heading Font Name/Size/Use System* - Set the heading font name and size (above diagram in caption bar). Particularly useful for non-English character sets.

4.15.1.4 Objects

The *Objects* section of the *Local Options* dialog shown below lets you configure how elements look and respond in diagrams, as well as some connector defaults.



Objects Settings:

- *Border Width* - Default border width (in pixels).
- *Font Size* - Default font size on diagrams.
- *Font Name* - Font name and style to use for elements.
- *Connector Width* - Default connector width.
- *Connector Style* - Default connector link style for new connectors.
- *Version* - Default version for new elements.
- *Phase* - Default phase for new elements.
- *Highlight References* - Highlight parameters in operations that are passed by Reference rather than value.
- *Reference Char(s)* - Specify a character to use for the above.
- *Prefix/Suffix* - Indicate whether to use Reference Char(s) as a prefix (before) or a suffix (after).
- *No warning for spaces in class names* - If unchecked, a warning message will appear if an element name has embedded spaces.
- *Classes honor analysis stereotypes* - If checked, classes will be shown as their stereotype - eg. if a class is stereotyped as a boundary, it will appear as a boundary rather than a class.
- *Association default = source -> target* - Set the navigability of new associations to Source->Target (ie. with an arrow head at target).
- *Show stereotype icon for requirements* - Show/hide a small icon in Requirement, Change and Issue elements
- *Support for Composite Objects* - Support embedded or composite elements through automatic aggregation.
- *Generalization link style default = tree* - If checked, generalizations will be shown as [tree style hierarchies](#).
- *Auto-resize marks diagram 'dirty'* - Make auto-re sizing of elements (eg. classes) mark the current diagram as dirty.
- *Highlight {abstract} elements* - Highlight abstract elements with a suitable tag "{abstract}" in the top right

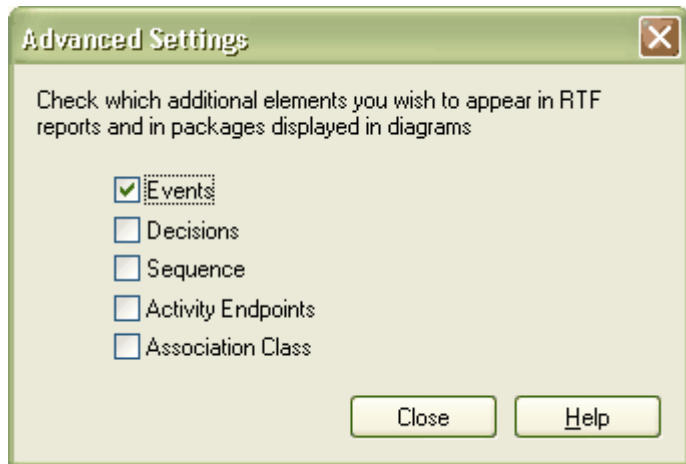
of the class.

- *Allow elongated Use Cases* - If checked, use cases or use case extension points with long names can be stretched to a disproportionate width to allow space for the name. If unchecked, use case re sizing is proportional.
- *Color code Requirement status* - If checked this option enables color coding for requirements.
- *Copy inheritance links on duplicate* - Checked by default, this option duplicates inheritance and realization links when an edit/copy is performed (**Ctrl + Shift + V**).
- *Bold Object Names* - Elements in diagrams will be shown bold face.
- *Shadows On* - Toggle element shadows.
- *Show Uses arrowheads* - Show an arrowhead on the Actor->Use Case associations.
- *Edit Connector on New* - Automatically show the connector properties dialog when a new connector added.
- *Edit Object on New* - Automatically show the element properties dialog when a new element added.
- *Show <<column>> stereotype* - Hide or show the <<column>> stereotype used when data modeling.
- *Extend Complexity* - If checked, five levels of complexity will be available in the Complexity option on the Properties tab. Otherwise only three levels are available.
- *UML 1.5 Components* - Use UML 1.5 components (Enterprise Architect version 4 and over supports UML 2.0).
- *Show State Compartment* - This shows or hides the visibility of the State Compartment divider under the state name.
- *Allow Duplicate Tags* - This allows for tags to be duplicated.
- *Port and Part type visible by default* - This allows for the ability to show Port and Part types by default.
- [Advanced](#) button - Use this to set visibility of certain elements in reports and in diagram packages.

4.15.1.4.1 Element Visibility

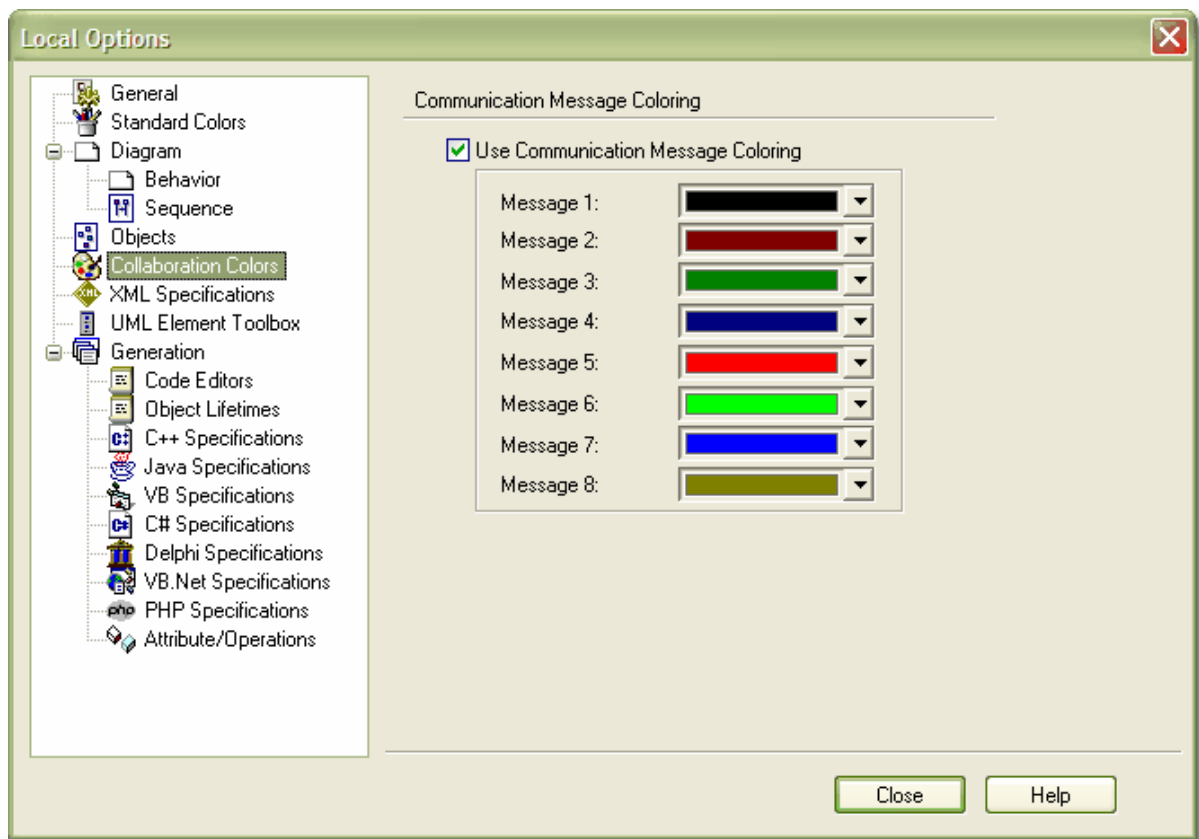
Some elements do not appear in packages and in RTF output by default. Select *Advanced* from the *Objects* section of the *Local Options* dialog to specify which elements should be visible.

See the section on [customizing element visibility](#) for more details.



4.15.1.5 Communication Colors

The *Communication Colors* settings in the *Local Options* dialog allow you to configure the colours used in collaboration diagrams. When you enable this option, collaboration messages will appear in different colors depending on the sequence group they belong to on a diagram - eg. 1.n are black, 2.n are red, 3.n are green etc.

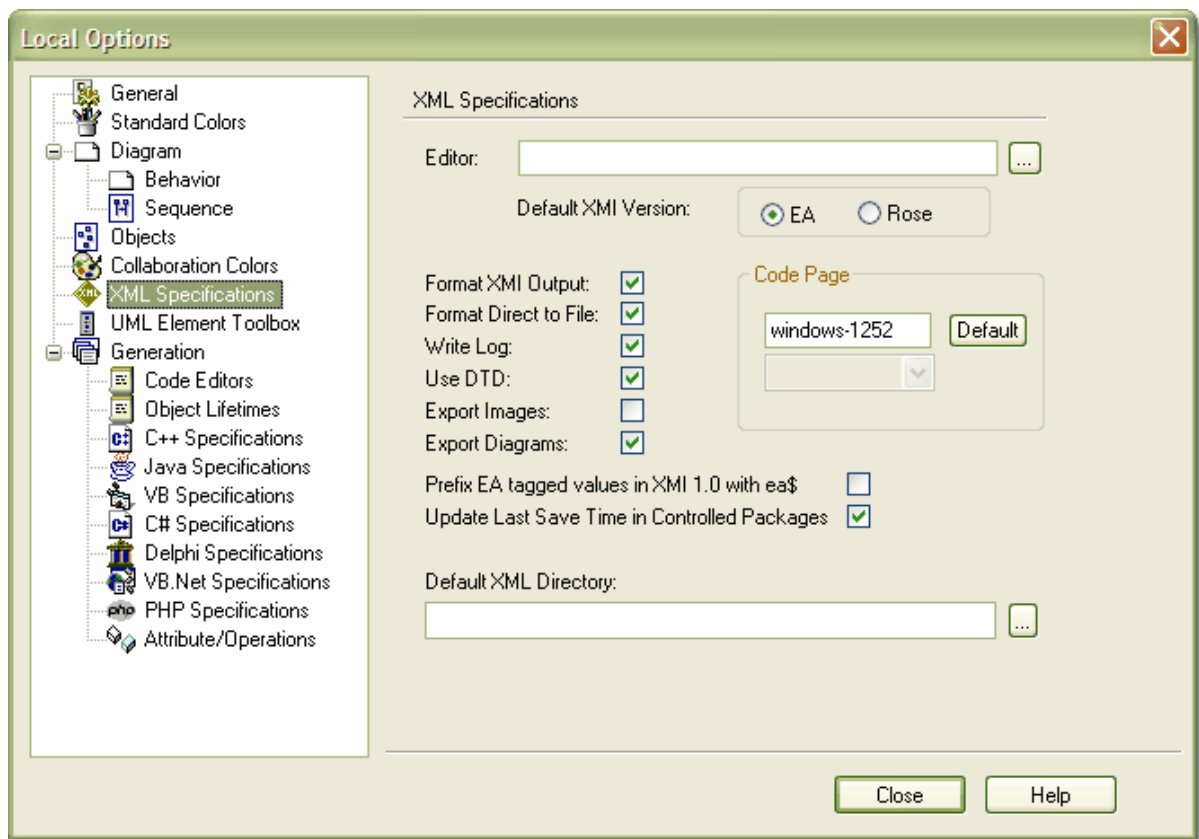


Check the *Use Communication Message Coloring* to turn on coloured messages.

Set the color sequence to what you require - the pattern will repeat after 8 sequence groups.

4.15.1.6 XML Specifications

The *XML Specifications* section of the *Local Options* dialog is shown below. This allows you to configure various settings for working with XML.

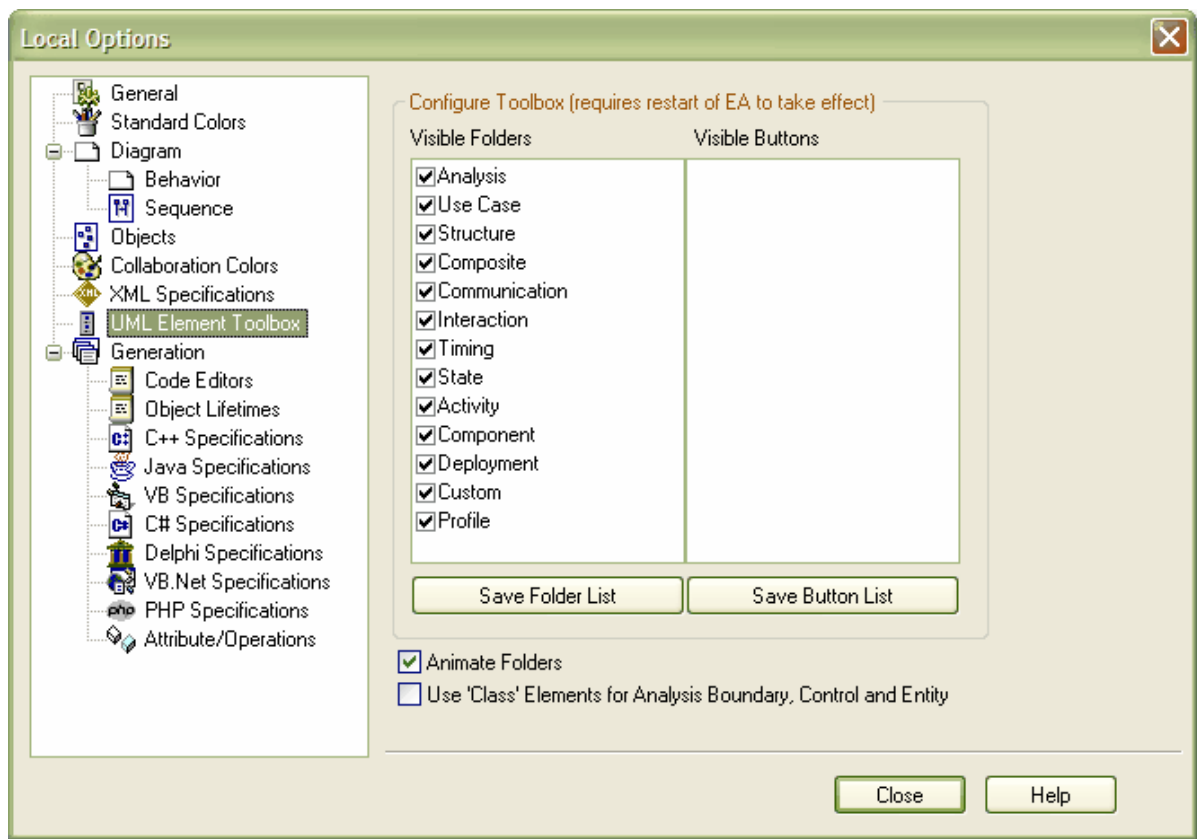


XML Specifications Settings:

- *Editor* - Set the default editor for XML documents you open within EA.
- *Default XMI Version* - XMI version to use - EA or Rose.
- *Format XML Output* - This option allows the user to determine whether formatting will be applied to your XML output.
- *Format Direct to file* - This option speeds up the performance of writing XML files by writing directly to the file a line at a time, this option should not be selected when performing writes to XML files to locations with slow remote storage.
- *Write Log* - Set whether to write to a log file when you import or export XML.
- *Use DTD* - Set whether to use DTD.
- *Export Images* - Set whether to export images when you export XML.
- *Code Page* - Sets the Code Page to use, setting a NULL encoding string will result in the encoding tag being entirely omitted from the XML output.
- *Export Diagrams* - Set whether to export diagrams when you export XML.
- *Prefix EA tagged values in XMI 1.0 with ea\$* - Set whether to prefix any EA tagged values within any XMI 1.0 you create with ea\$.
- *Update Last Save Time in Controlled Packages* - Set whether to update the time you last saved in controlled packages.
- *Default XML Directory* - Default XML directory to use when importing and exporting XML.

4.15.1.7 UML Element Toolbox

The *UML Element Toolbox* contains the model elements you will use in diagrams. You can configure the appearance and behavior of the toolbar.



Outlook Toolbar Settings:

- *Animate Folders* - When checked, the toolbar will be animated whenever a new category is selected.
- *Use 'Class' Elements for Analysis Boundary, Control and Entity* - Ensures that a class element is used for analysis boundaries, control and entities.

Assigning Visible Folders:

To assign the visibility of folders use the following procedure:

1. Highlight the folder of interest, and uncheck the check box to hide all of the entire button group under this category.
2. Press the *Save Folder List* button
3. Restart Application to have the new settings take effect.

Assigning Visible Buttons:

To assign the visibility of folders use the following procedure:

1. Highlight the folder of interest, and in the Visible buttons section uncheck the check boxes that apply to the buttons that you wish to hide from use.
2. Press the *Save Button List* button.
3. Restart the application to have the new settings take effect.

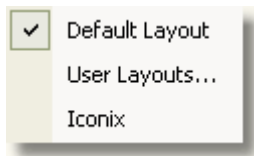
4.15.1.8 Generation

Information about the Code Generation settings can be found in the [Code Engineering Settings](#) section.

4.15.2 Custom Layouts

EA supports some customization of the default desktop layout and toolbox folder sequence. This is useful for resetting your current layout to the standard layout, and for using custom layouts for working with specific processes.

This feature is accessible from the *View* | *Visual Layouts* submenu.

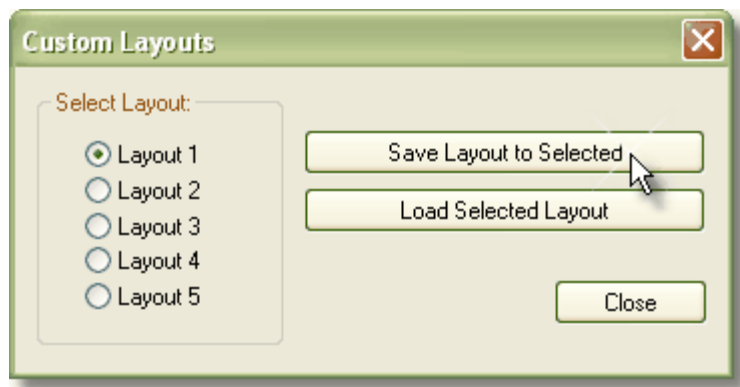


When you select an option EA will realign toolbars and docked windows to the defaults for that option. You can read more about custom layouts and the processes supported at:

<http://www.sparxsystems.com.au/customlayouts.htm>.

Setting User Layouts

If you have the layout of the EA windows and tools just as you want them, you can preserve your layout by saving it as a custom user layout. Go to *View* | *Visual Layouts* | *User Layouts* - this will open the *Custom Layouts* dialog as shown below:



You can save up to 5 custom layouts. To save the current layout to Layout 1 ensure the *Layout 1* option is selected, then press *Save Layout to Selected*.

If you want to make changes to an existing layout, you can update the currently saved layout by opening the *Custom Layouts* window, selecting the radio button for the layout you want to update, then pressing *Save Layout to Selected*.

Load a saved layout by going to the *Custom Layouts* window, selecting the layout you wish to load, and pressing *Load Selected Layout*.

Warning About Custom Layouts and Keyboard Shortcuts

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

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

For more information about setting keyboard shortcuts, see the [Customize Keyboard](#) topic.

4.15.3 Visual Styles

You can configure the overall look and feel of EA to suit your working environment. Options range from a classic windows application to an enhanced XP look.

To change the appearance of EA

1. Open the **View | Visual** Style submenu
2. Select the style you want from the list:

XP	Similar to applications like MS Office and MS Visual Studio
2003	Colourful modern style
2005	Rounded modern style.
Classic	Classic windows application look.

You can also [enable or disable menu shadows](#) and animation of [auto-hidden](#) windows.

Note: The 2005 style uses the [navigation compass](#) method for docking windows.

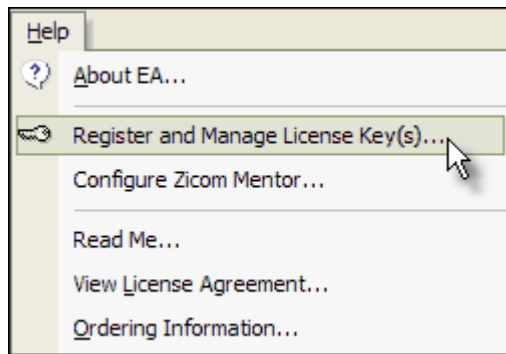
4.16 Register Add-In

You can register Add-ins for EA by using the following instructions:

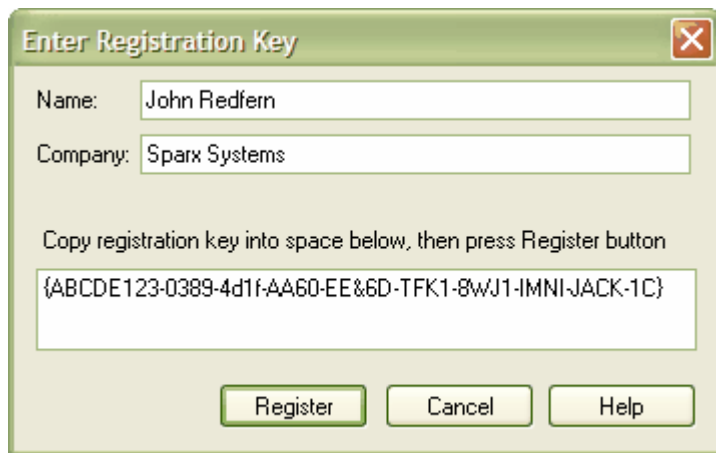
Note: Zicom Mentor requires a different method of registration. To register your version of Zicom Mentor go to the [Registering Zicom Mentor](#) page.

Registering an Add-in for EA

1. Purchase one or more licenses of the Add-in from your Add-in provider.
2. Once you have paid for a licensed version of the Add-in, you will receive (via email or other suitable means) a licence key for the product.
3. Save the license key and the latest full version of the Add-in.
4. Run the Add-in's setup program to install the Add-in.
3. From the **Help** menu, select the **Register and Manage Licence Key(s)** option.



4. This will bring up the Licence Management Dialog, press the *Add Key button*.
5. The user will be presented with the *Enter Registration Key* dialog (enter the key that you received with the Add-in - **including the { and } characters**).



6. When the Add-in has been added successfully, close down EA and restart to apply the new changes.

4.16.1 Register Zicom Mentor

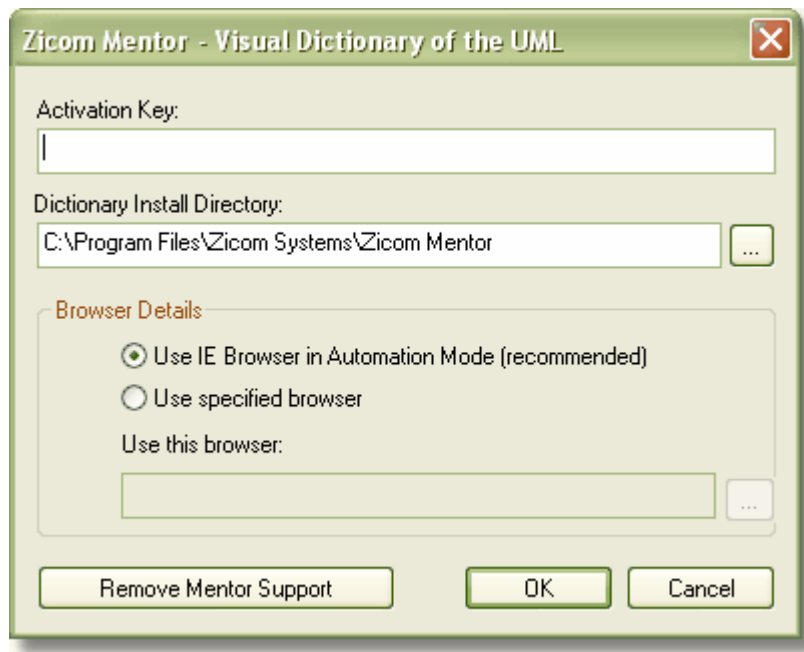
[Zicom Mentor](#) is a detailed Visual Dictionary for UML by **Zicom Systems**. It is HTML based and integrates into Enterprise Architect, such that you can get context sensitive help on any UML Element directly from within EA. To use UML Mentor you will need to first purchase a license from Sparx Systems, then install UML Mentor and register the product in EA.

Integrating UML Mentor with EA

Integration can only be carried out if both Enterprise Architect and the Zicom Mentor Visual UML Dictionary have been installed.

1. Start Enterprise Architect.
2. From the *Help* menu, under select the *Configure Zicom Mentor* option.
3. Enter your Zicom Mentor serial number (use "evaluation" for the trial version).
4. Enter (or browse [...]) to the location where you installed Zicom Mentor.
5. Enter your Browser location.

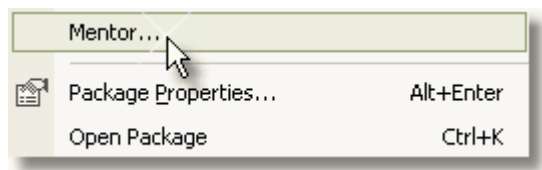
6. Click **OK**.



Accessing Zicom Mentor from Enterprise Architect

The Visual UML Dictionary can be accessed directly in EA from the diagram context menu (right-click menu) or selected Project Browser item context menus.

1. Right click on a UML element or diagram.
2. From the context menu, select the **Mentor** option.



Removing Zicom Mentor Integration in Enterprise Architect

Zicom Mentor integration can be removed from Enterprise Architect at any time. Removal of the integration does not uninstall Zicom Mentor.

1. Start Enterprise Architect.
2. From the **Help** menu, select the **Configure Zicom Mentor** option.
3. Click **Remove Mentor Support**.

4.17 Keyboard Shortcuts

The table below lists the default keyboard shortcuts functions within EA:

Function	Shortcut	Category
Create a new Enterprise Architect project	Ctrl + N	File
Open an Enterprise Architect project	Ctrl+ O	File
Print the active diagram	Ctrl + P	File
Reload the current project	Ctrl + Shift + F11	File
Add a single element to the clipboard list	Ctrl + Space	Edit
Paste element as link	Shift + Insert	Edit
Paste element as new	Ctrl + Shift + V	Edit
Bookmark current element with red marker	Alt + Space	Edit
Delete selected element(s)	Ctrl + D	Edit
Search for elements in the project	Ctrl + F	Edit
Paste element(s) from the clipboard as a link	Shift + Insert	Edit
Paste element as new element in model	Ctrl + Shift + V	Edit
Autohide the current window	Ctrl + Shift + F4	View
Close the current window	Ctrl + F4	View
View element hierarchy docked window	Ctrl + Shift + 4	View
Show or hide the maintenance window	Alt + 4	View
Show or hide the notes window	Ctrl + Shift + 1	View
Show or hide project workspace window	Alt + 0	View
View the properties window	Alt + 1	View
View element relationships docked window	Ctrl + Shift + 2	View
Show resource view	Alt + 6	View
View requirements and constraints docked window	Ctrl + Shift + 3	View
View element scenarios docked window	Ctrl + Shift + 5	View
View and edit source code	Alt + 7	View
Show/hide the system docked window	Alt + 2	View
Show or hide the tagged values docked window	Ctrl + Shift + 6	View
View or hide the dockable test window	Alt + 3	View
Display the UML toolbox	Alt + 5	View
Batch generate selected classes	Shift + F11	Project

Function	Shortcut	Category
Import package from XMI	Ctrl + Alt + I	Project
Import package from directory	Ctrl + Shift + U	Project
Manage locks applied by current user	Ctrl + Shift + L	Project
Configure package control	Ctrl + Alt + P	Project
Rich text documentation	F8	Project
Synchronize current element	Ctrl + R or F7	Project
Synchronize package contents	Ctrl + Alt + M	Project
Diagram properties	F5	Diagram
Repeat last connector	F3	Diagram
Save	Ctrl + S	Diagram
Save image to clipboard	Ctrl + B	Diagram
Save image to file	Ctrl + T	Diagram
Set visible relations	Ctrl + Shift + I	Diagram
Space elements evenly horizontally	Alt + -	Element
Add attribute	Ctrl + Shift + F9	Element
Add operation	Ctrl + Shift + F10	Element
Add other	Ctrl + F11	Element
Attributes	F9	Element
Auto size selected elements	Alt + Z	Element
Align bottom edges of selected elements	Ctrl + Alt + Down	Element
Configure element appearance	Ctrl + Shift + E	Element
Delete selected feature from model	Ctrl + Shift + Delete	Element
Edit selected	F2	Element
Align selected elements on left boundaries	Ctrl + Alt + Left	Element
Align selected elements on right boundaries	Ctrl + Alt + Right	Element
Space selected elements evenly	Alt + =	Element
Manage embedded elements	Ctrl + Shift + B	Element
Insert new feature after current selection	Ctrl + Shift + Insert	Element
Locate in browser	Alt + G	Element
New element	Ctrl + M	Element
View source code in default editor	Ctrl + E or F12	Element

Function	Shortcut	Category
Operation	F10	Element
Override inherited features	Ctrl + Shift + O	Element
Configure element properties	Alt + Enter	Element
Project resourcing, metrics and risk	Ctrl + Shift + K	Element
Select alternate image	Ctrl + Shift + W	Element
Specify feature visibility	Ctrl + Shift + Y	Element
Set element parent or implement interface(s)	Ctrl + I	Element
Set references to other elements and diagrams	Ctrl + J	Element
View element usage	Ctrl + U	Element
Add tagged value	Ctrl + Shift + T	Element
Align top edges of selected elements	Ctrl + Alt + Up	Element
View properties dialog	Enter	Element
Check data integrity	Shift + F9	Tools
Configure system options	Ctrl + F9	Tools
Spell check current package	Ctrl + Shift + F7	Tools
Spell check model	Ctrl + F7	Tools
Edit code generation templates	Ctrl + Shift + P	Configuration

Part



5 The UML Language

Enterprise Architect's modeling platform is based on the Unified Modeling Language (UML). This language is designed to be flexible, extendable and comprehensive, but generic enough to serve as a foundation for all system modeling needs. With its specification, there is a wide range of elements characterized by the diagrams they serve, and the attributes they provide. All can be further specified by using stereotypes, tags, and profiles. EA also provides additional custom diagrams and elements, to address further modeling interests. This section is intended to provide an introduction to EA's diagrams, elements, connectors and its modeling process, along with illustrating its alignment, when applicable, to the Unified Modeling Language.

This section is divided into three parts:

- [UML Diagrams](#)
- [UML Elements](#), which is itself divided into:
 - [Behavioral Diagram Elements](#)
 - [Structural Diagram Elements](#)
- [UML Connections](#)

5.1 What is UML

The Unified Modeling Language

Enterprise Architect's modeling platform is based on the Unified Modeling Language (UML), a standard that defines rules and notations for specifying business and software systems. The notation supplies a rich set of graphic elements for modeling object oriented systems, and the rules state how those elements may be connected and used. UML is not a tool for creating software systems. Instead, it is a visual language for communicating, modeling, specifying and defining systems.

This language is designed to be flexible, extendable and comprehensive, yet generic enough to serve as a foundation for all system modeling needs. With its specification, there is a wide range of elements characterized by the kinds of diagrams they serve, and the attributes they provide. All can be further specified by using stereotypes, tags, and profiles.

Wide range of Application

Although initially conceived as a language for software development, UML may be used to model a wide range or real world domains. For example, UML can be used to model many real world Processes (in business, science, industry, education and elsewhere), Organizational Hierarchies, Deployment maps and much more.

EA also provides additional custom diagrams and elements, to address further modeling interests. This section is intended to provide an introduction to EA's diagrams, elements, connectors and its modeling process, along with illustrating its alignment, when applicable, to the Unified Modeling Language.

Extending UML for new Domains

Using UML Profiles, UML Patterns, Grammars, Data types, Constraints and other extensions, UML and EA may be tailored to address a particular modeling Domain not explicitly covered in the original UML specification. EA makes extending the UML simple and straightforward, and best of all, the extension mechanism is still part of the UML Specification.

Find out more

UML is a standard specified by the Object Management Group. Further information, including the full UML 2.0 documentation can be found on their website at <http://www.omg.org>

See also:

- [UML Diagrams](#)
- [UML Elements](#)
- [UML Connections](#)

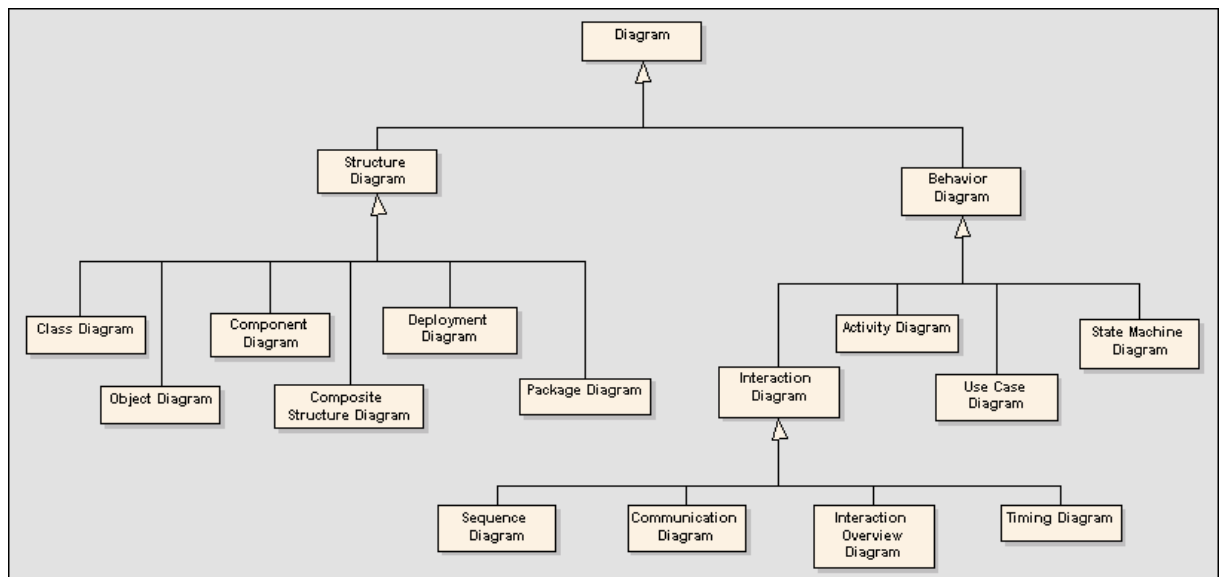
- [Modeling with UML](#)

5.2 UML Diagrams

Types of Diagram

The figure below shows the taxonomy of the 13 UML diagrams, as defined by the Object Development Group's UML 2.0 specification. As can be seen, there are two major groupings of diagrams: [Structural diagrams](#) which show a static view of the model; and [Behavioral diagrams](#) which show a dynamic view of the model. For more insight into building these diagrams, click on the links:

- **Structural Diagrams**
 - [Class Diagrams](#)
 - [Object Diagrams](#)
 - [Component Diagrams](#)
 - [Composite Structure Diagrams](#)
 - [Deployment Diagrams](#)
 - [Package Diagrams](#)
- **Behavioral Diagrams**
 - [Interaction Diagrams](#)
 - [Sequence Diagrams](#)
 - [Communication Diagrams](#)
 - [Interaction Overview Diagrams](#)
 - [Timing Diagrams](#)
 - [Activity Diagrams](#)
 - [Use Case Diagrams](#)
 - [State Machine Diagrams](#)



Refer to figure 464 (*UML 2.0 Superstructure*, p. 590).

Working with Diagrams

Diagrams are developed in the main workspace in which you create and connect model elements. They are

created by right-clicking a package and selecting *New Diagram*, or loaded by double-clicking their diagram icon in the Project Browser. For full details on how to work with diagrams, see [Common Diagram Tasks](#).

Non-UML Diagrams

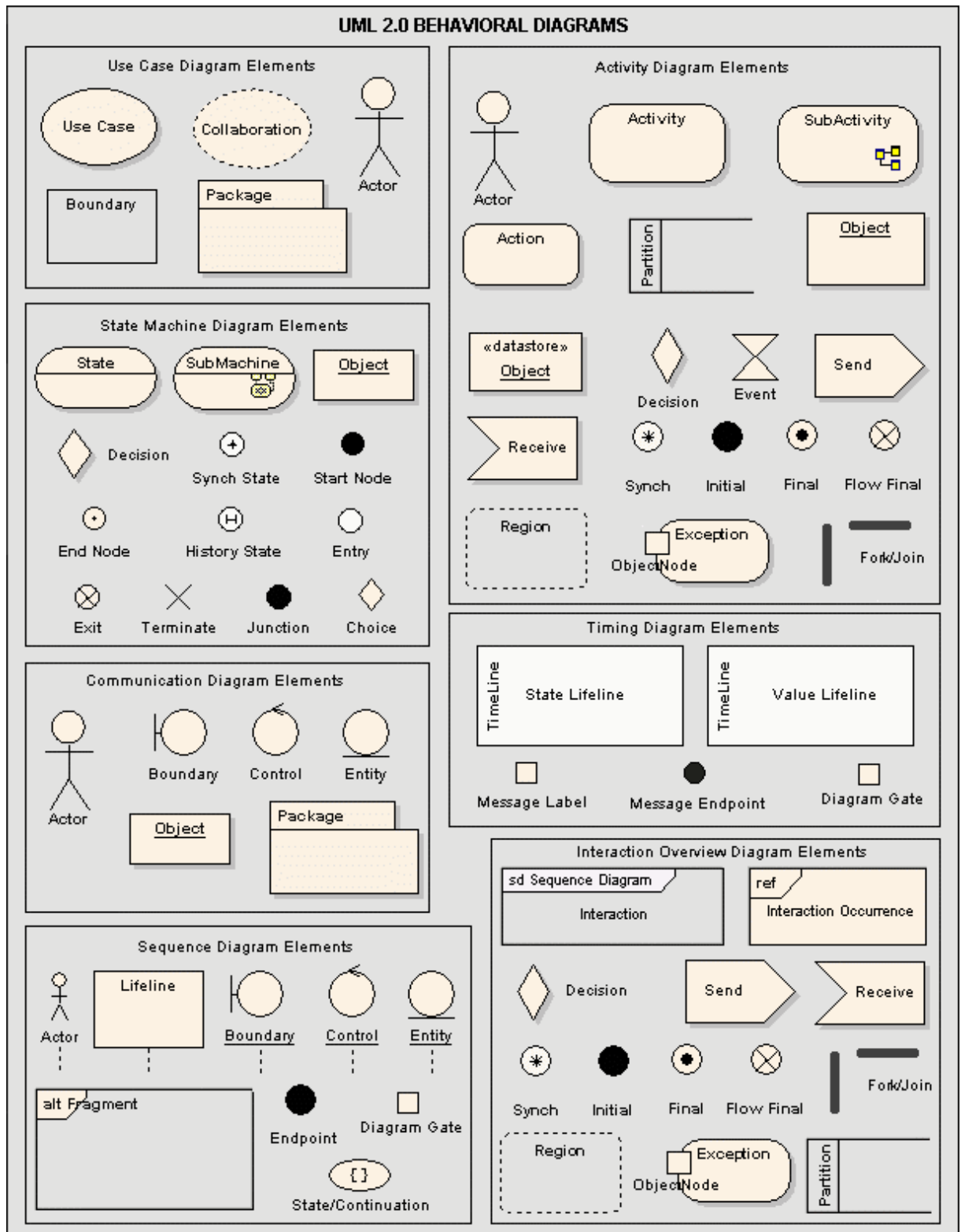
In addition to the 13 UML diagrams, EA provides a number of non-UML diagrams:

- [Analysis](#) diagrams
- [Custom](#) diagrams
- [Database](#) diagrams
- [Robustness](#) diagrams

5.2.1 Behavioral Diagrams

Behavioral diagrams depict the behavioral features of a system or business process. Behavioral diagrams include:

- [Activity diagrams](#)
- [Use Case diagrams](#)
- [State Machine diagrams](#)
- [Timing diagrams](#)
- [Sequence diagrams](#)
- [Communication diagrams](#)
- [Interaction Overview diagrams](#)



5.2.1.1 Activity Diagram

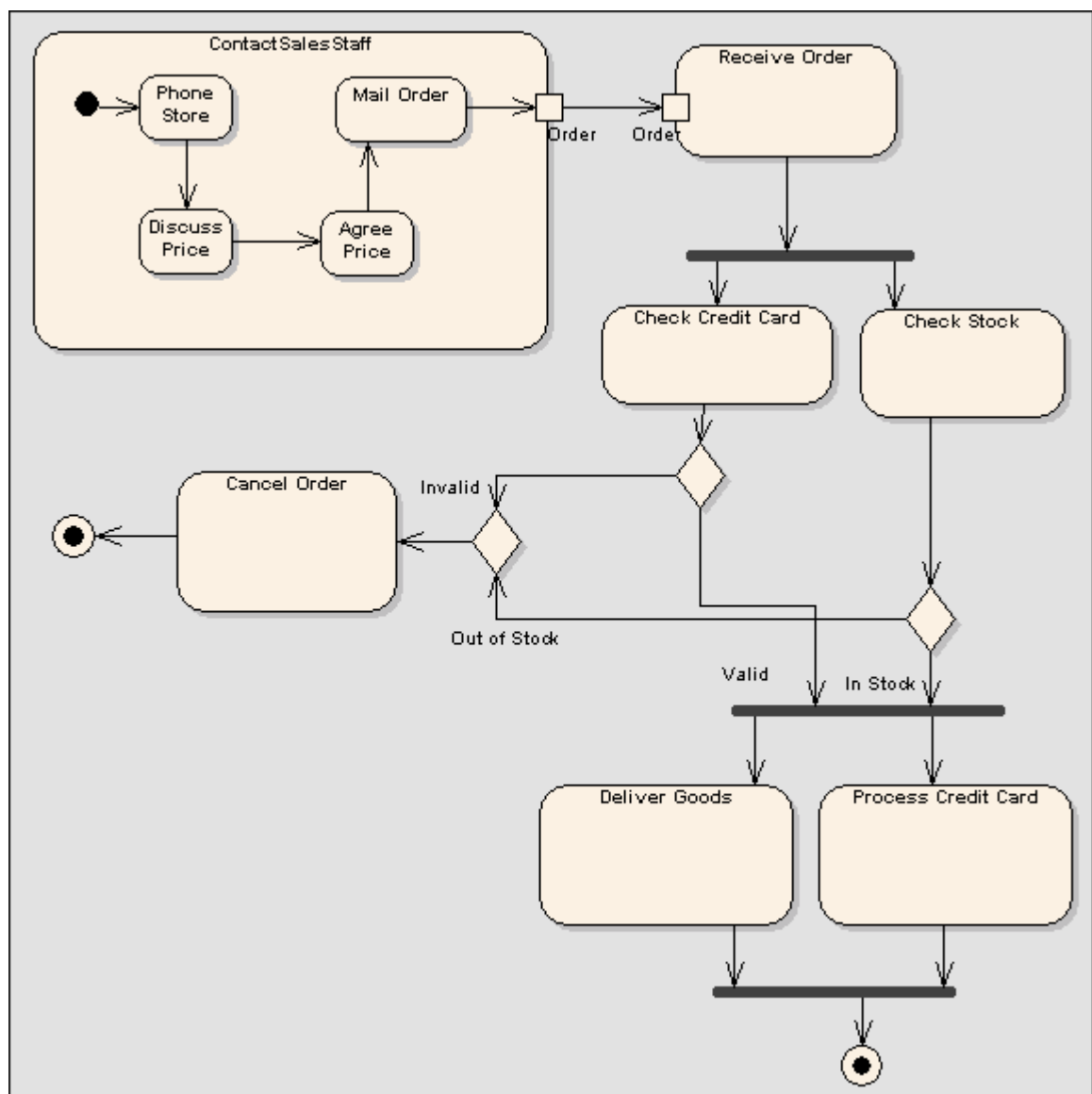
[Example Diagram](#)...[Elements/Connectors](#)...[Related Topics](#)...[OMG UML Specification](#)

Activity diagrams are used to model the behaviors of a system, and the way in which these behaviors are related in an overall flow of the system. The logical paths a process follows, based on various conditions, concurrent processing, data access, interruptions and other logical path distinctions, are all used to construct a process, system or procedure.

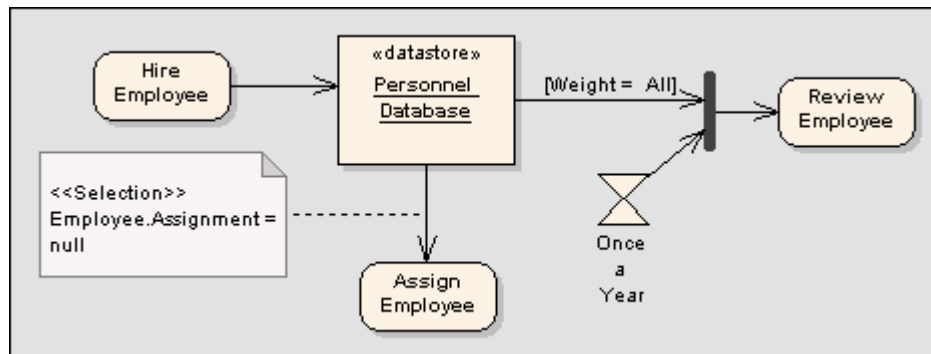
Note: Analysis diagrams (Simplified Activity), containing the elements most useful for business process modeling, can be created using the [New Diagram](#) dialog.

Example Diagram

The diagram below illustrates some of the features of Activity diagrams, including activities, actions, start nodes, end nodes and decision points.



Below is another simple example using actions, a datastore and a time event action.



Refer to figure 326 (*UML 2.0 Superstructure*, p. 319).

Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

Activity Diagram Elements	Activity Diagram Connectors
Activity	Fork/Join
Subactivity	Fork/Join
Action	Control Flow
Partition	Object Flow
Object	Dependency
Datastore	Interrupt Flow
Decision	
Send	
Receive	
Synch	
Initial	
Final	
Flow Final	
Region	
Exception	

Related Topics

- [Analysis Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 284) states:

"An activity specifies the coordination of executions of subordinate behaviors, using a control and data flow model. The subordinate behaviors coordinated by these models may be initiated because other behaviors in the model finish executing, because objects and data become available, or because events occur external to the flow. The flow of execution is modeled as activity nodes connected by activity edges. A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents. Activity nodes also include flow-of-control constructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to operations that are directly invoked.

"Activities may describe procedural computation. In this context, they are the methods corresponding to operations on classes. Activities may be applied to organizational modeling for business process engineering and workflow. In this context, events often originate from inside the system, such as the finishing of a task, but also from outside the system, such as a customer call. Activities can also be used for information system modeling to specify system level processes.

"Activities may contain actions of various kinds:

- occurrences of primitive functions, such as arithmetic functions.*
- invocations of behavior, such as activities.*
- communication actions, such as sending of signals.*
- manipulations of objects, such as reading or writing attributes or associations.*

"Actions have no further decomposition in the activity containing them. However, the execution of a single action may induce the execution of many other actions. For example, a call action invokes an operation which is implemented by an activity containing actions that execute before the call action completes."

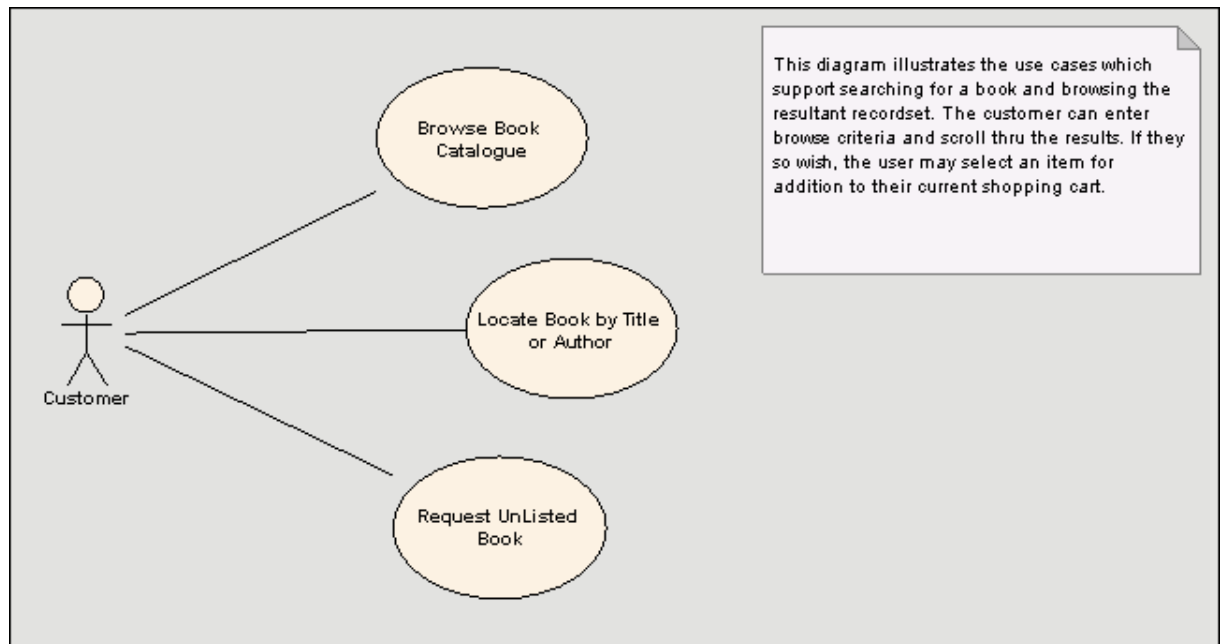
5.2.1.2 Use Case Diagram

[Example Diagram](#)..[Elements/Connectors](#)..[Related Topics](#)..[OMG UML Specification](#)

A *Use Case diagram* captures use cases and actor interactions. It describes the functional requirements of the system, the manner that outside things (actors) interact at the system boundary and the response of the system.














Example Diagram

The diagram below illustrates some features of Use Case diagrams:



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

<i>Use Case Diagram Elements</i>	<i>Use Case Diagram Connectors</i>
 Actor	 Use
 Use Case	 Associate
 Collaboration	 Generalize
 Boundary	 Include
 Package	 Extend
	 Realize
	 Dependency
	 Trace

Related Topics

- [Use Case Extension Points](#)
- [Using Rectangle Notation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

"A diagram that shows the relationships among actors and the subject (system), and use cases."

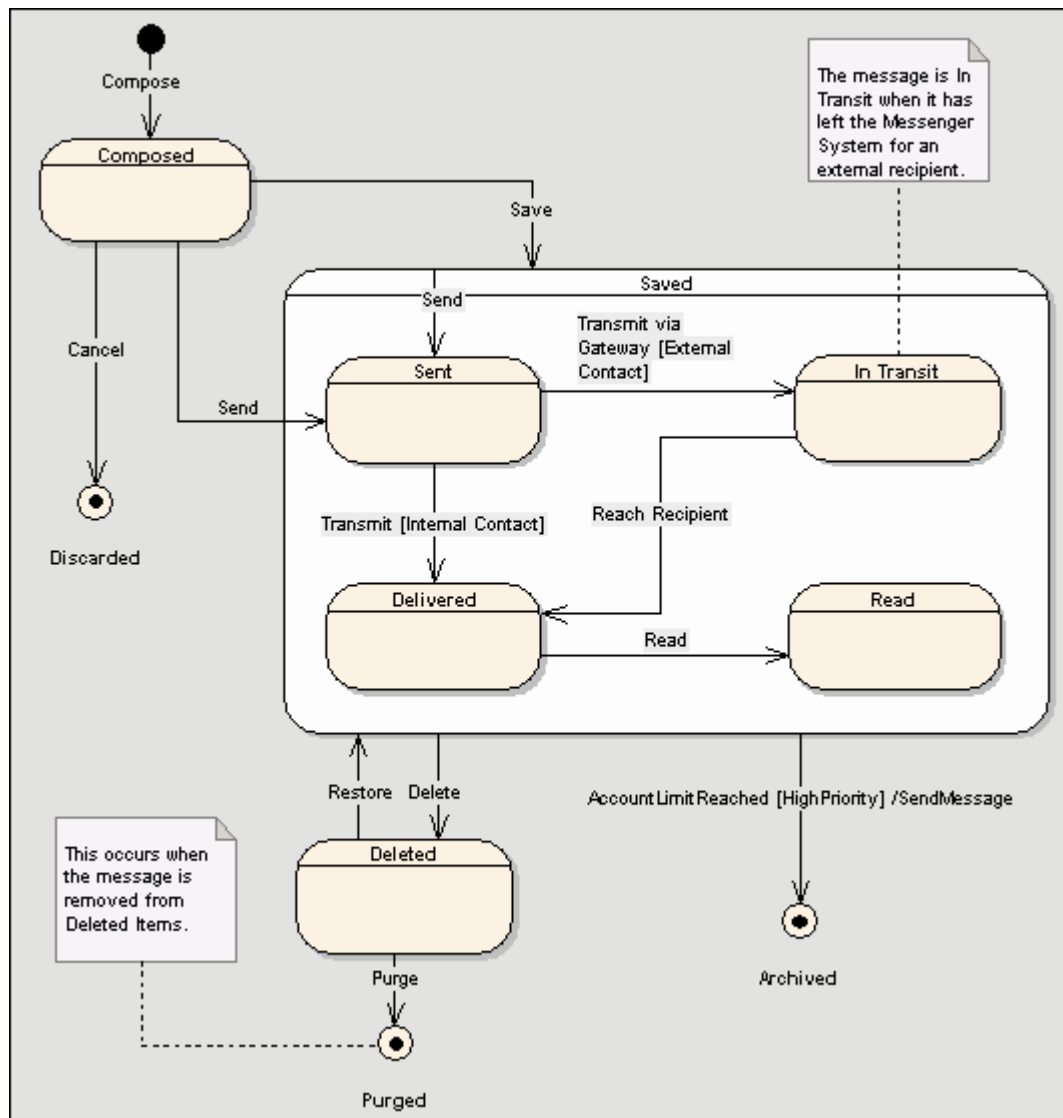
5.2.1.3 State Machine Diagram (formerly State Diagram)

[Example Diagram](#)...[Elements/Connectors](#)...[Related Topics](#)...[OMG UML Specification](#)

A *State Machine diagram* illustrates how an element, often a class, can move between states classifying its behavior, according to transition triggers, constraining guards and other aspects of State Machine diagrams that depict and explain movement and behavior.
















Example Diagram

The diagram below illustrates some features of State Machine diagrams. The *Saved* state is a composite state, and enclosed states are sub-states. Initial and final pseudo-states indicate the entry to and exit from the state machine.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

State Machine Diagram Elements	State Machine Diagram Connectors
 State	 Fork/Join
 Sub-Machine	 Fork/Join
 Initial	 Transition
 Final	 Object Flow
 History	
 Synch	
 Object	
 Decision	
 Junction	
 Entry	
 Exit	

Related Topics

- [Regions](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 490) states:

"A state machine owns one or more regions, which in turn own vertices and transitions.

"The behaviored classifier context owning a state machine defines which signal and call triggers are defined for the state machine, and which attributes and operations are available in activities of the state machine. Signal triggers and call triggers for the state machine are defined according to the receptions and operations of this classifier. As a kind of behavior, a state machine may have an associated behavioral feature (specification) and be the method of this behavioral feature. In this case the state machine specifies the behavior of this behavioral feature. The parameters of the state machine in this case match the parameters of the behavioral feature and provide the means for accessing (within the state machine) the behavioral feature parameters. A state machine without a context classifier may use triggers that are independent of receptions or operations of a classifier, i.e. either just signal triggers or call triggers based upon operation template parameters of the (parameterized) statemachine."

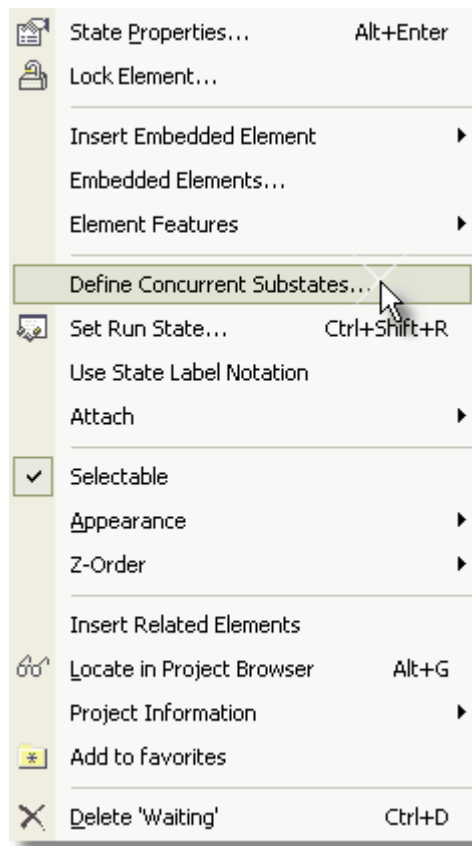
5.2.1.3.1 Regions

Related Topics...OMG UML Specification

Regions can be created in composite states or state machines. Regions indicate concurrency, such that a single state is active in each region. Multiple transitions can occur from a single event dispatch, so long as similarly triggered transitions are divided by regions.

To create a region in a composite state:

1. Right-click on a state, which will produce the following dialog:



2. Select *Define Concurrent Substates...*
3. This will open the *State Regions* dialog (below). Construct the regions of a state, which can be named or anonymous. Press *OK*.



Related Topics

- [Composite State](#)
- [State Machine Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 476) states:

"A region is an orthogonal part of either a composite state or a state machine. It contains states and transitions."

5.2.1.3.2 Pseudo-States

Pseudo-states are a UML 2 abstraction for various types of transient vertices used in State Machine diagrams. Pseudo-states are used to express complex transition paths. For more details on the kinds of pseudo-states, consult the list below.

Common Usage

- [Initial](#)
- [Entry Point](#)
- [Exit Point](#)
- [Choice](#)
- [Junction](#)
- [History](#)
- [Terminate](#)
- [Final](#)
- [Fork](#)
- [Join](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 469) states:

"A pseudostate is an abstraction that encompasses different types of transient vertices in the state machine graph... Pseudostates are typically used to connect multiple transitions into more complex state transitions paths. For example, by combining a transition entering a fork pseudostate with a set of transitions exiting the fork pseudostate, we get a compound transition that leads to a set of orthogonal target states."

5.2.1.4 Interaction Diagrams

An interaction is a generalization for a type of interaction diagram. Interaction diagrams can be any of the following:

- [Timing Diagrams](#)
- [Sequence Diagrams](#)
- [Interaction Overview Diagrams](#)
- [Communication Diagrams](#)

5.2.1.4.1 Timing Diagram

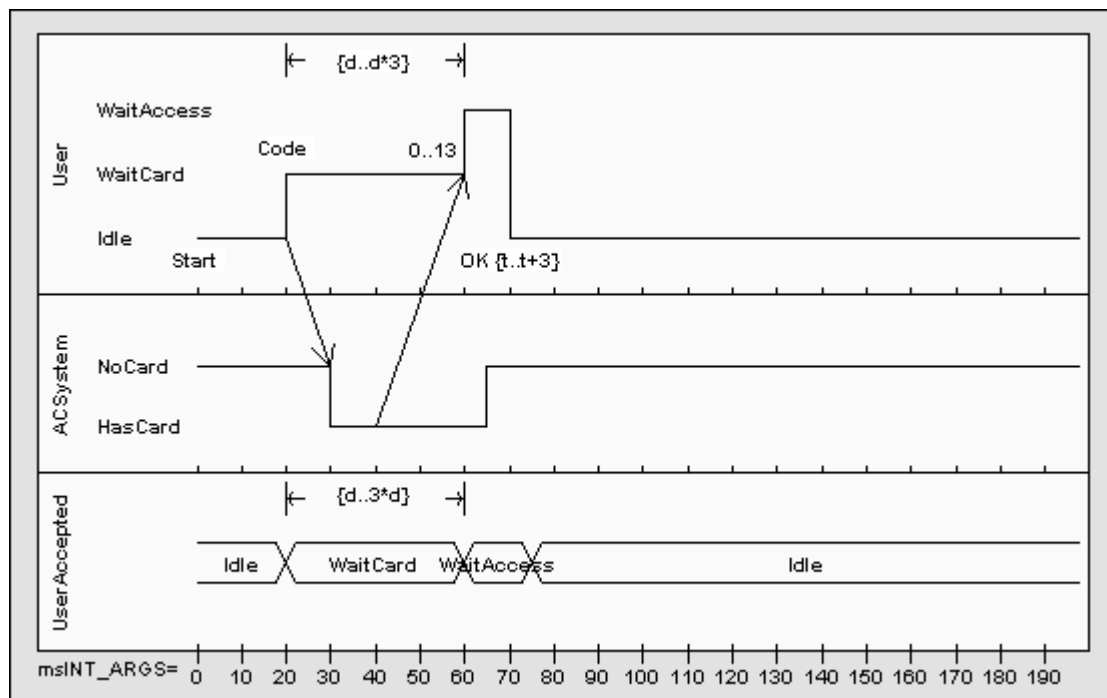
[Example Diagram...](#) [Elements/Connectors...](#) [Related Topics...](#) [OMG UML Specification](#)

The *Timing diagram* defines the behavior of different objects within a time-scale. It provides a visual representation of objects changing state and interacting over time.

Timing diagrams can be used for defining hardware-driven or embedded software components. This may be embedded software such as that used in a fuel injection system, a microwave controller, etc. They can also be used for specifying time-driven business processes.

Example Diagram







Below is an example Timing diagram:



Refer to figures 351 and 352 (*UML 2.0 Superstructure*, p. 454).

Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

<i>Timing Diagram Elements</i>	<i>Timing Diagram Connectors</i>
 State Lifeline	
 Value Lifeline	
 Message Label	
 Message Endpoint	
 Diagram Gate	
 Message	

Related Topics

- [Timing Message](#)
- [State Lifeline](#)
- [Value Lifeline](#)
- [Timing Message](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 450) states:

"Timing Diagrams are used to show interactions when a primary purpose of the diagram is to reason about time. Timing diagrams focus on conditions changing within and among Lifelines along a linear time axis. Timing diagrams describe behavior of both individual classifiers and interactions of classifiers, focusing attention on time of occurrence of events causing changes in the modeled conditions of the Lifelines."

The OMG UML specification (*UML 2.0 Superstructure*, p. 453) also states:

"The primary purpose of the timing diagram is to show the change in state or condition of a lifeline (representing a Classifier Instance or Classifier Role) over linear time. The most common usage is to show the change in state of an object over time in response to accepted events or stimuli. The received events are annotated as shown when it is desirable to show the event causing the change in condition or state."

5.2.1.4.2 Sequence Diagram

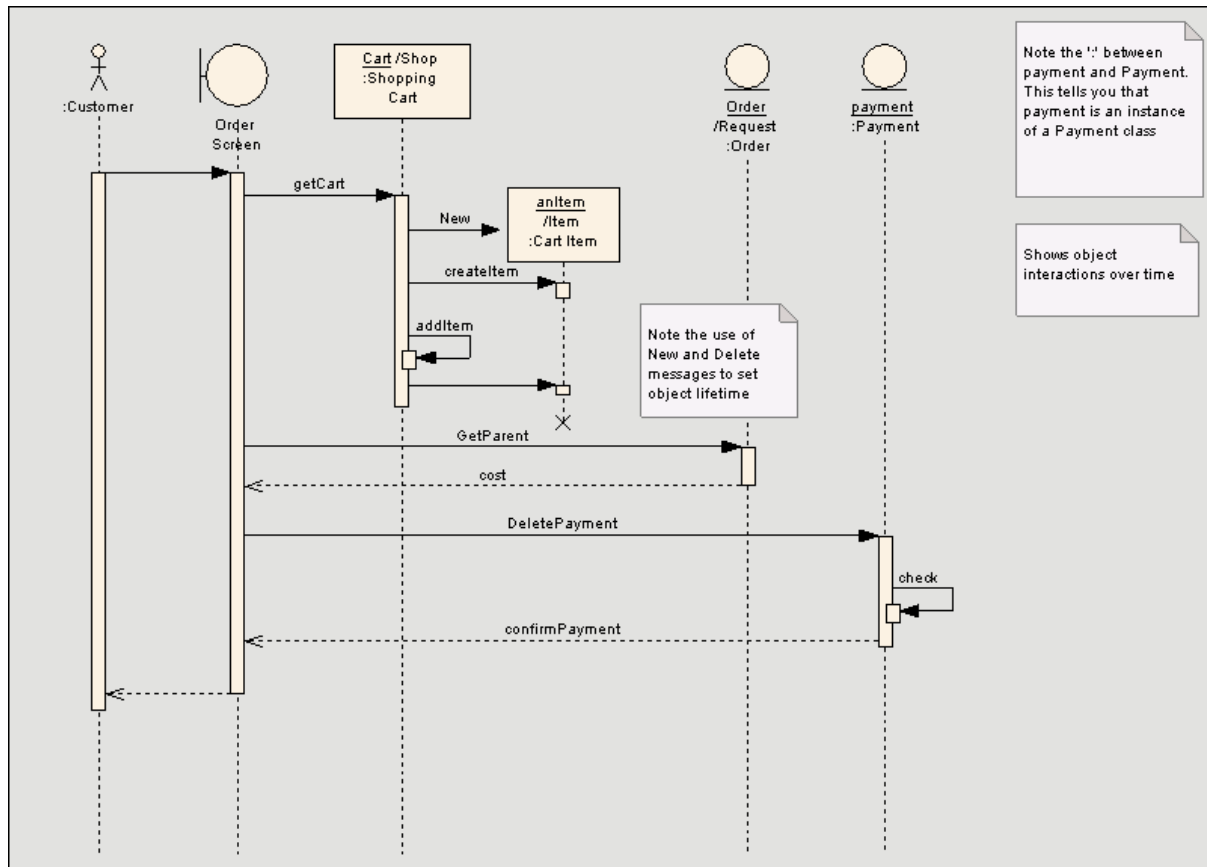
[Example Diagram](#).. [Elements/Connectors](#).. [Related Topics](#).. [OMG UML Specification](#)

A *Sequence diagram* is a structured representation of behavior as a series of sequential steps over time. It is used to depict work flow, message passing and how elements in general cooperate over time to achieve a result.

- Each [sequence element](#) is arranged in a horizontal sequence, with messages passing back and forward between elements.
- An actor element may be used to represent the user initiating the flow of events.
- Stereotyped elements, such as [boundary](#), [control](#) and [entity](#), may be used to illustrate screens, controllers, and database items, respectively.
- Each element has a dashed stem called a lifeline, where that element exists and potentially takes part in the interactions.

Example Diagram

Below is an example Sequence diagram demonstrating several different elements:



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

Sequence Diagram Elements	Sequence Diagram Connectors
Actor	Message
Lifeline	Self-Message
Boundary	Recursion
Control	Call
Entity	
Fragment(s)	
Endpoint	
Diagram Gate	
State/Continuation	

Related Topics

- [Denote the Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Message Label Visibility](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 14) states:

"A diagram that depicts an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding event occurrences on the lifelines.

"Unlike a communication diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and communication diagrams express similar information, but show it in different ways."

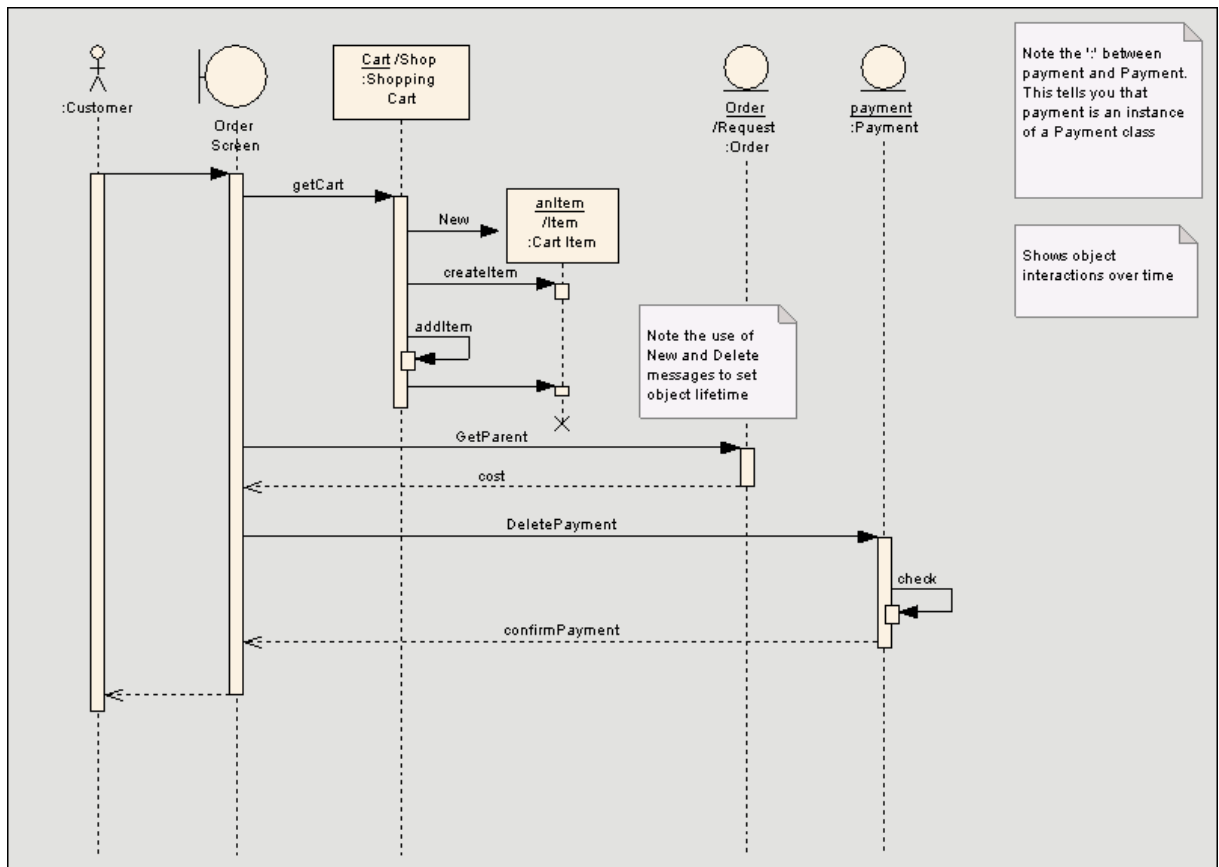
5.2.1.4.2.1 Denote Lifecycle of an Element

Related Topics

You can capture element lifetimes using messages which are denoted as *New* or *Delete* message types. To do this, follow the steps below:

1. Right-click on a message within a Sequence diagram to view the context menu.
2. Select the *Message Properties...* option to view the *Message Properties* dialog.
3. From the *Lifecycle* selection menu, choose *New* or *Delete*.
4. Press *OK* to save changes.

The example below shows two elements that have specific creation and deletion times.



Related Topics

- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)

5.2.1.4.2.2 Layout of Sequence Diagrams

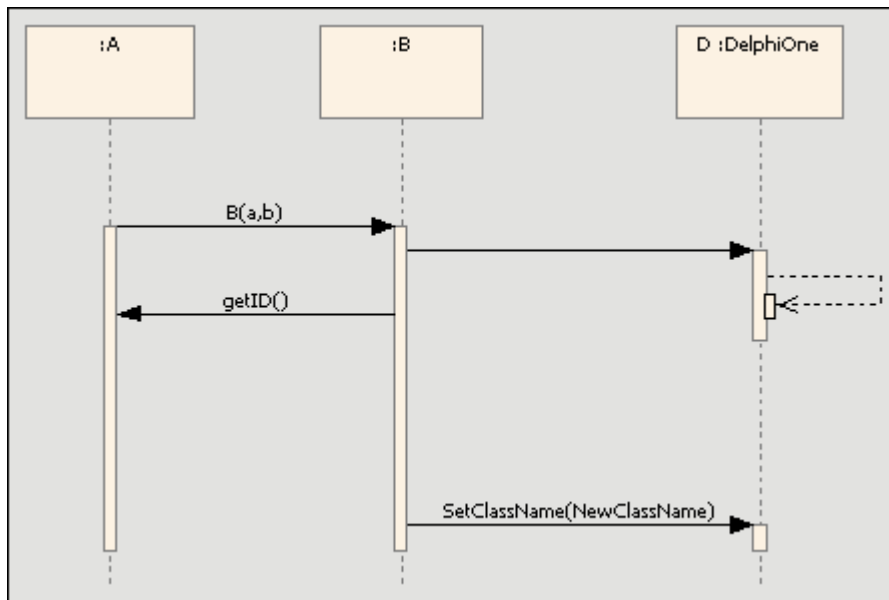
Related Topics

You can modify the vertical height of sequence messages to get an attractive and effective layout. To offset message positions, follow the steps below:

1. Select the appropriate message in a sequence diagram.
2. Use the mouse to drag the message up or down as required.

Warning: If you drag up or down past the next or previous message, EA will interpret that as the desire to swap positions, rather than simply offset a message position.

The example below shows an economical use of space in a Sequence diagram.



Related Topics

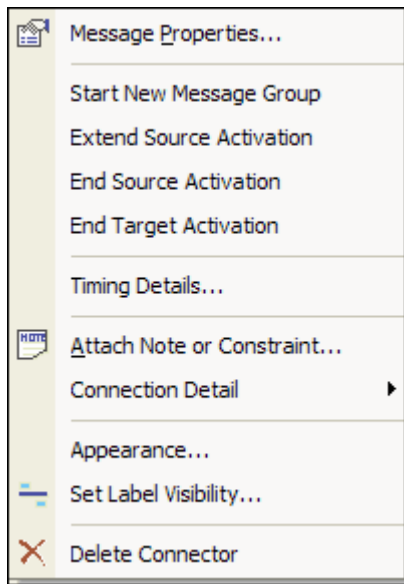
- [Denote Lifecycle of an Element](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)
- [Message Label Visibility](#)

5.2.1.4.2.3 Sequence Element Activation

Related Topics

Sequence elements in a [Sequence diagram](#) have a focus of control or *Activation* rectangle drawn along their lifeline. This rectangle describes the time the element is active during the overall period of processing. This visual representation can be suppressed by right-clicking the sequence diagram, and selecting "Suppress Focus of Control".

In general, EA will calculate this period of Activation for you - but in some cases you may wish to fine tune the rectangle length. There are several context menu options on a sequence message that you can use to accomplish this. To access the following context menu, *right-click* on the message.



Start New Message Group: will start off a new round of processing in the current diagram. This allows you to describe more than one processing scenario in a single diagram.

Extend Source Activation: Use this option to force an element to stay active beyond the normal processing period. This could be used to express an element which continues its own processing concurrently with other processes.

End Source Activation: this option will truncate the activation of the source element after the current message. This is useful where you need to express an asynchronous message after which the source element becomes idle.

End Target Activation: Use this to end a Forced Activation started by the "Extend Source Activation" command.

The commands **Extend Method Invocation** and **Raise Activation Level** will appear on the context menu only where they can be used. So, for example, after a self-message the next message will start by default at a lower activation level but will provide an **Extend Method Invocation** command on its context menu to allow you to raise its level. **Lower Activation Level** also appears where appropriate.

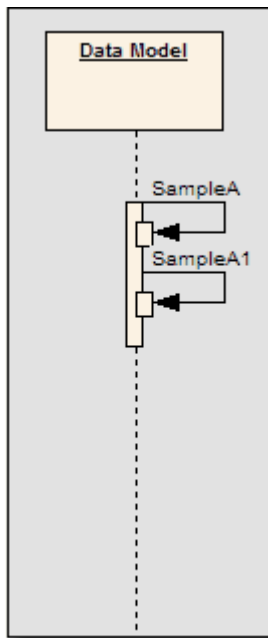
Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Lifeline Activation Levels](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)

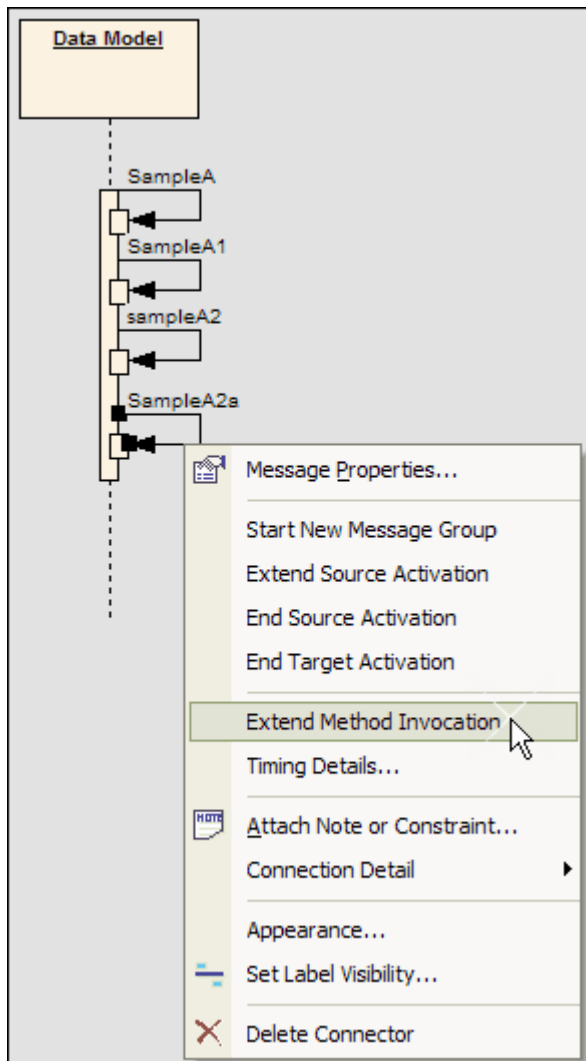
5.2.1.4.2.4 Lifeline Activation Levels

Related Topics

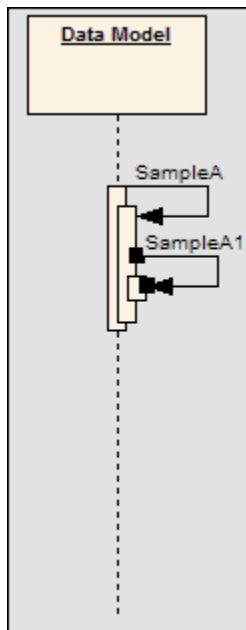
Complicated processing systems can be easily negotiated and reflected in Sequence diagrams, by adding activation layers on a single lifeline. For example, the below class invokes the method sampleA, which in turn calls sampleA1. By dragging the self-message connectors onto the lifeline, the following will appear:



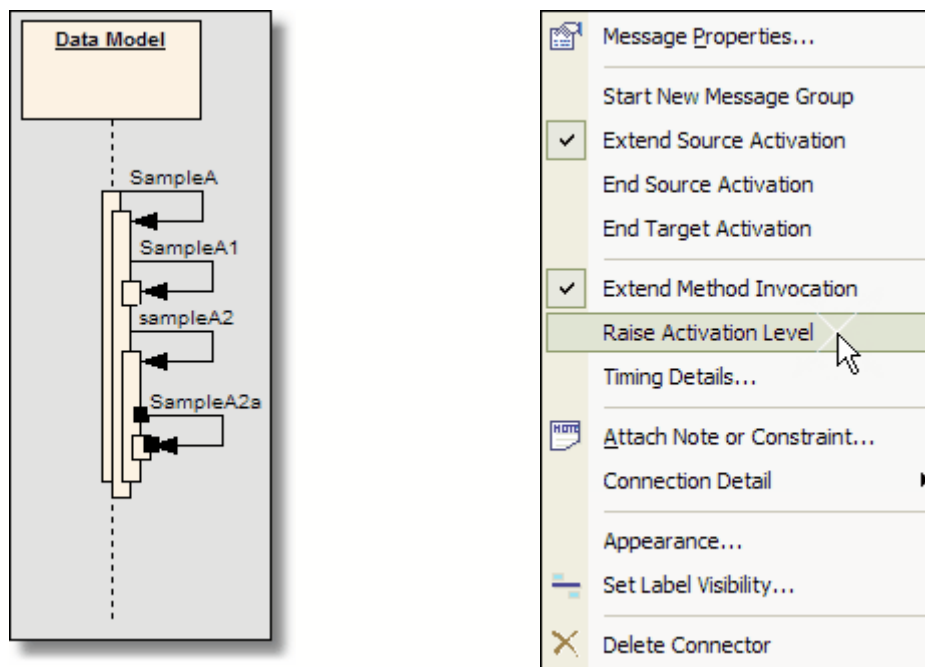
Right-click the invoked method, which in this case is sampleA1. Select *Extend Method Invocation*.



The lifeline now visually depicts that method sampleA1 is called during the processing of sampleA.



To visualize layers of internal processing, this same process can be applied to any internally invoked method. To indicate a new layer of processing, these self-messages extending method invocation have a new option. In the example below, a few more self-messages have been added. To depict that a message sampleA2a is called from sampleA2, as shown below, right-click on the message and select *Raise Activation Level*. It is only possible to raise the activation level of a message when the message has been had the source activation extended.



With multiple layers of activation, the option *Lower Activation Level* will also appear in the context menu, to control processing levels.

Related Topics

- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Sequence Elements](#)
- [Sequence Diagram](#)
- [Message Label Visibility](#)

5.2.1.4.2.5 Sequence Elements**Related Topics**

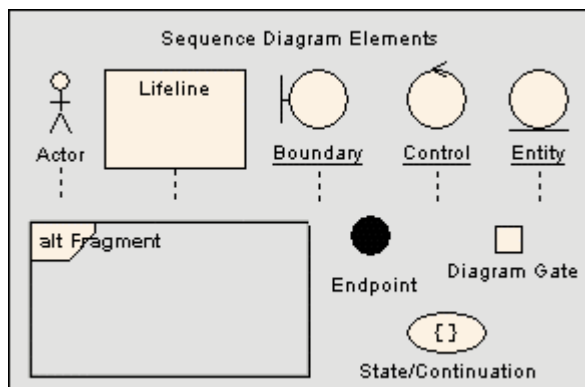
A [Sequence diagram](#) models a dynamic view of the interactions between model elements at runtime.

Sequence diagrams are commonly used as explanatory models for use case scenarios. By creating a sequence diagram with an actor and elements involved in the use case, you can model the sequence of steps the user and the system undertake to complete the required tasks. An element in a Sequence diagram is usually either an actor (the stimulus that starts the interaction) or collaborating elements.

Note: A Sequence diagram is often attached directly under a use case to which it refers. This helps keep elements together, both in the model and when documentation is produced. To do this, right-click the use case on the diagram and select **Composite Element**. Alternatively, from the Project Browser, right-click the use case and select **New Child Diagram**.

The example below shows some possible elements of sequence diagrams and their stereotyped display.

- *Actor* - An instance of an actor at runtime.
- *Object* - A standard element - not typed.
- *Boundary* - Represents a user interface screen or input/output device
- *Entity* - A persistent element - typically implemented as a database table or element.
- *Controller* - The active component that controls what work gets done, when and how.



Tip: Use Sequence diagrams early in analysis to capture the flow of information and responsibility throughout the system. Messages between elements will eventually become method calls in the class model.

Related Topics

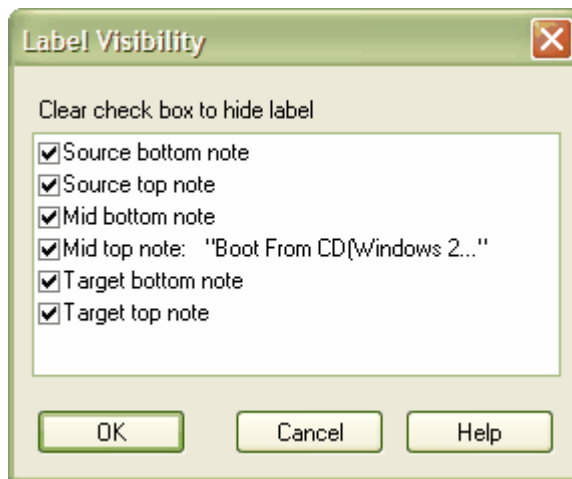
- [Denote Lifecycle of an Element](#)
- [Layout of Sequence Diagrams](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)
- [Sequence Diagram](#)

5.2.1.4.2.6 Sequence Message Label Visibility

[Related Topics](#)

Sequence messages may have the control of the label visibility controlled via the message context menu. To hide and show the labels used in sequence messages use the following instructions:

1. Right click on the message within the sequence diagram.
2. From the message context menu select *Set Label Visibility*.
3. Show or hide message labels by checking and unchecking the label from the list as appropriate.



Related Topics

- [Lifeline Activation Levels](#)
- [Layout of Sequence Diagrams](#)

5.2.1.4.3 Communication Diagram (formerly Collaboration Diagram)

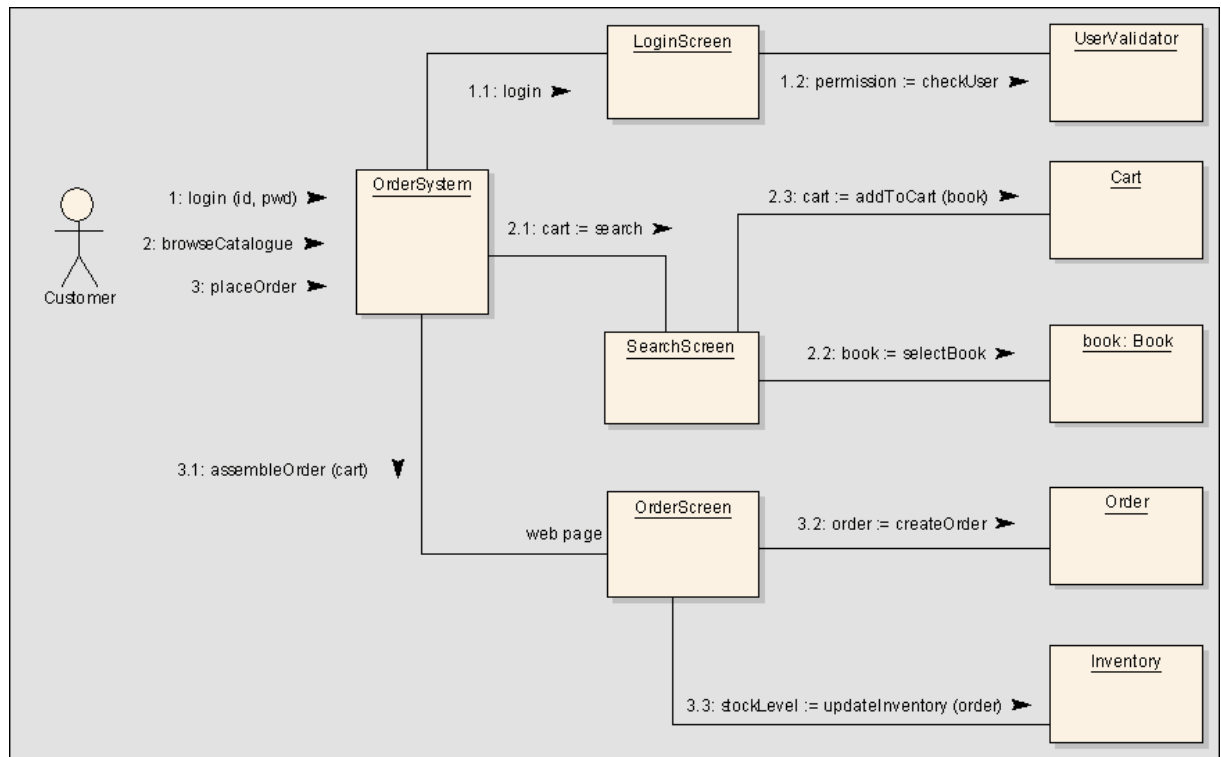
[Example Diagram](#)..|.. [Elements/Connectors](#)..|.. [Related Topics](#)..|.. [OMG UML Specification](#)

A *Communication diagram* shows the interactions between elements at run-time in much the same manner as a Sequence diagram. However, Communication diagrams are used to visualize inter-object relationships, while Sequence diagrams are more effective at visualizing processing over time.

Communication diagrams employ ordered, labeled associations to illustrate processing. Numbering is important to indicate the order and nesting of processing. A numbering scheme could be 1, 1.1, 1.1.1, 1.1.2, 1.2, etc. A new number segment begins for a new layer of processing, and would be equivalent to a method invocation.

Example Diagram

The example below illustrates a Communication diagram among co-operating object instances. Note the use of message levels to capture related flows.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

Communication Diagram Elements	Communication Diagram Connectors
Actor	Associate
Object	Dependency
Boundary	Realize
Control	Nesting
Entity	
Package	

Related Topics

- [Messages for Communication Diagrams](#)
- [Communication Diagrams in Color](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure, p. 14*) states:

"A diagram that focuses on the interaction between lifelines where the architecture of the internal structure and how this corresponds with the message passing is central. The sequencing of messages is given through a sequence numbering scheme. Sequence diagrams and communication diagrams express similar information, but show it in different ways."

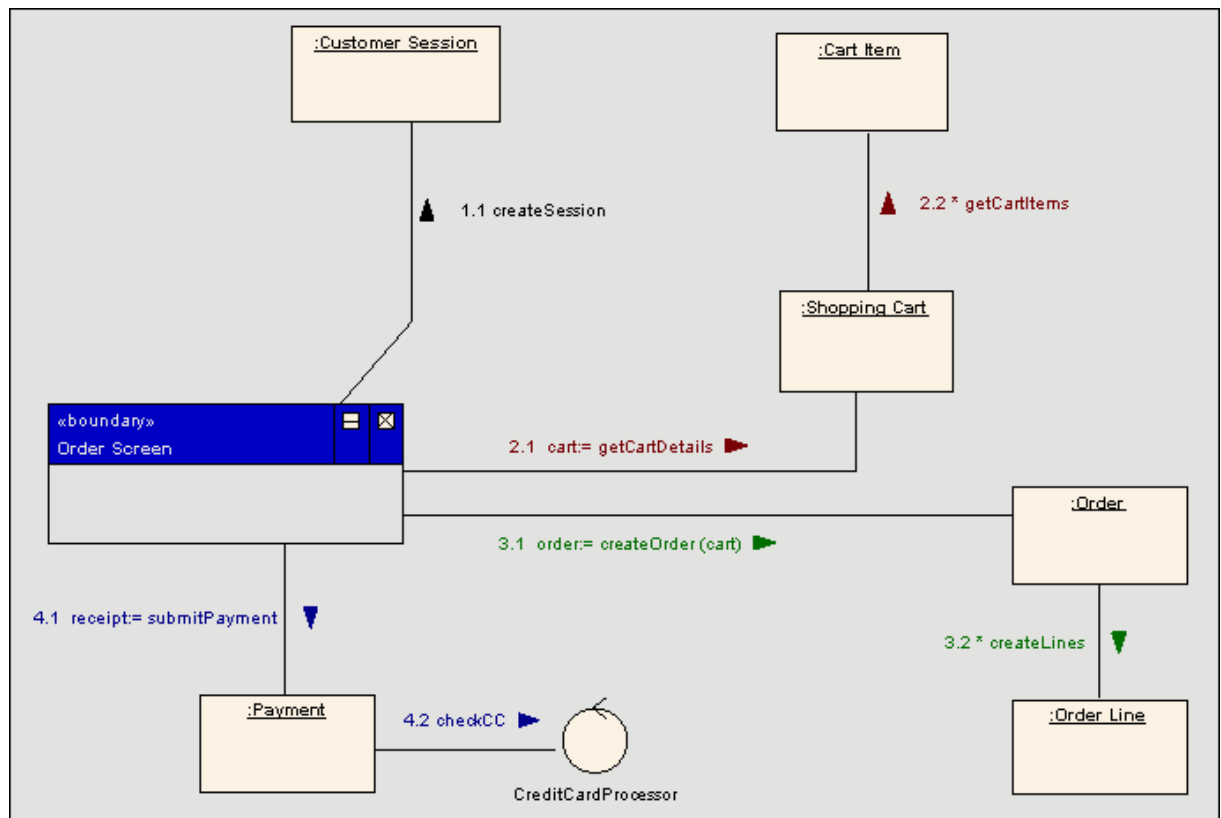
Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.2.1.4.3.1 Communication Diagrams in Color

Enterprise Architect allows you to highlight particular message flows in a Communication diagram using different colors for each message set.

To highlight the colors in a Communication diagram, follow the steps below:

1. Open the **Tools | Options** dialog and go to the **Communication Colors** tab.
2. Make sure the **Use Communication Color** check box is ticked.
3. Select the desired colors.
4. On your Communication diagram, each sequence group of messages will now appear in a different color as shown below.



Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.2.1.4.4 Interaction Overview Diagram

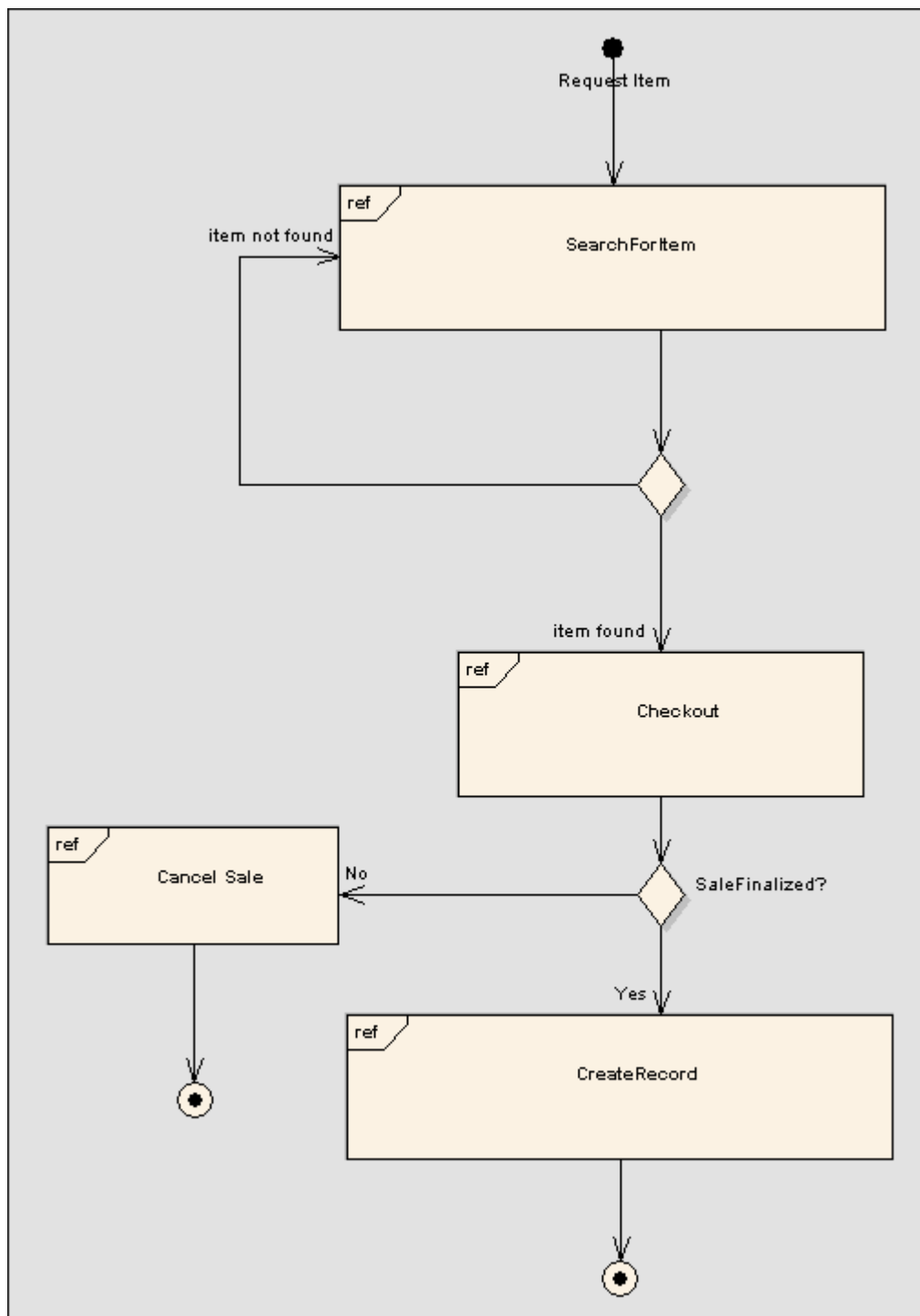
[Example Diagram](#) | [Elements/Connectors](#) | [Related Topics](#) | [OMG UML Specification](#)

Interaction Overview diagrams visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose. As Interaction Overview diagrams are a variant of activity diagrams, most of the diagram notation is similar, as is the process in constructing the diagram. Decision points, forks, joins, start points, and end points are the same. Instead of activity elements, however, rectangular elements are used. There are two types of these elements, interaction elements and interaction occurrence elements. Interaction elements display an inline Interaction diagram, which can be a Sequence diagram, Communication diagram, Timing diagram, or Interaction Overview diagrams. Interaction occurrence elements are references to an existing Interaction diagram. They are visually represented by a frame, with "ref" in the frame's title space. The diagram name is indicated in the frame contents.

















To create an interaction occurrence, simply drag an Interaction diagram from the Project Browser onto your Interaction Overview diagram. The "ref" frame will appear, encapsulating an instance of the Interaction diagram.

Example Diagram

The following example depicts a sample sale process, shown in an Interaction Overview diagram, with sub-processes abstracted within interaction occurrences. The diagram appears very similar to an activity diagram, and is conceptualized the same way; as the flow moves into an interaction, that respective interaction's process must be followed before the interaction overview's flow can advance.

**Toolbox Elements and Connectors**

Tip: Click the elements and connectors below for more information.

<i>Interaction Overview Diagram Elements</i>	<i>Interaction Overview Diagram Connectors</i>
 Partition	 Fork/Join
 Decision	 Fork/Join
 Send	 Control Flow
 Receive	 Object Flow
 Synch	 Dependency
 Initial	 Interrupt Flow
 Final	
 Flow Final	
 Region	
 Exception	

Related Topics

- [Activity Diagram](#)
- [Sequence Diagram](#)
- [Communication Diagram](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 447) states:

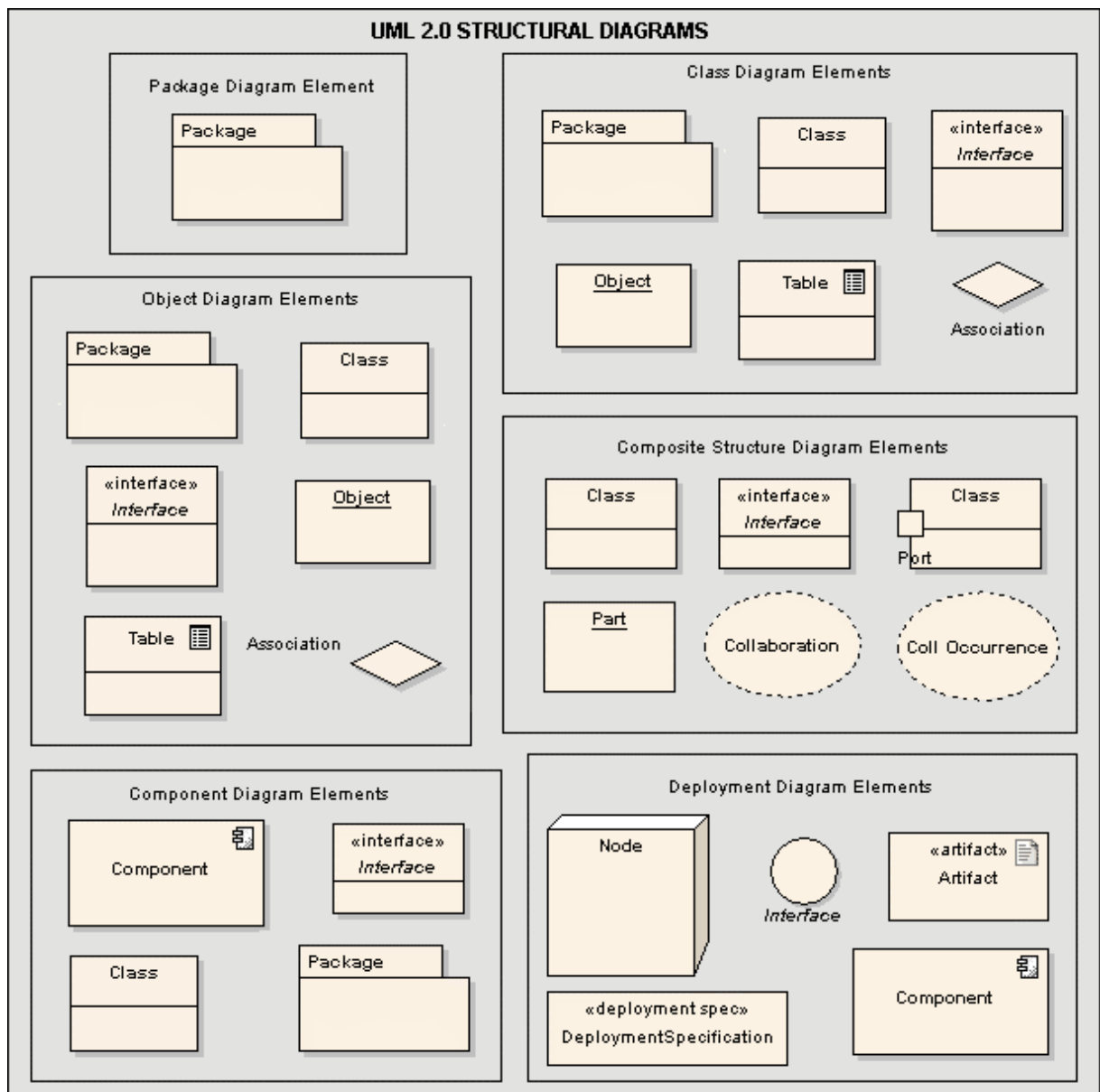
"Interaction Overview Diagrams define Interactions (described in Chapter 14, "Interactions") through a variant of Activity Diagrams (described in Chapter 6, "Activities") in a way that promotes overview of the control flow. Interaction Overview Diagrams focus on the overview of the flow of control where the nodes are Interactions or InteractionOccurrences. The Lifelines and the Messages do not appear at this overview level."

5.2.2 Structural Diagrams

Structural diagrams depict the structural elements composing a system or function. These diagrams can reflect the static relationships of a structure, as do class or package diagrams, or run-time architectures, such as Object or Composite Structure diagrams.

Structural diagrams include:

- [Class diagrams](#)
- [Composite Structure diagrams](#)
- [Component diagrams](#)
- [Deployment diagrams](#)
- [Object diagrams](#)
- [Package diagrams](#)



5.2.2.1 Package Diagram

[Example Diagram..|. Elements/Connectors..|.OMG UML Specification](#)

Package diagrams are used to reflect the organization of packages and their elements, and provide a visualization of their corresponding namespaces.

Example Diagram

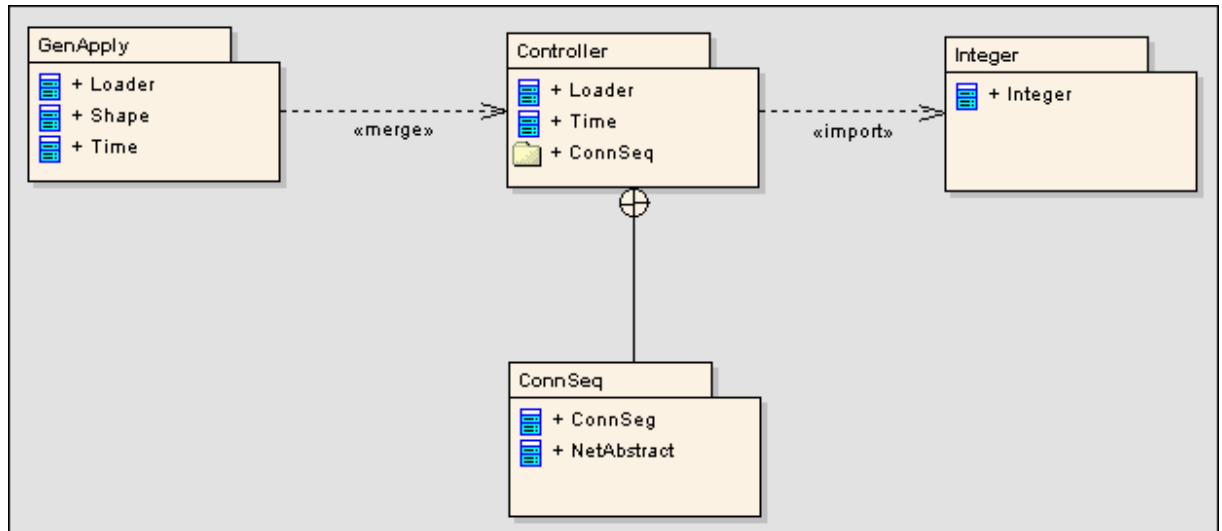
The following example demonstrates a basic Package diagram. The nesting connector between ConnSeq and Controller reflects what the package contents reveal. Package contents can be listed by accessing the diagram's property window, and selecting "Show Package Contents".

The <<import>> connector indicates that the elements within the target Integer package, which in this

example is a single class, Integer, will be imported into the package Controller. The Controller's namespace will gain access to the Integer class; the Integer namespace is not affected.



















The <<merge>> connector indicates that the package Controller's elements will be imported into GenApply, including Controller's nested and imported contents. If an element already exists within GenApply, such as Loader and Time, these elements' definitions will be expanded by those included in the package Controller. All elements added or updated by the merge are noted by a generalization relationship back to that package.

Note: Private elements within a package cannot be imported or merged.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

<i>Package Diagram Elements</i>	<i>Package Diagram Connectors</i>
 Package	 Association
 Class	 Associate
 Interface	 Generalize
 Object	 Compose
 Table	 Aggregate
	 Association Class
	 Assembly
	 Dependency
	 Realize
	 Trace
	 Nesting
	 Pkg Merge
	 Pkg Import

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

"A diagram that depicts how model elements are organized into packages and the dependencies among them, including package imports and package extensions."

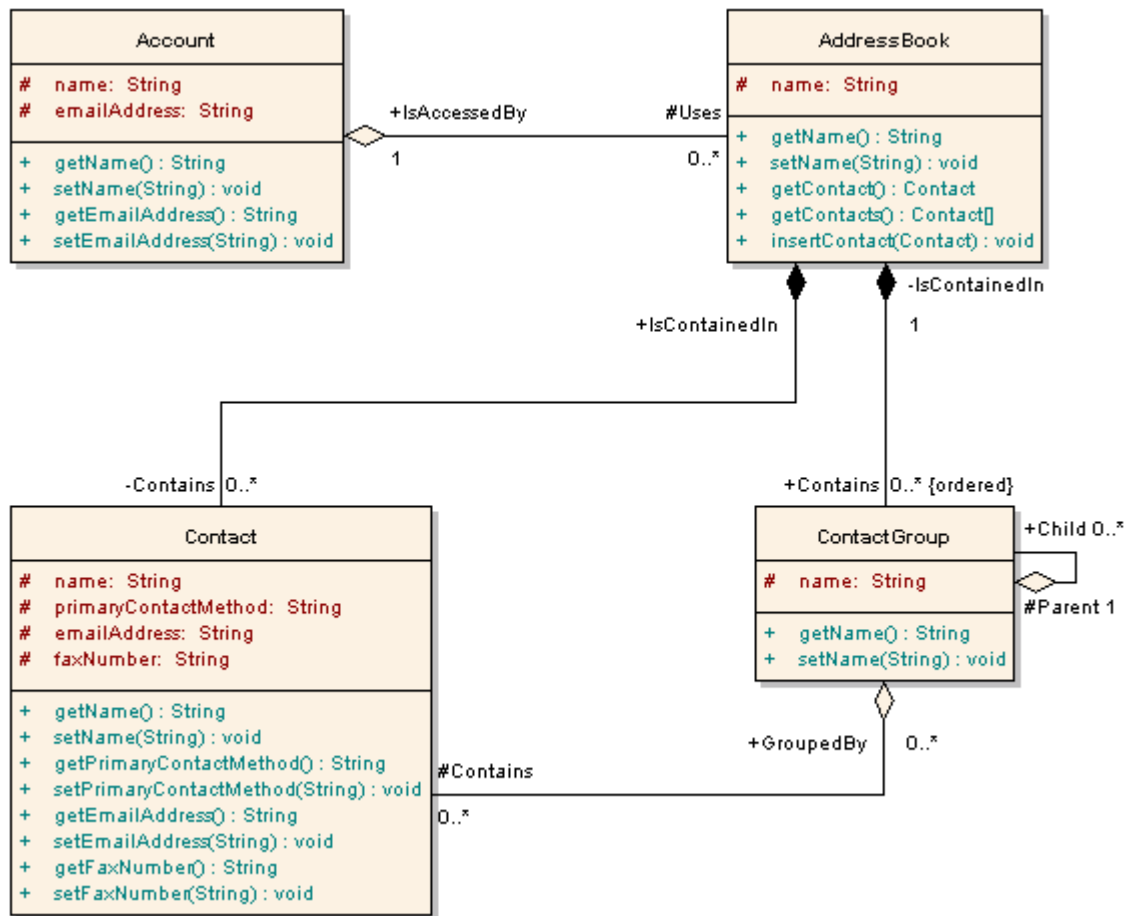
5.2.2.2 Class Diagram

[Example Diagram](#)...[Elements/Connectors](#)...[Related Topics](#)...[OMG UML Specification](#)

The *Class diagram* captures the logical structure of the system - the classes and things that make up the model. It is a static model, describing what exists and what attributes and behavior it has, rather than how something is done. Class diagrams are most useful to illustrate relationships between classes and interfaces. Generalizations, aggregations, and associations are all valuable in reflecting inheritance, composition or usage, and connections, respectively.



















Example Diagram

The lighter aggregation indicates that the class Account uses AddressBook, but does not necessarily contain AddressBook. The strong, composite aggregation by the other connectors indicate ownership or containment by the target classes (at the diamond end) of the source classes.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

Class Diagram Elements	Class Diagram Connectors
 Package	 Association
 Class	 Associate
 Interface	 Generalize
 Object	 Compose
 Table	 Aggregate
	 Association Class
	 Assembly
	 Dependency
	 Realize
	 Trace
	 Nesting
	 Pkg Merge
	 Pkg Import

Related Topics

- [Parameterized Classes \(Templates\)](#)
- [Active Classes](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 6) states:

"A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships."

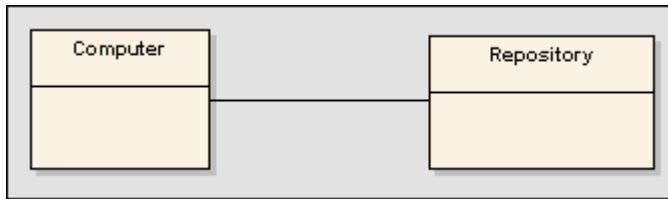
5.2.2.3 Object Diagram

[Example Diagram](#)..[Elements/Connectors](#)..[Related Topics](#)..[OMG UML Specification](#)

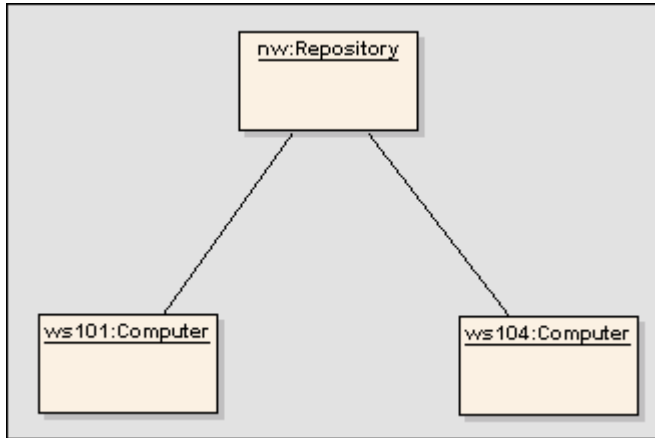
An *Object diagram* is closely related to a Class diagram, with the distinction that it depicts object instances of classes at a point in time. This might appear similar to a Composite Structure diagram, which also models run-time behavior; the difference is that Object diagrams exemplify the static Class diagrams, whereas Composite Structure diagrams reflect run-time architectures different from their static counterparts. Object diagrams do not reveal architectures varying from their corresponding Class diagrams, but reflect multiplicity and the roles instantiated classes could serve. They are useful in understanding a complex Class diagram, by creating different cases in which the relationships and classes are applied. An Object diagram can also be a kind of Communication diagram, which also models the connections between objects, but additionally sequences events along each path.

Example Diagram

The following example first shows a simple Class diagram, with two class elements connected.




















The classes above are instantiated below as objects in an Object diagram. There are two instances of Computer in this model, which can prove useful for considering the relationships and interactions classes play in practice, as objects.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

<i>Object Diagram Elements</i>	<i>Object Diagram Connectors</i>
 Package	 Association
 Interface	 Associate
 Object	 Generalize
 Table	 Compose
	 Aggregate
	 Association Class
	 Assembly
	 Dependency
	 Realize
	 Trace
	 Nesting
	 Pkg Merge
	 Pkg Import

Related Topics

- [Composite Structure Diagram](#)
- [Class Diagram](#)
- [Communication Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

"A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a communication diagram."

Note: *Communication diagrams were known as Collaboration diagrams in UML 1.4.*

5.2.2.4 Composite Structure Diagram

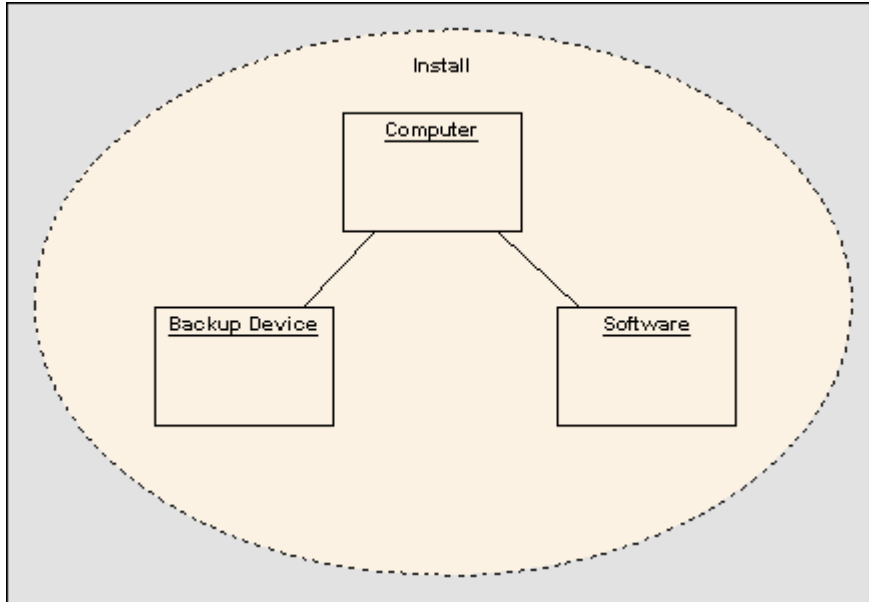
[Example Diagram](#) .. [Elements/Connectors](#) .. [Related Topics](#) .. [OMG UML Specification](#)

A *Composite Structure diagram* reflects the internal collaboration of classes, interfaces or components to describe a functionality. Composite Structure diagrams are similar to Class diagrams, except that they model a specific usage of the structure. Class diagrams model a static view of class structures, including their attributes and behaviors. A Composite Structure diagram is used to express run-time architectures, usage patterns, and the participating elements' relationships, which might not be reflected by static diagrams.

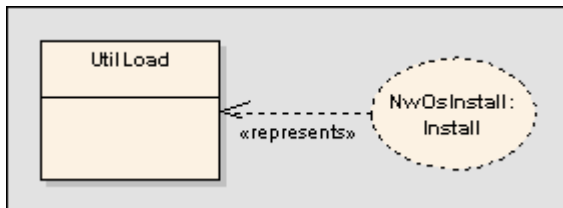
In a Composite Structure diagram, classes are accessed as parts or run-time instances fulfilling a particular role. These parts can have multiplicity, if the role filled by the class requires multiple instances. Ports defined by a part's class should be represented in the composite structure, maintaining that all connecting parts provide the required interfaces specified by the port. There is extensive flexibility, and an ensuing complexity, that come with modeling composite structures. To optimize your modeling, consider building collaborations to represent reusable patterns responding to your design issues.

Example Diagram

The following diagram shows a collaboration used in Composite Structure diagrams to model common patterns. This particular example shows a relationship for performing an installation.














The following diagram uses the Install collaboration in a collaboration occurrence, and applies it to the UtilLoad class via a <<represents>> relationship. This indicates that the classifier UtilLoad uses the collaboration pattern within its implementation. For further examples on composite structure diagrams, refer to the toolbox elements listed below.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

Composite Structure Diagram Elements	Composite Structure Diagram Connectors
 Class	 Connector
 Interface	 Assembly
 Part	 Role Binding
 Port	 Represents
 Collaboration	 Occurrence
 Expose Interface	

Related Topics

- [Properties](#)
- [Collaboration Occurrence](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 7) states:

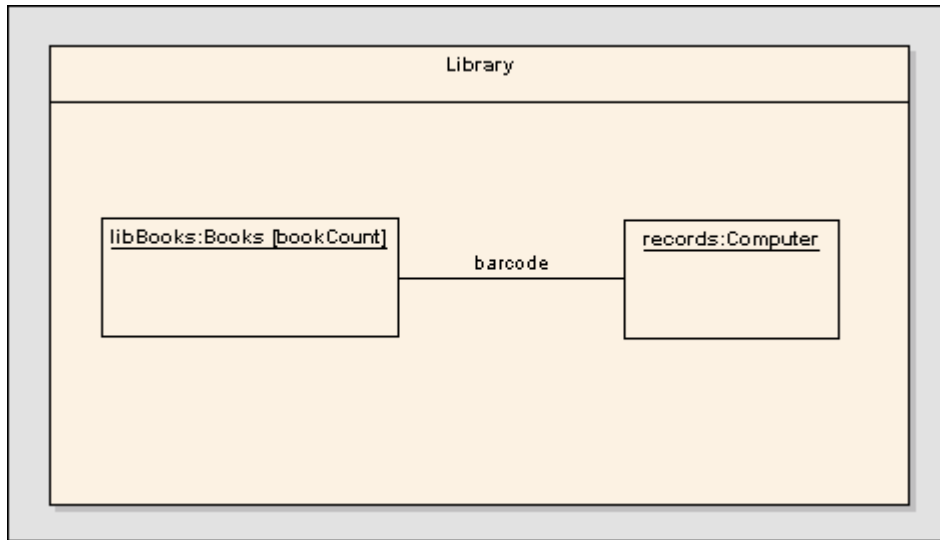
"A diagram that depicts the internal structure of a classifier, including the interaction points of the classifier to other parts of the system. It shows the configuration of parts that jointly perform the behavior of the containing classifier. The architecture diagram specifies a set of instances playing parts (roles), as well as their required relationships given in a particular context."

5.2.2.4.1 Properties

Related Topics

A property is a nested structure within a classifier, which is usually a class or an interface. The contained structure reflects instances and relationships reflected within the containing classifier. Properties can have multiplicity.

To demonstrate properties, consider the following diagram, which demonstrates some properties belonging to the library class.

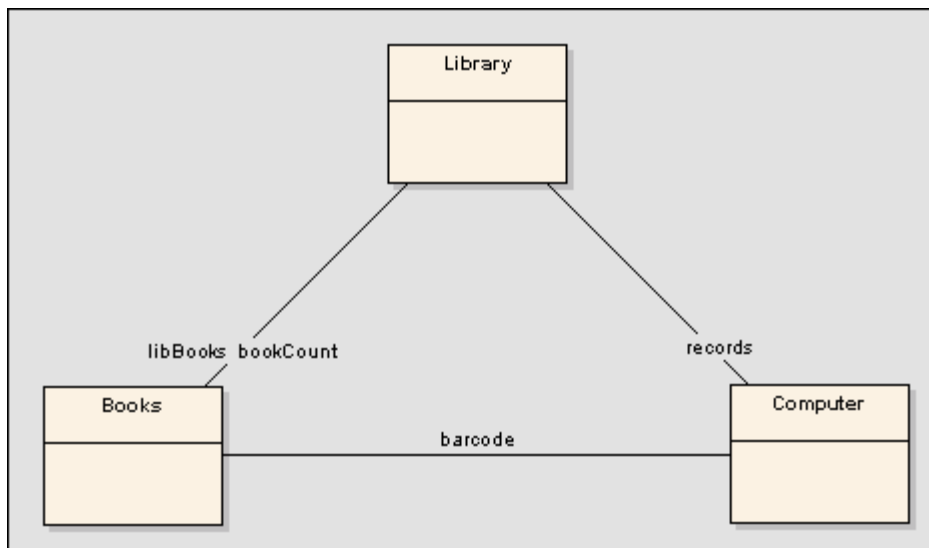


There are two parts, libBooks and records, which are instances corresponding to the classes Books and Computer respectively. After dragging parts from the toolbox out to the workspace, right-click on a part and select *Set Classifier Instance* to link to a classifier.

Note: If parts 'disappear' when dragged onto the class, adjust the Z-order of the class (right-click, select *Z-Order*).

The relationship between the two parts is indicated by the connector, reflecting that communication between the parts is done via the barcode. This contained structure and its parts are properties owned by the library class. To indicate a property that is not owned by composition to the containing classifier, use a box symbol with a dashed outline, indicating association. To do this, right-click the part and select *Advanced Properties*. Set the *IsReference* option to true.

Properties can also be reflected using a normal composite structure (without containing it in a class), with the appropriate connectors, parts and relationships indicated through connections to the class. This alternate representation is shown below. However, this depiction fails to express the ownership immediately reflected by containing properties within a classifier.



Related Topics

- [Composite Structure Diagram](#)

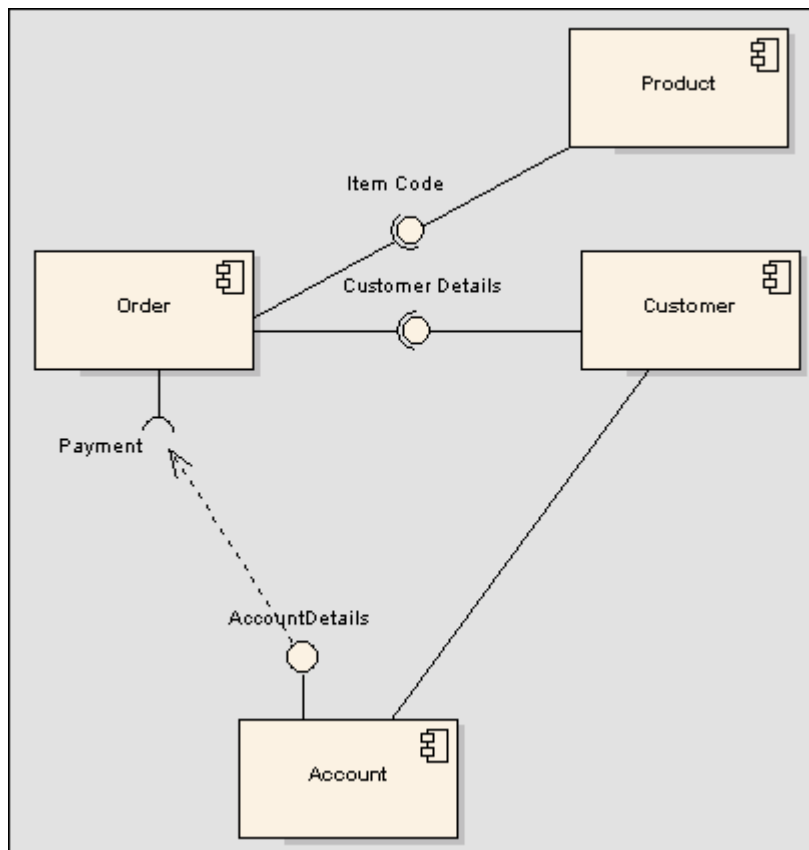
5.2.2.5 Component Diagram

[Example Diagram](#).. [Elements/Connectors](#).. [Related Topics](#).. [OMG UML Specification](#)
















A *Component diagram* illustrates the pieces of software, embedded controllers, etc. that will make up a system. A Component diagram has a higher level of abstraction than a Class diagram - usually a component is implemented by one or more classes (or objects) at runtime. They are building blocks, such that eventually a component can encompass a large portion of a system.

Example Diagram

The diagram below demonstrates some components and their inter-relationships. Assembly connectors 'link' the provided interfaces supplied by Product and Customer to the required interfaces specified by Order. A dependency relationship maps a customer's associated account details to the required interface, 'Payment', indicated by Order.

**Toolbox Elements and Connectors**

Tip: Click the elements and connectors below for more information.

Component Diagram Elements	Component Diagram Connectors
 Package	 Assembly
 Component	 Delegate
 Class	 Associate
 Interface	 Realize
 Object	 Dependency
 Port	 Trace
 Expose Interface	 Generalize
 Artifact	

Related Topics

- [Class Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 7) states:

"A diagram that shows the organizations and dependencies among components."

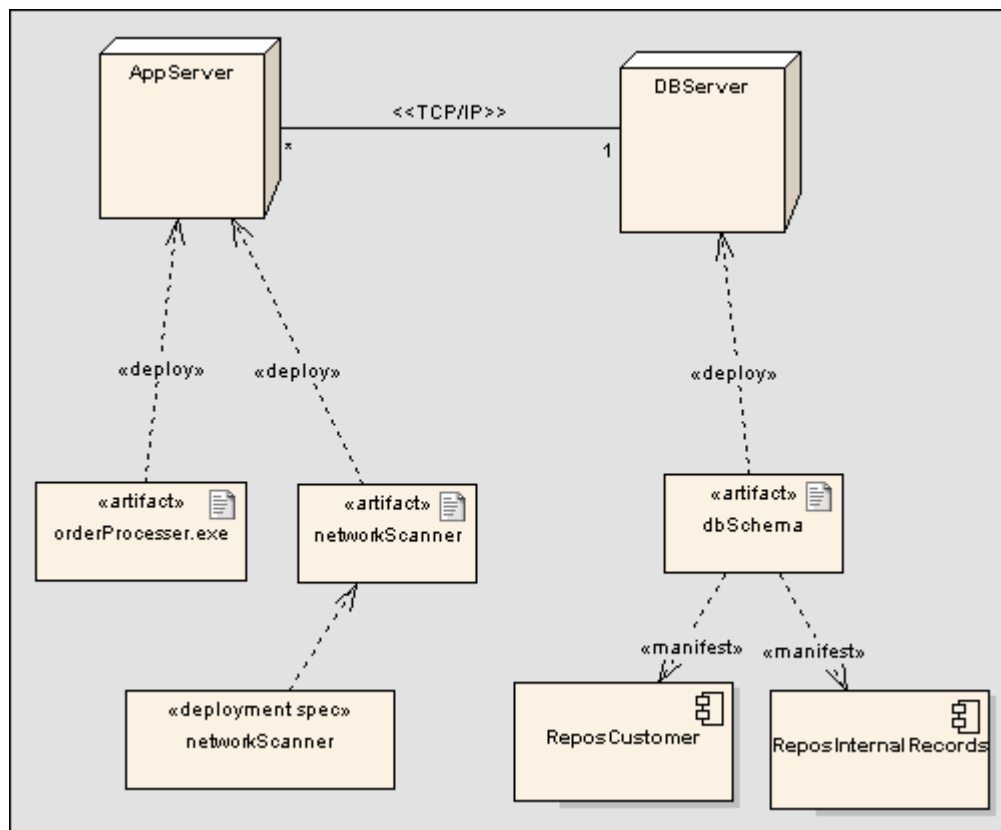
5.2.2.6 Deployment Diagram

[Example Diagram](#)..|..[Elements/Connectors](#)..|..[OMG UML Specification](#)

A *Deployment diagram* shows how and where the system will be deployed. Physical machines and processors are reflected as nodes, and the internal construction can be depicted by embedding nodes or artifacts. As artifacts are allocated to nodes to model the system's deployment, the allocation is guided by the use of deployment specifications.

Example Diagram

Below is an example Deployment diagram. The two nodes have a TCP/IP communication path indicated. Deployment relationships indicate the deployment of artifacts. Furthermore, a deployment spec defines the process of deployment for the networkScanner artifact. The manifestation relationships reveals the physical implementation of components ReposCustomer and ReposInternalRecords.



Toolbox Elements and Connectors

Tip: Click the elements and connectors below for more information.

Deployment Diagram Elements	Deployment Diagram Connectors
Node	Associate
Component	Association Class
Interface	Generalize
Artifact	Realize
Deployment Spec	Deployment
Package	Manifest
	Dependency
	Trace
	Object Flow
	Nesting

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 8) states:

"A diagram that depicts the execution architecture of systems. It represents system artifacts as nodes, which are connected through communication paths to create network systems of arbitrary complexity. Nodes are typically defined in a nested manner, and represent either hardware devices or software execution environments."

5.2.3 Additional Diagrams

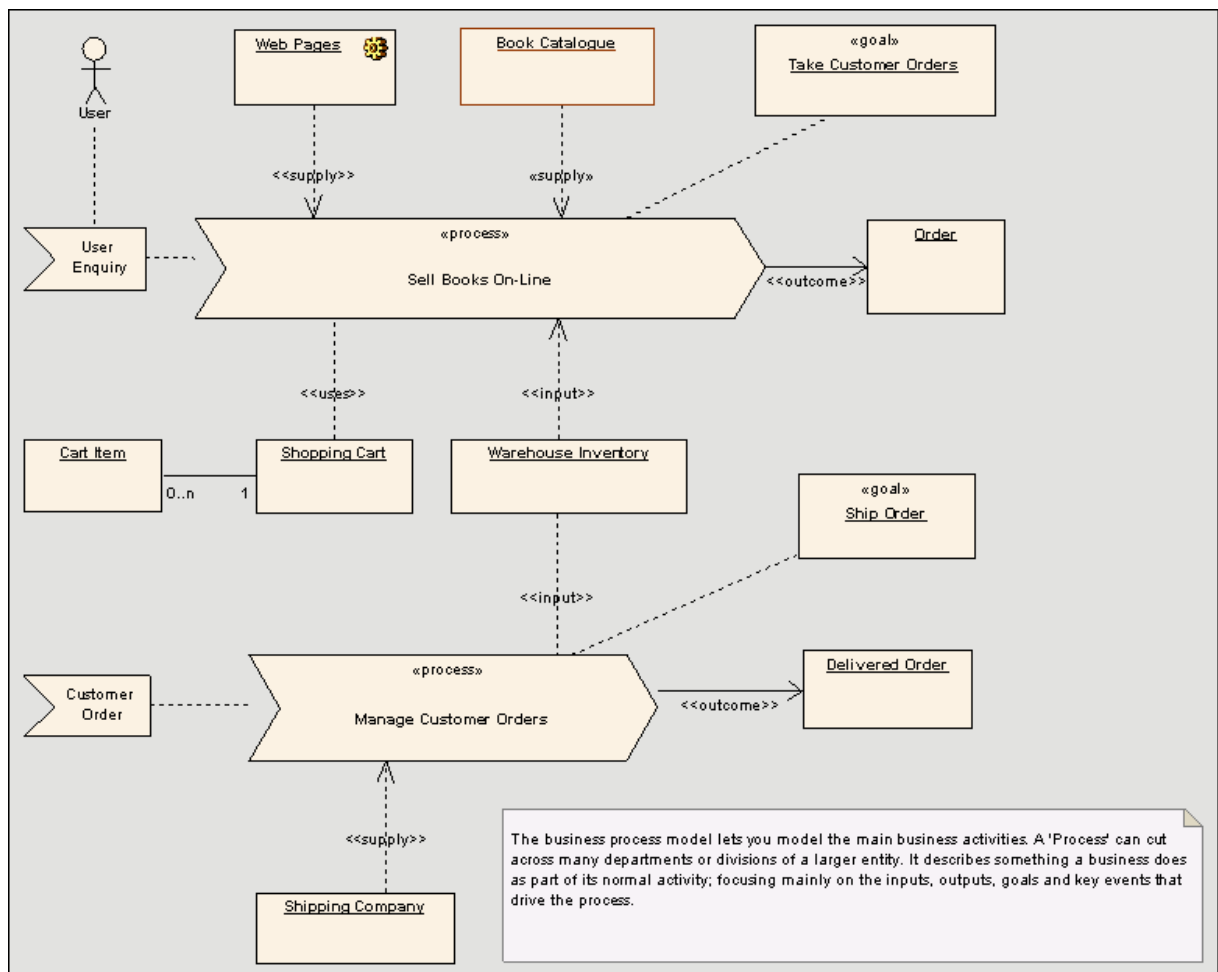
In addition to diagrams defined by the UML, EA provides some additional diagram platforms to model business processes or custom diagrams.

- [Analysis Diagram](#)
- [Custom Diagram](#)
- [Database Schema](#)
- [Robustness Diagram](#)

5.2.3.1 Analysis Diagram

An *Analysis diagram* is used to capture high level business processes and early models of system behavior and elements. It is less formal than some other diagrams, but provides a good means of capturing the essential business characteristics and needs.

EA supports some of the Eriksson-Penker Business Extensions which facilitate business process modeling. The complete Eriksson-Penker Business Extensions UML Profile may also be loaded into EA and used to create detailed process models.



See also: [Business Modeling](#)

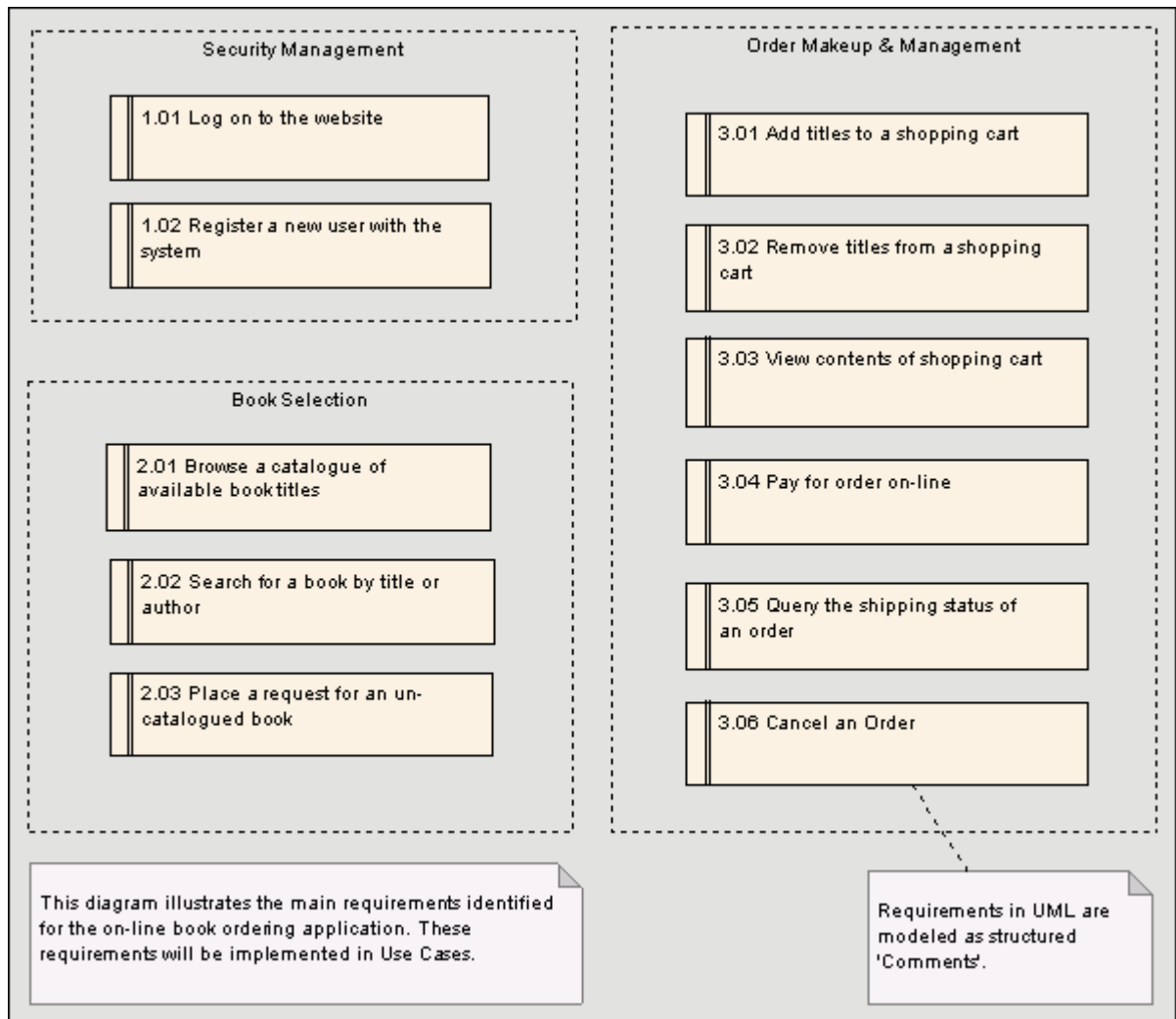
5.2.3.2 Custom Diagram (Extended Class)

A *Custom diagram* can be used to capture requirements, user interfaces or custom-design models.

The below example reflects a requirements diagram. Requirement elements may then be linked back to use cases and components in the system to illustrate how a particular system requirement is met.

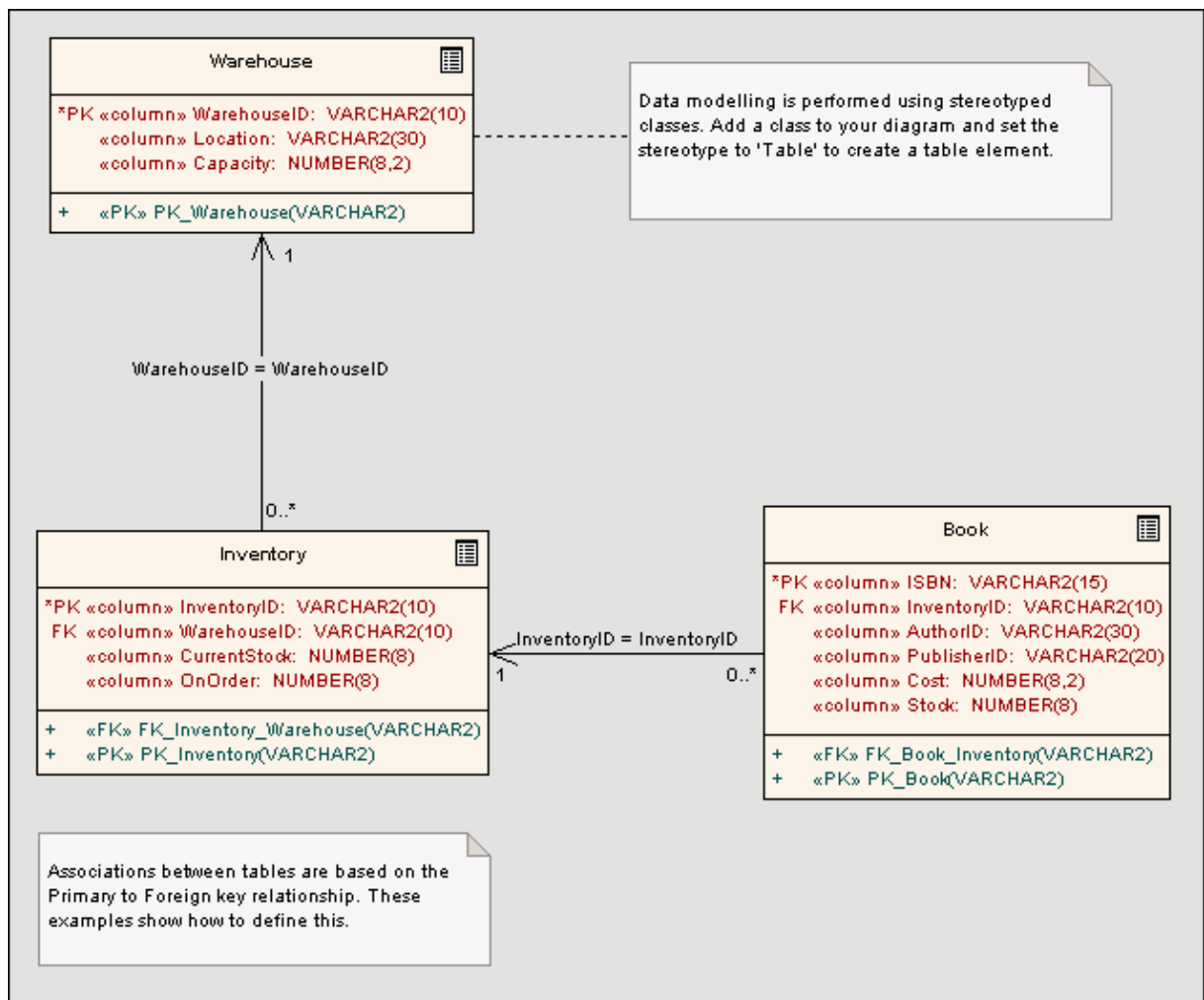
Screen design is supported through a stereotyped screen element and UI Controls. Use this model to design high level system prototypes.

Custom models provide a few extensions to the UML model and allow for some exploratory and non-rigorous experimentation with model elements and diagrams.



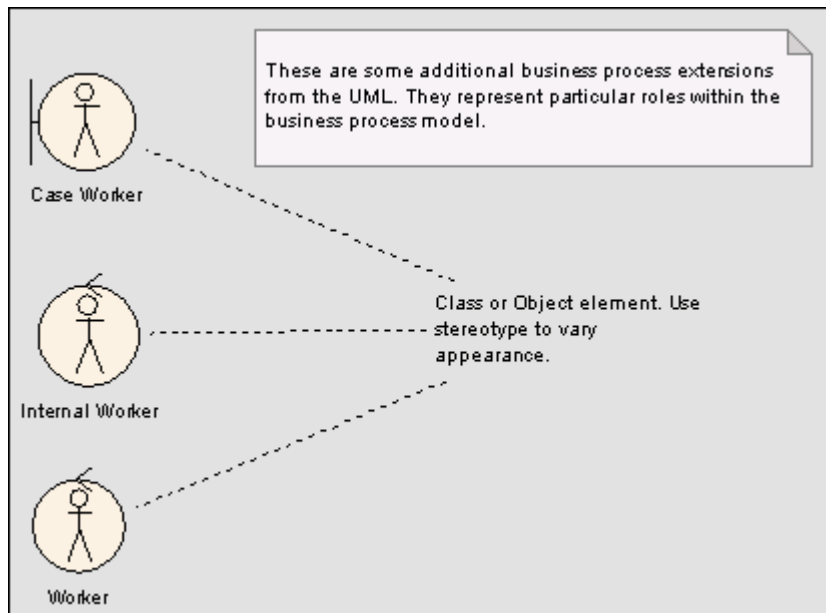
5.2.3.3 Database Schema

Below is an example *Database Schema*:



5.2.3.4 Robustness Diagram

Enterprise Architect supports business process modeling extensions from the UML business process model profile. Examples of these are below:



Robustness diagrams are used in the Iconix Process - you can read more about this at www.sparxsystems.com.au/iconix/iconixsw.htm.

5.3 UML Elements

UML Elements

Models in the UML are constructed from elements. Each element has a different purpose, different rules and different notation. Model elements are used at different stages of the design process for different purposes. Elements include [Classes](#), [Objects](#), [Interfaces](#), [Use Cases](#), [Components](#) and [Nodes](#).

The features of Enterprise Architect cannot be fully utilized without a good understanding of UML:

- During early analysis, use cases, activities, business processes, objects and collaborations are used to capture the problem domain.
- During elaboration, sequence diagrams, objects, classes and state machines are used to refine the system specification.
- Components and nodes are used to model larger parts of the system as well as the physical entities that will be created and deployed into a production environment.

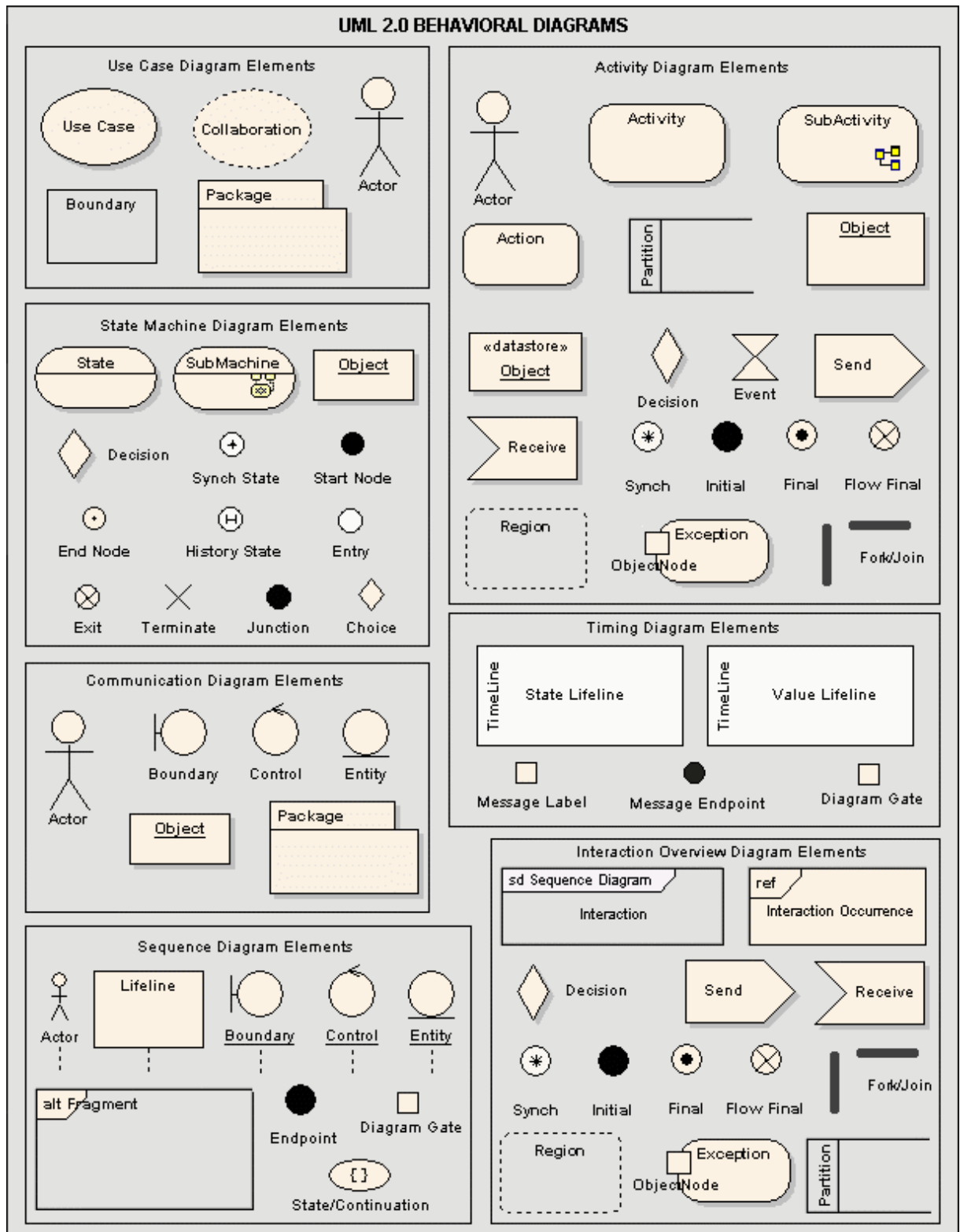
UML Elements can be divided into two categories: those used on [Behavioral Diagrams](#) and those used on [Structural Diagrams](#). This basic set can be extended almost without limit using [Stereotypes](#) and [UML Profiles](#).

5.3.1 Behavioral Diagram Elements

The following figure illustrates the main UML Elements that are used in Behavioral Diagrams. For more insight into using these elements, click on a link:

- **A:** [Action](#), [Activity](#), [Actor](#)
- **B:** [Boundary](#)
- **C:** [Choice](#), [Collaboration](#), [Combined Fragment](#), [Continuation](#)
- **D:** [Datastore](#), [Decision](#), [Diagram Gate](#)

- E: [Endpoint](#), [Entry Point](#), [Exception](#), [Expansion Region](#), [Exit Point](#)
- F: [Final](#), [Flow Final](#), [Fork](#)
- H: [History](#)
- I: [Initial](#), [Interaction Occurrence](#), [Interruptible Activity Region](#)
- J: [Join](#), [Junction](#)
- L: [Lifeline](#)
- O: [Object](#)
- P: [Package](#), [Partition](#)
- R: [Receive](#), [Region](#)
- S: [Send](#), [State](#), [State Lifeline](#), [State/Continuation](#), [SubActivity](#), [SubMachine](#), [Synch](#), [System Boundary](#)
- T: [Terminate](#)
- U: [Use Case](#)
- V: [Value Lifeline](#)

**See also:**

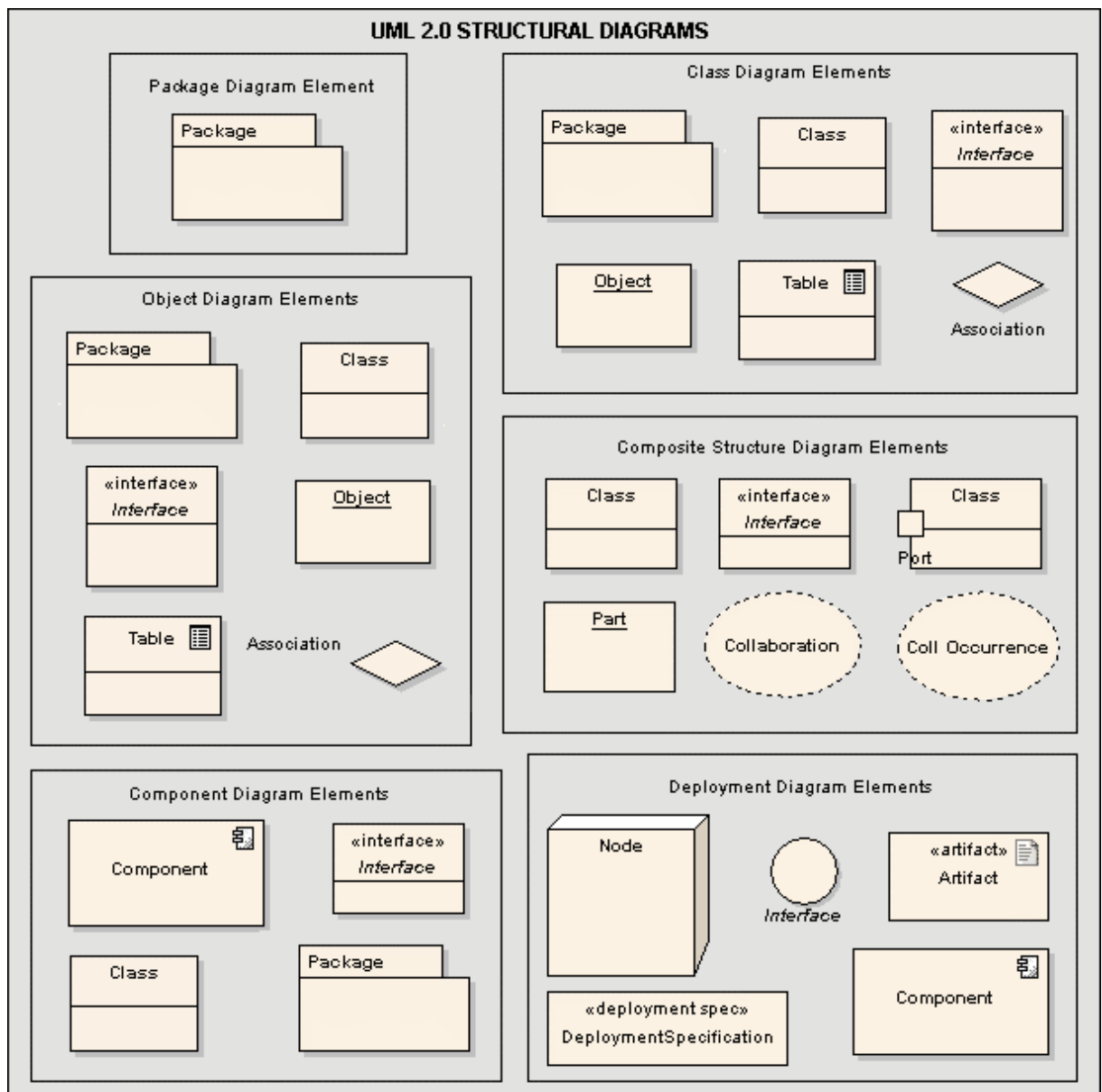
- [Introduction to the UML Language](#)
- [Introduction to UML Elements](#)

- [Structural Diagram Elements](#)

5.3.2 Structural Diagram Elements

The following figure illustrates the main UML Elements that are used in Structural Diagrams. For more insight into using these elements, click on a link:

- **A:** [Artifact](#)
- **C:** [Class](#), [Collaboration](#), [Collaboration Occurrence](#), [Component](#)
- **D:** [Deployment Specification](#)
- **I:** [Interface](#)
- **N:** [Node](#)
- **O:** [Object](#)
- **P:** [Package](#), [Part](#), [Port](#)
- **Q:** [Qualifiers](#)

**See also:**

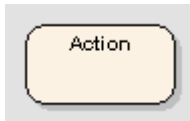
- [Introduction to the UML Language](#)
- [Introduction to UML Elements](#)
- [Behavioral Diagram Elements](#)

5.3.3 Basic Elements

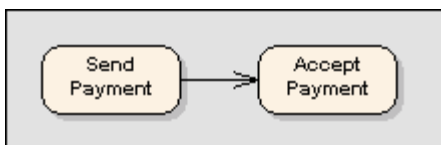
An introduction to elements defined by the UML follows, which together compose the backbone of modeling. The most conceivable modeling elements are stereotypes or extensions of the elements introduced in this section.

5.3.3.1 Action

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

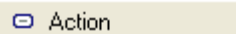


An *action* element describes a basic process or transformation that occurs within a system. It is the basic functional unit within an Activity diagram. Actions can be thought of as children of [activities](#). Both represent processes, but activities can contain multiple steps or decomposable processes, each of which can be embodied in an action. An action cannot be further broken down or decomposed.



Common Usage

- [Activity Diagram](#)



Further Information

- [Action Notation](#)
- [Action Expansion Node](#)
- [Activity Pre and Post Conditions](#)
- [Action Pin](#)
- [Activities](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 280) states:

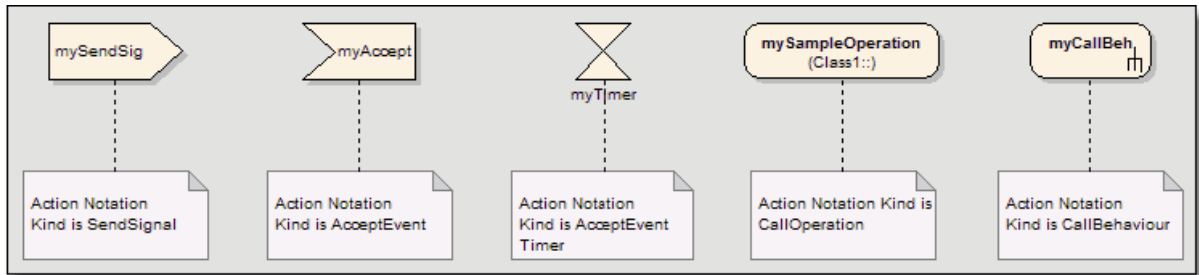
"An action is an executable activity node that is the fundamental unit of executable functionality in an activity, as opposed to control and data flow among actions. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise."

"An action may have sets of incoming and outgoing activity edges that specify control flow and data flow from and to other nodes. An action will not begin execution until all of its input conditions are satisfied. The completion of the execution of an action may enable the execution of a set of successor nodes and actions that take their inputs from the outputs of the action."

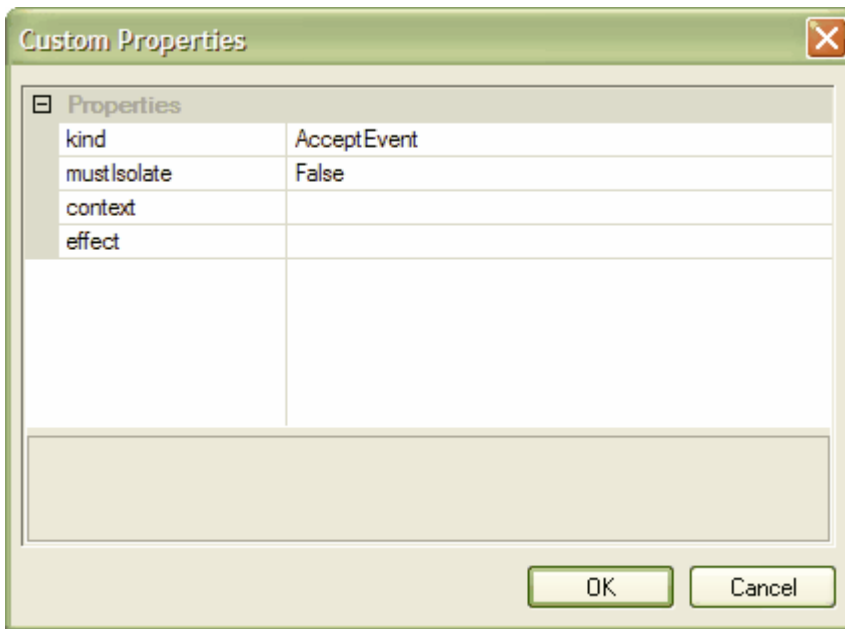
5.3.3.1.1 Action Notation

[Further Information](#)

Some properties can be graphically depicted on an action element, as exemplified below.



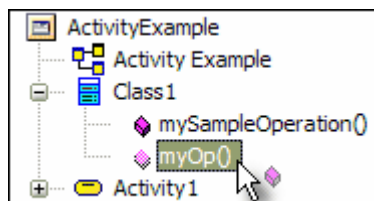
These properties can be defined by right-clicking on the activity and selecting *Advanced Properties*, which opens the following *Custom Properties* dialog box.



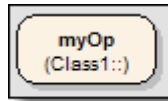
Class Operations in Activity Diagrams

Operations from classes may be displayed on Activity diagrams as an action. When an operation is shown as an action the notation of the action, will display the name of the class which features the operation. To add an operation to an Activity diagram use the following procedure:

1. Open an Activity diagram.
2. From the *Project View* open a class and locate the operation that is to be added to the activity diagram.
3. Drag the operation on to the diagram.



4. When the operation has been added to the Activity diagram the Action will display the namer of the class which features the operation.

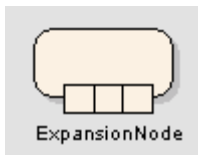


Further Information

- [Action](#)

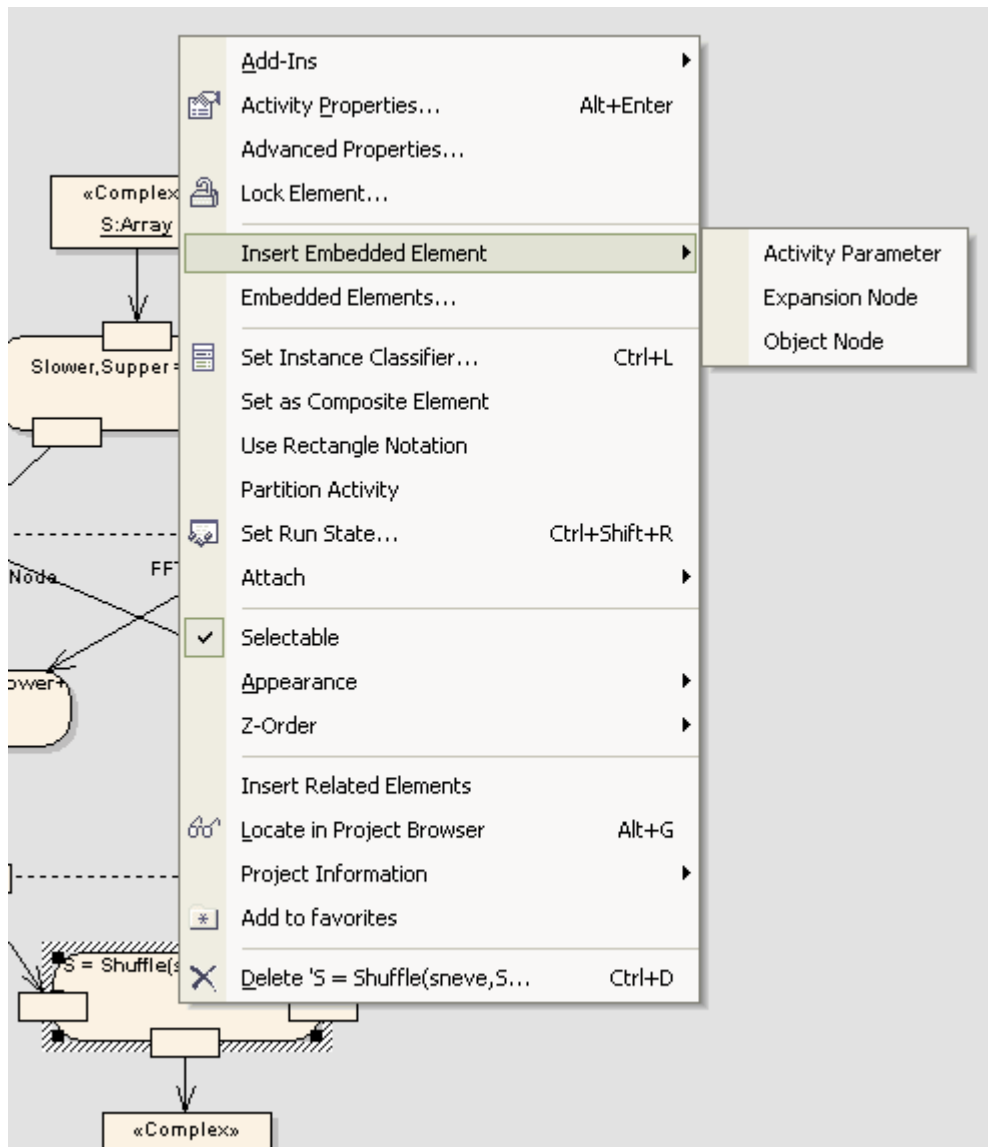
5.3.3.1.2 Action Expansion Node

[Further Information](#)

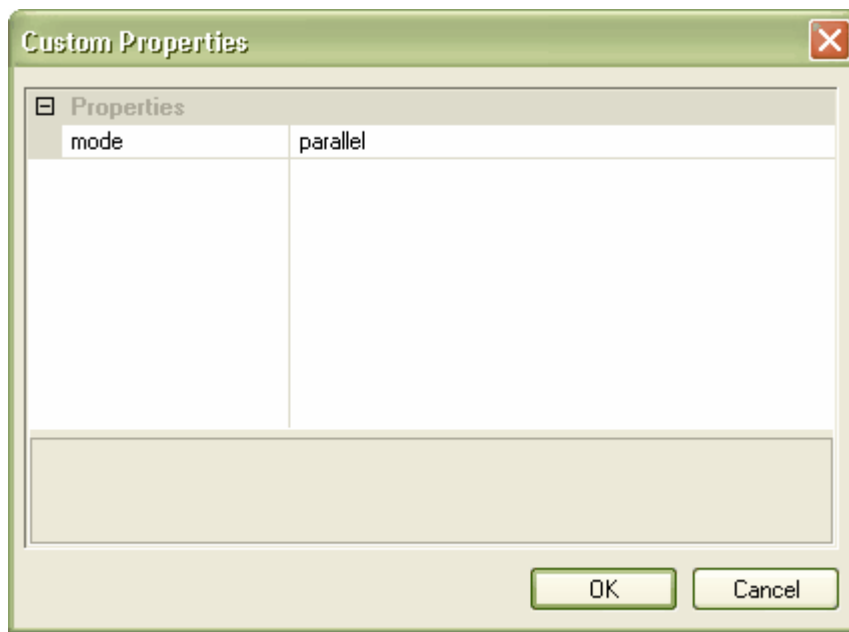


Representing an action as an *expansion node* is a shorthand notation to indicate that action composes an [expansion region](#).

To specify an action is an expansion node, right-click on the action, which will render the dialog below. Navigate to the *Expansion Node* option. After designating an action as an expansion node, modifications or deletions can be made by accessing the *Insert Embedded Elements...* option.



To modify the mode of the expansion region, right-click the action and select *Advanced Properties*.

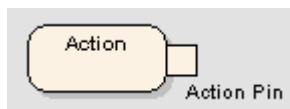


Further Information

- [Expansion Region](#)
- [Action](#)

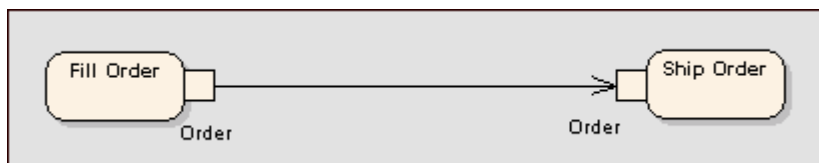
5.3.3.1.3 Action Pin

[Further Information...](#) [OMG UML Specification](#)



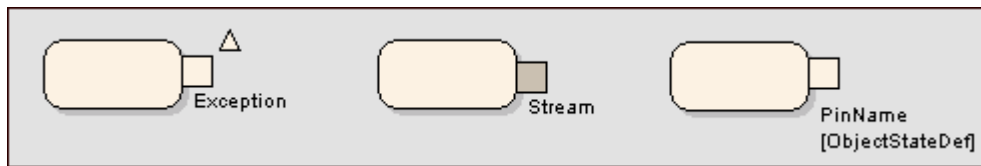
An *action pin* is used to define the data flow out of and into an action. An input pin provides values to the action, whereas an output pin contains the results from that action.

Action pins are used below to connect two actions:

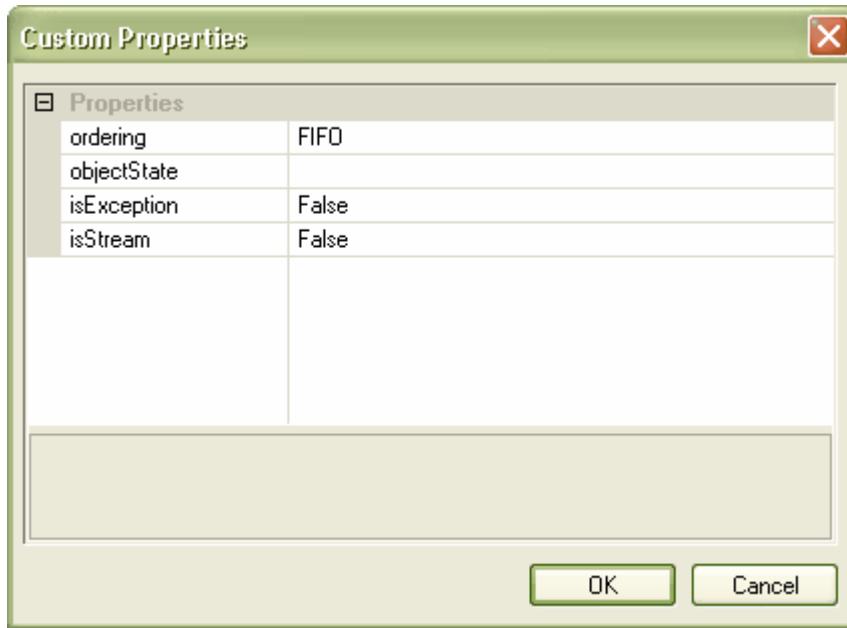


Refer to figure 271 (*UML 2.0 Superstructure*, p. 327).

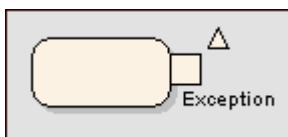
Action pins can be further characterized as defining exception parameters, streams, or states. Associating a state with a pin defines the state of input or output values. For instance, the pin could be called "Orders", but the state could be "Validated" or "Canceled".



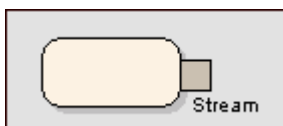
To add an action pin to an action, right click on the action and from the context menu select *Insert Embedded Element | Action Pin*. To change the properties of an action pin right-click on the pin and select *Advanced Properties*. The following properties can be set:



On setting *isException* to *True*, indicating a pin is for an exception parameter, the diagram returns:



On setting *isStream* to *True*, the diagram returns a greyed pin to represent an input or output pin style for streaming.



On defining an *ObjectState*, the Object State definition name is represented in square brackets as follows:



Further Information

- [Action](#)

OMG UML Specification

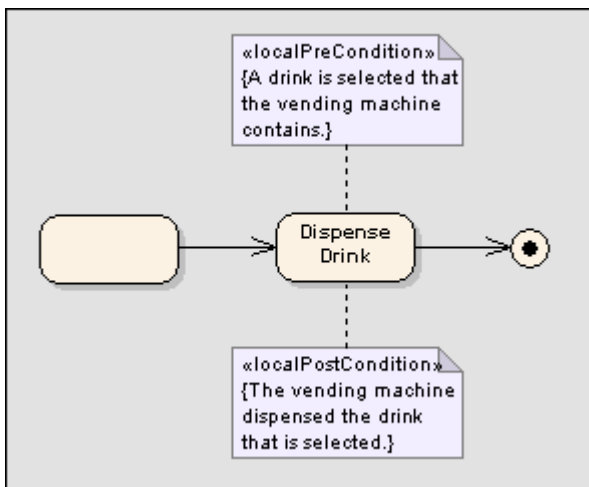
The OMG UML specification (*UML 2.0 Superstructure*, p. 13) states:

"A model element that represents the data values passed into a behavior upon its invocation as well as the data values returned from a behavior upon completion of its execution."

5.3.3.1.4 Local Pre/Post Conditions

[Further Information](#)

Actions can be further defined with *pre-condition* and *post-condition* notes, which constrain an action's entry and exit. These notes can be added to an action as defined below.

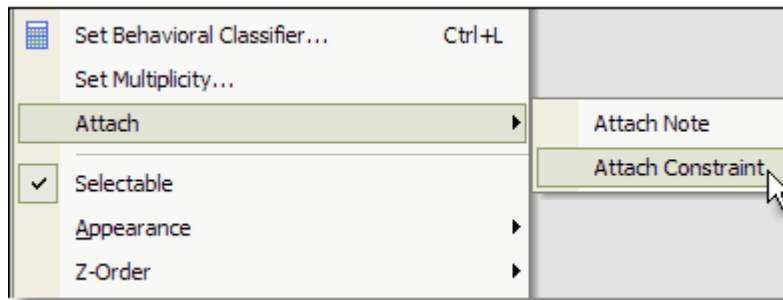


Refer to figure 200 (*UML 2.0 Superstructure*, p. 283).

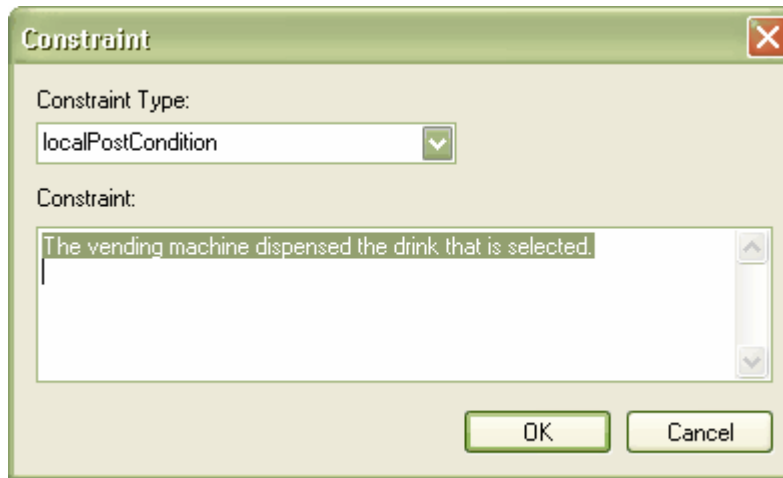
Creating a Constraint

To attach a constraint to an action use the following steps:

1. Right-click on the action. This will give the following:



2. Select *Attach*. From the sub-menu select *Attach Constraint*.
3. Right-click on the newly created Note.
4. Select *Constraint Properties*.



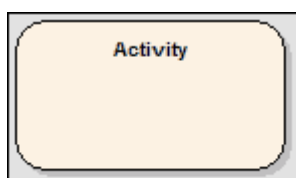
5. Select the *Constraint Type*.
6. Add the text for the constraint.
7. Select *OK* to save it.

Further Information

- [Action](#)

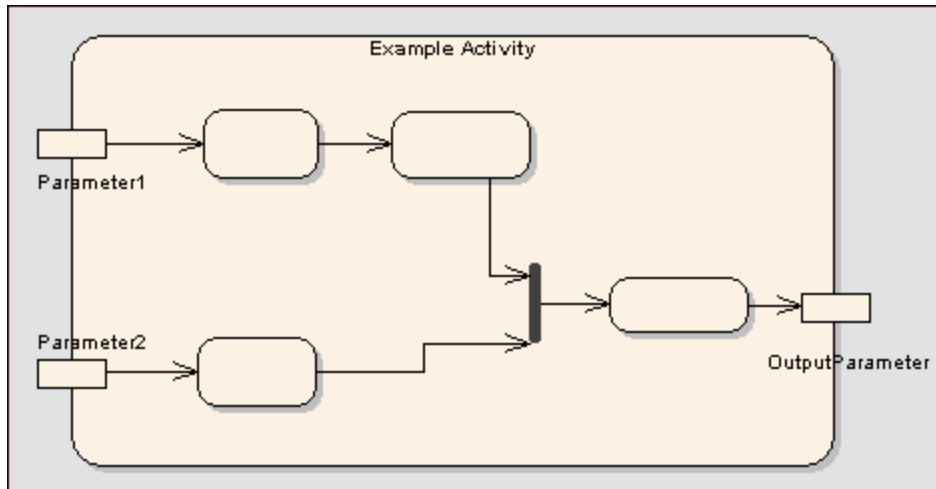
5.3.3.2 Activity

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



An *activity* organizes and specifies the participation of subordinate behaviors, such as sub-activities or actions, to reflect the control and data flow of a process. Activities are used for various modeling purposes, from procedural-type application development for system design, to business process modeling of organizational structures or workflow.

Below is a simple diagram of an activity containing action elements and including input parameters and output parameters.



Common Usage

- [Activity Diagrams](#)

 Activity

Further Information

- [Activity Pre and Post Conditions](#)
- [Action Pin](#)
- [Activities](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 284) states:

"An activity specifies the coordination of executions of subordinate behaviors, using a control and data flow model. The subordinate behaviors coordinated by these models may be initiated because other behaviors in the model finish executing, because objects and data become available, or because events occur external to the flow. The flow of execution is modeled as activity nodes connected by activity edges. A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents. Activity nodes also include flow-of-control constructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to operations that are directly invoked."

"Activities may describe procedural computation. In this context, they are the methods corresponding to operations on classes. Activities may be applied to organizational modeling for business process engineering and workflow. In this context, events often originate from inside the system, such as the finishing of a task, but also from outside the system, such as a customer call. Activities can also be used for information system modeling to specify system level processes."

"Activities may contain actions of various kinds:

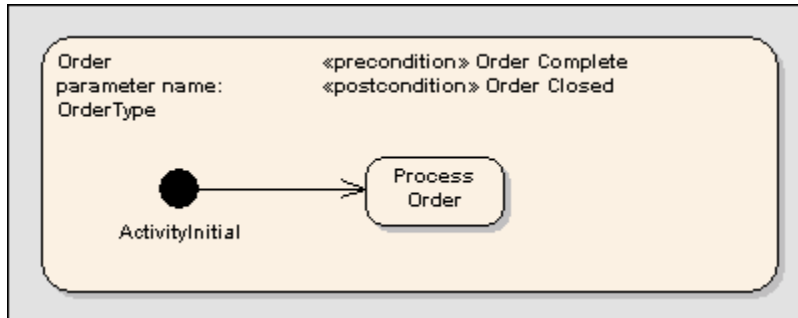
- *occurrences of primitive functions, such as arithmetic functions.*
- *invocations of behavior, such as activities.*
- *communication actions, such as sending of signals.*
- *manipulations of objects, such as reading or writing attributes or associations.*

"Actions have no further decomposition in the activity containing them. However, the execution of a single action may induce the execution of many other actions. For example, a call action invokes an operation which is implemented by an activity containing actions that execute before the call action completes."

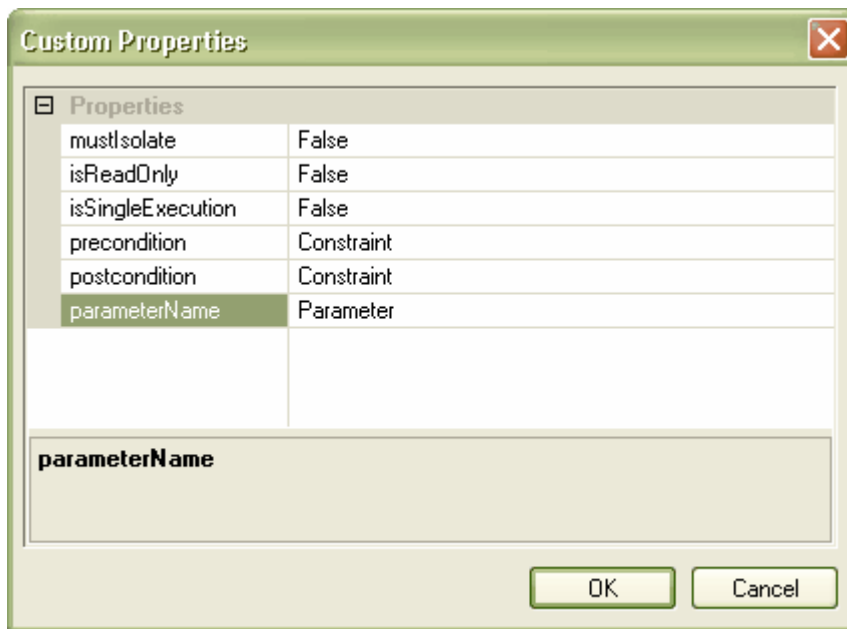
5.3.3.2.1 Activity Notation

[Further Information](#)

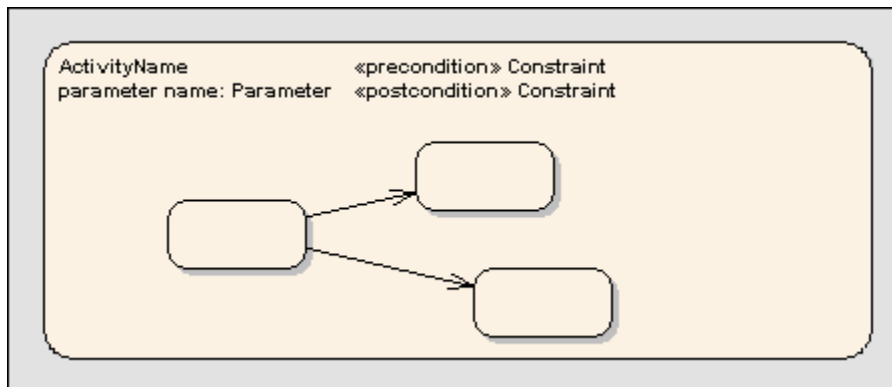
Some properties can be graphically depicted on an activity element, as exemplified below.



These properties can be defined by right-clicking on the activity and selecting *Advanced Properties*, which opens the following *Custom Properties* dialog box.



The resulting activity, whose properties were specified as shown above, appears below:



Further Information

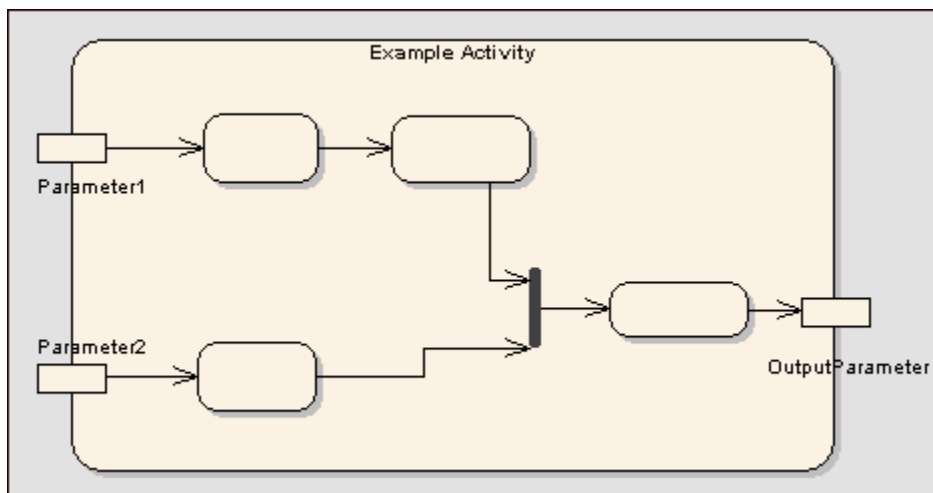
- [Activity](#)

5.3.3.2.2 Activity Parameter Nodes

[Further Information...](#) [OMG UML Specification](#)

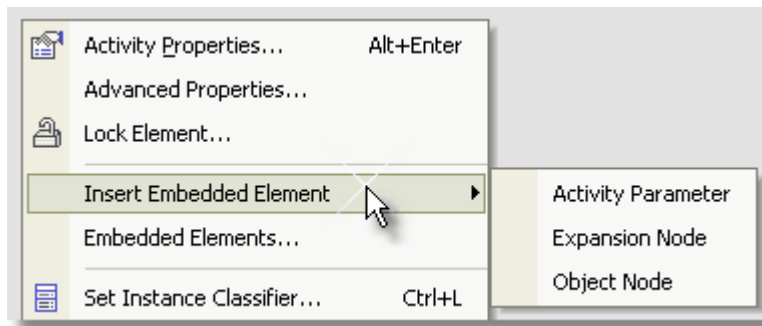
An activity parameter node is used for accepting input to an activity and providing output from an activity.

The following example depicts two entry parameters and one output parameter defined for the activity.



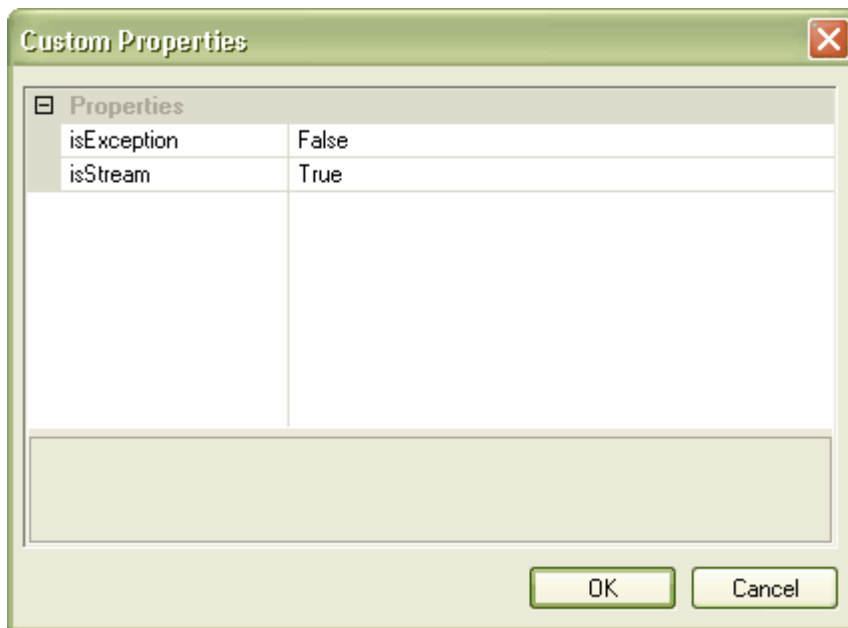
An activity parameter node can be defined for an activity by following these steps:

1. Right-click on the element, selecting *Insert Embedded Element* and *Activity Parameter*.



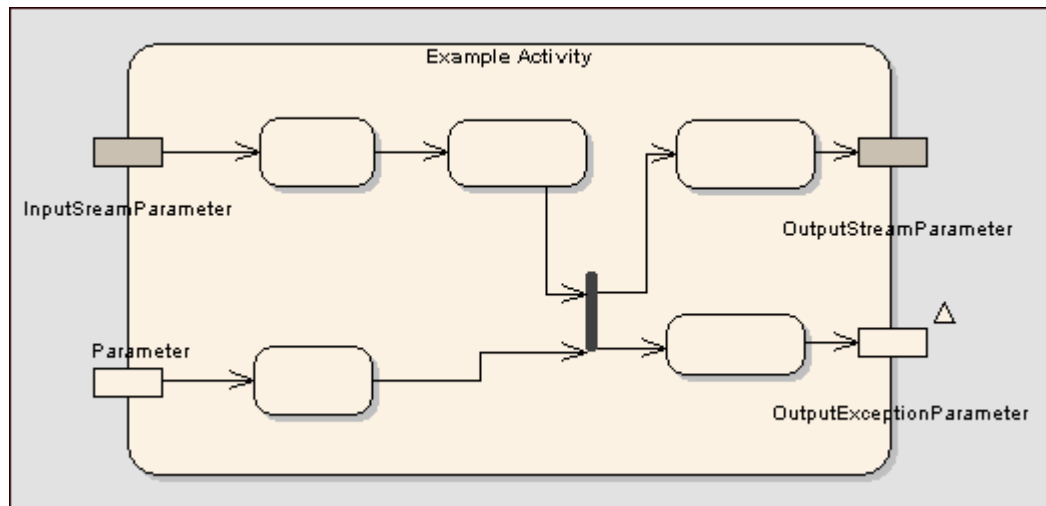
2. You will be prompted with the *Properties* dialog, which asks for the Name, etc. of the embedded element.

After closing this dialog, you can further define the new activity parameter. Right-click on the activity parameter, and select *Advanced Properties*, which opens the following dialog:



Similar to characterizing action pins, activity parameters also have the *isException* and *isStream* options. *isException* indicates that a parameter can emit a value at the exclusion of other outputs, usually because of some error. *isStream* indicates whether a parameter can accept or post values during the execution of the activity.

The following is an example using the above settings:



Further Information

- [Activity](#)
- [Action Pin](#)

OMG UML Specification

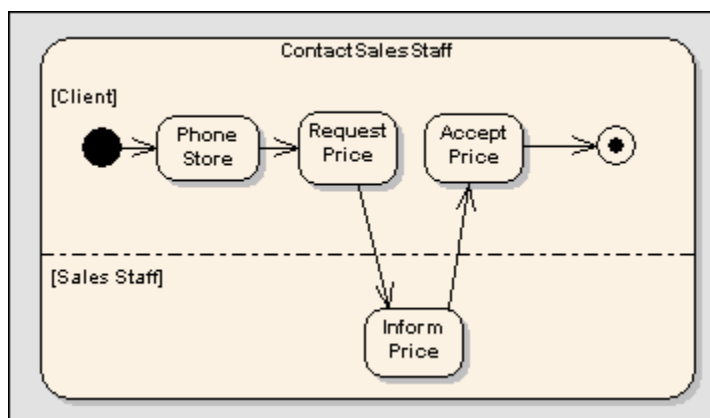
The OMG UML specification (*UML 2.0 Superstructure*, p. 304) states:

"An activity parameter node is an object node for inputs and outputs to activities....Activity parameters are object nodes at the beginning and end of flows, to accept inputs to an activity and provide outputs from it. (CompleteActivities) Activity parameters inherit support for streaming and exceptions from Parameter."

5.3.3.2.3 Activity Partition

Further Information

Activity partitions are used to logically organize an activity. They do not affect the token flow of an activity diagram, but help structure the view or parts of an activity. An example of a partitioned activity follows:



To define partitions:

1. Right-click on the activity element.

2. Select *Partition Activity*. The *Activity Partitions* window will appear, as shown below.



3. Enter the name of a partition and click *Save*.
4. Repeat Step 3 for each partition to be created.

Further Information

- [Activity](#)

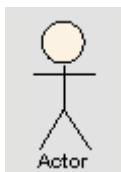
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 307) states:

"Partitions divide the nodes and edges to constrain and show a view of the contained nodes. Partitions can share contents. They often correspond to organizational units in a business model. They may be used to allocate characteristics or resources among the nodes of an activity."

5.3.3.3 Actor

[Common Usage](#)...[OMG UML Specification](#)



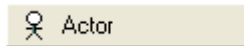
An *actor* is a user of the system; "user" can mean a human user, a machine, or even another system. Anything that interacts with the system from the outside or system boundary is termed an actor. Actors are typically

associated with [Use Cases](#).

Actors may use the system through a graphical user interface, through a batch interface or through some other media. An actor's interaction with a use case is documented in a use case scenario and details the functions a system must provide to satisfy the user requirements.

Note: Remember, an actor can be a human user -OR- a machine -OR- another system -OR- another subsystem in the model.

Common Usage



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

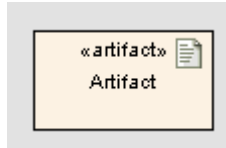
OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 5) states:

"A construct that is employed in use cases that define a role that a user or any other system plays when interacting with the system under consideration. It is a type of entity that interacts, but which is itself external to the subject. Actors may represent human users, external hardware, or other subjects. An actor does not necessarily represent a specific physical entity. For instance, a single physical entity may play the role of several different actors and, conversely, a given actor may be played by multiple physical entities."

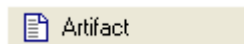
5.3.3.4 Artifact

[Common Usage](#) | [OMG UML Specification](#)



An *artifact* is any physical piece of information used or produced by a system. Artifacts can have associated properties or operations, and can be instantiated or associated with other artifacts. Examples of artifacts include model files, source files, database tables, development deliverables or support documents.

Common Usage



- [Deployment Diagram](#)

OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 185) states:

"An Artifact defined by the user represents a concrete element in the physical world. A particular instance (or 'copy') of an artifact is deployed to a node instance. Artifacts may have composition associations to other artifacts that are nested within it. For instance, a deployment descriptor artifact for a component may be contained within the artifact that implements that component. In that way, the component and its descriptor are deployed to a node instance as one artifact instance."

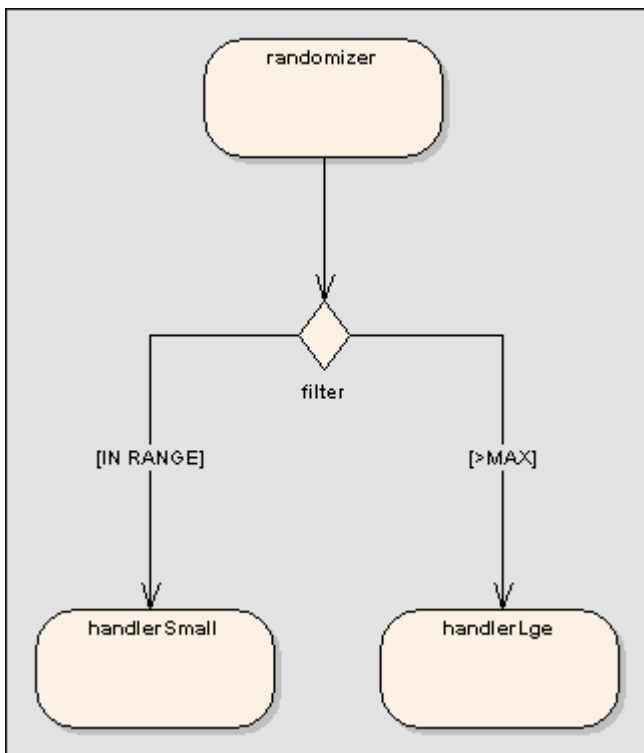
5.3.3.5 Choice

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



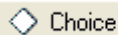
The *choice* pseudo-state is used to compose complex transitional paths, where the outgoing transition path is decided by dynamic, run-time conditions. The run-time conditions are determined by the actions performed by the state machine on the path leading to the choice.

The following example depicts the choice element. Upon reaching the filter pseudo-state, a transition will fire to the appropriate state based on the run-time value passed to the filter. Very similar in form to a junction pseudo-state, the choice pseudo-state's distinction is deciding transition paths at run-time.



Common Usage

- [State Machine Diagram](#)



Further Information

- [Pseudo-States](#)

OMG UML Specification

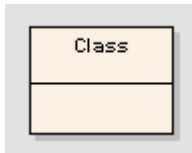
The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"choice vertices which, when reached, result in the dynamic evaluation of the guards of the triggers of its outgoing transitions. This realizes a dynamic conditional branch. It allows splitting of transitions into multiple outgoing paths such that the decision on which path to take may be a function of the results of prior actions performed in the same run-to-completion step. If more than one of the guards evaluates to true, an arbitrary one is selected. If none of the guards evaluates to true, then the model is considered ill-formed. (To avoid this, it is recommended to define one outgoing transition with the predefined "else")

guard for every choice vertex.) Choice vertices should be distinguished from static branch points that are based on junction points."

5.3.3.6 Class

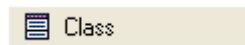
[Common Usage](#).. [Further Information](#).. [OMG UML Specification](#)



A *class* is a representation of object(s), that reflects their structure and behavior within the system. It is a template from which actual running instances are created. A class may have attributes (data) and methods (operations or behavior). Classes may inherit characteristics from parent classes and delegate behavior to other classes. Class models usually describe the logical structure of the system and are the building blocks from which components are built.

The top section of the class below shows the attributes (or data elements) associated with a class. These hold the 'state' of an object at run-time. If the information is saved to a data store and can be reloaded, it is termed 'persistent'. The lower section contains the class operations (or methods at run-time). Operations describe the behavior a class offers to other classes, and the internal behavior it has (private methods).

Common Usage



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

Further Information

- [Parameterized Classes \(Templates\)](#)
- [Active Classes](#)
- [Attributes](#)
- [Operations](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 87) states:

"The purpose of a class is to specify a classification of objects and to specify the features that characterize the structure and behavior of those objects.

"Objects of a class must contain values for each attribute that is a member of that class, in accordance with the characteristics of the attribute, for example its type and multiplicity.

"When an object is instantiated in a class, for every attribute of the class that has a specified default, if an initial value of the attribute is not specified explicitly for the instantiation, then the default value specification is evaluated to set the initial value of the attribute for the object.

"Operations of a class can be invoked on an object, given a particular set of substitutions for the parameters of the operation. An operation invocation may cause changes to the values of the attributes of that object. It may also return a value as a result, where a result type for the operation has been defined. Operation invocations may also cause changes in value to the attributes of other objects that can be navigated to, directly or indirectly, from the object on which the operation is invoked, to its output parameters, to objects navigable from its parameters, or to other objects in the scope of the operation's execution. Operation invocations may also cause the creation and deletion of objects."

5.3.3.6.1 Active Classes

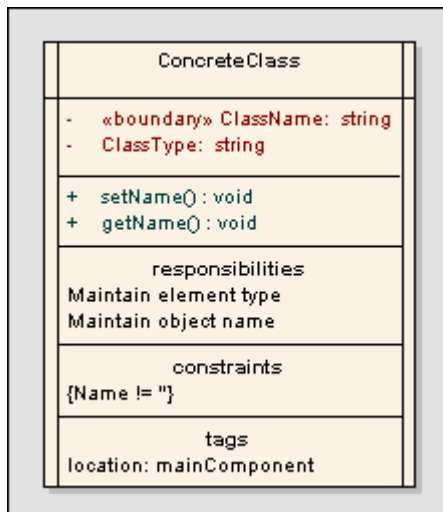
[.Further Information..](#) | [..OMG UML Specification](#)

An *active class* indicates that, when instantiated, it will control its own execution. Rather than being invoked or activated by other objects, it can operate standalone, and define its own thread of behavior.

To define an active class in EA:

1. Highlight a class, and open its *Class Property* dialog.
2. Press the *Advanced* button.
3. Check the *IsActive* check box.
4. Press *OK* to save the details.

A sample active class is displayed in the example below:



Further Information

- [Class](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 4) states:

"An object that may execute its own behavior without requiring method invocation. This is sometimes referred to as "the object having its own thread of control." The points at which an active object responds to communications from other objects are determined solely by the behavior of the active object and not by the invoking object. This implies that an active object is both autonomous and interactive to some degree."

5.3.3.6.2 Parameterized Classes (Templates)

[.Further Information..](#) | [..OMG UML Specification](#)

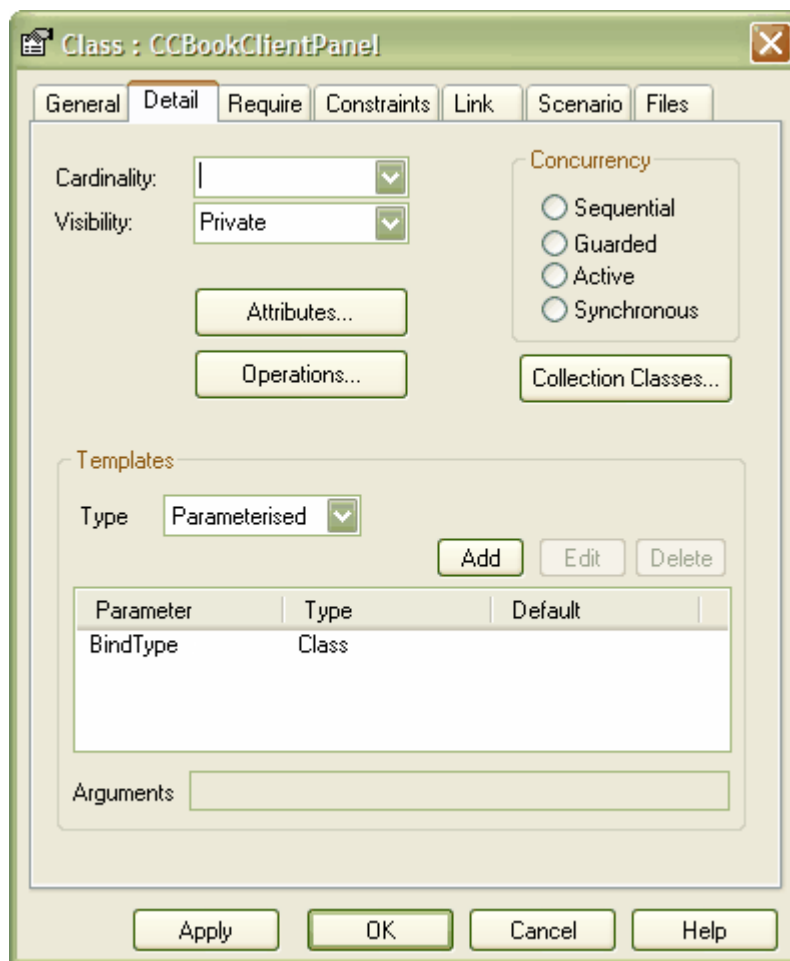
Enterprise Architect supports *template* or *parameterized classes*, which specify parameters that must be defined by any binding class. A template class allows for its functionality to be reused by any bound class. If a

default value is specified for a parameter, and a binding class doesn't provide a value for that parameter, the default is used. Parameterized classes are commonly implemented in C++.

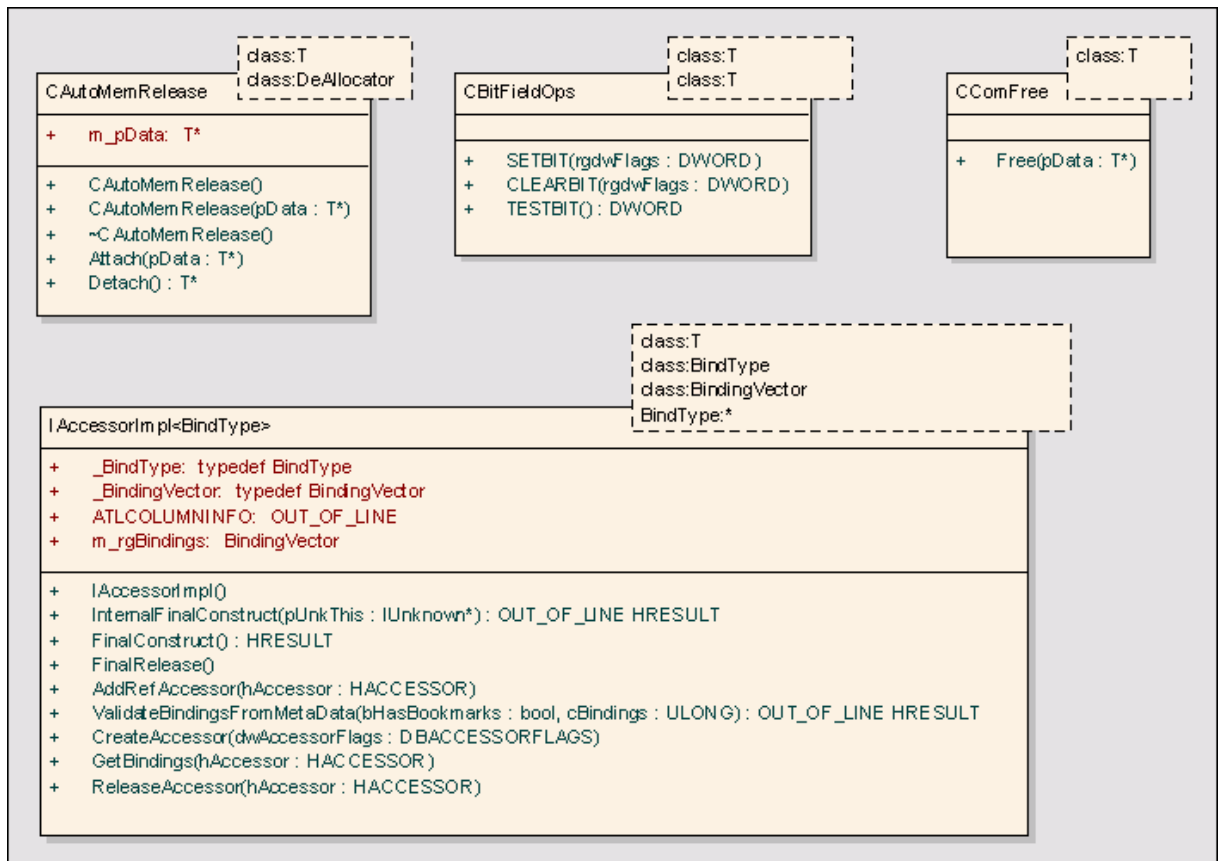
EA will import and generate templated classes for C++. Template classes are shown with the parameters in a dashed outline box in the upper right corner of a class.

To create a parameterized class:

1. Open the *Class Property* dialog for a class.
2. Select the *Detail* tab.
3. Under "Templates", select *Type* to be "Parameterized".
4. Define your parameters in the provided list dialog.



Notation example:



Further Information

- [Class](#)

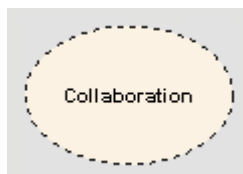
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 545) states:

"A template is a parameterized element that can be used to generate other model elements using TemplateBinding relationships. The template parameters for the template signature specify the formal parameters that will be substituted by actual parameters (or the default) in a binding."

5.3.3.7 Collaboration

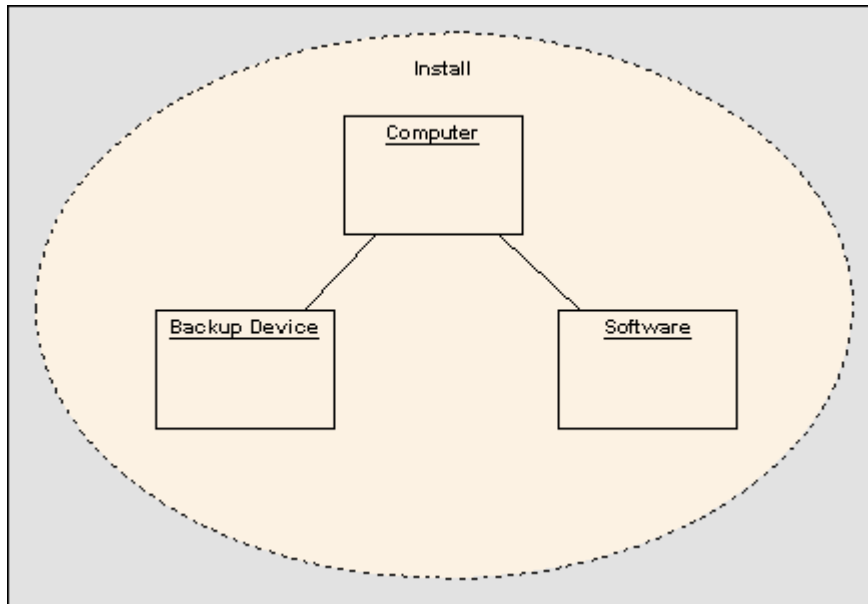
[Common Usage](#).. [Further Information](#).. [OMG UML Specification](#)



A *collaboration* defines a set of co-operating roles and their connectors. These are used to collectively illustrate a specific functionality. A collaboration should specify only the roles and attributes needed to accomplish a specific task or function. Although in practice a behavior and its roles could involve many

tangential attributes and properties, isolating the primary roles and their requisites simplifies and clarifies the behavior, as well as providing for reuse. A collaboration often implements a pattern to apply to various situations.


The following example illustrates an "Install" collaboration, with three roles connected as shown. The process for this collaboration can be demonstrated by attaching an interaction diagram.



To understand referencing a collaboration in a specific situation, see the topic [Collaboration Occurrence](#).

Common Usage

- [Composite Structure Diagrams](#)

 Collaboration

Further Information

- [Collaboration Occurrence](#)

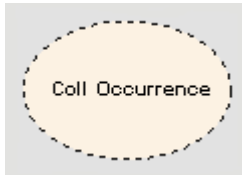
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 6) states:

"The specification of how an operation or classifier, such as a use case, is realized by a set of classifiers and associations playing specific roles used in a specific way. The collaboration defines an interaction."

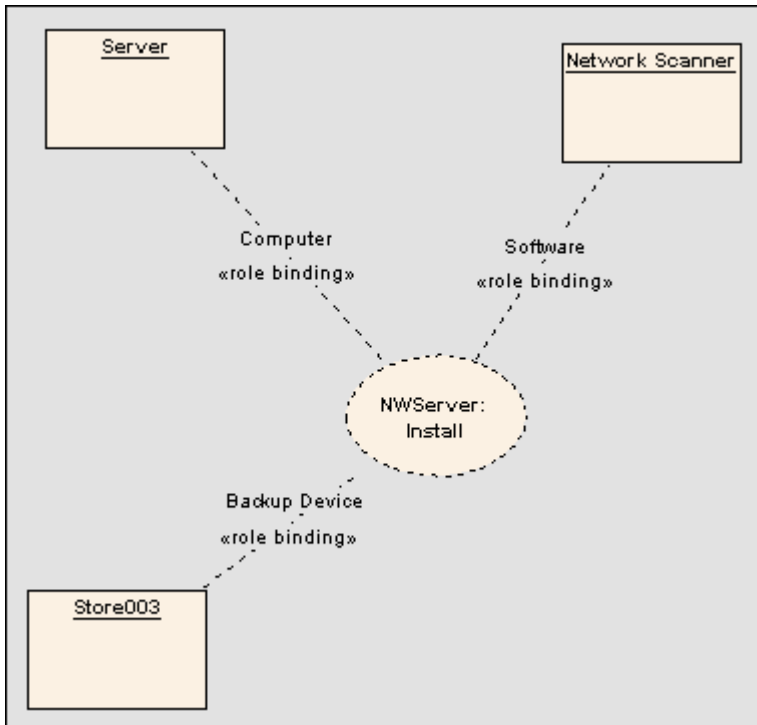
5.3.3.8 Collaboration Occurrence

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



Use a *collaboration occurrence* to apply a pattern defined by a collaboration to a specific situation.

The following example uses an occurrence, 'NWServer', of the collaboration 'Install', to define the installation process of a network scanner. This process can be defined by an interaction attached to the collaboration. Refer to the [Collaboration](#) topic to see the 'Install' collaboration, depicted below.



Common Usage

- [Composite Structure Diagrams](#)

Further Information

- [Collaboration](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 6) states:

"A particular use of a collaboration to explain the relationships between the parts of a classifier or the properties of an operation. It may also be used to indicate how a collaboration represents a classifier or an operation. A collaboration occurrence indicates a set of roles and connectors that cooperate within the classifier or operation according to a given collaboration, indicated by the type of the collaboration occurrence. There may be multiple occurrences of a given collaboration within a classifier or operation,

each involving a different set of roles and connectors. A given role or connector may be involved in multiple occurrences of the same or different collaborations."

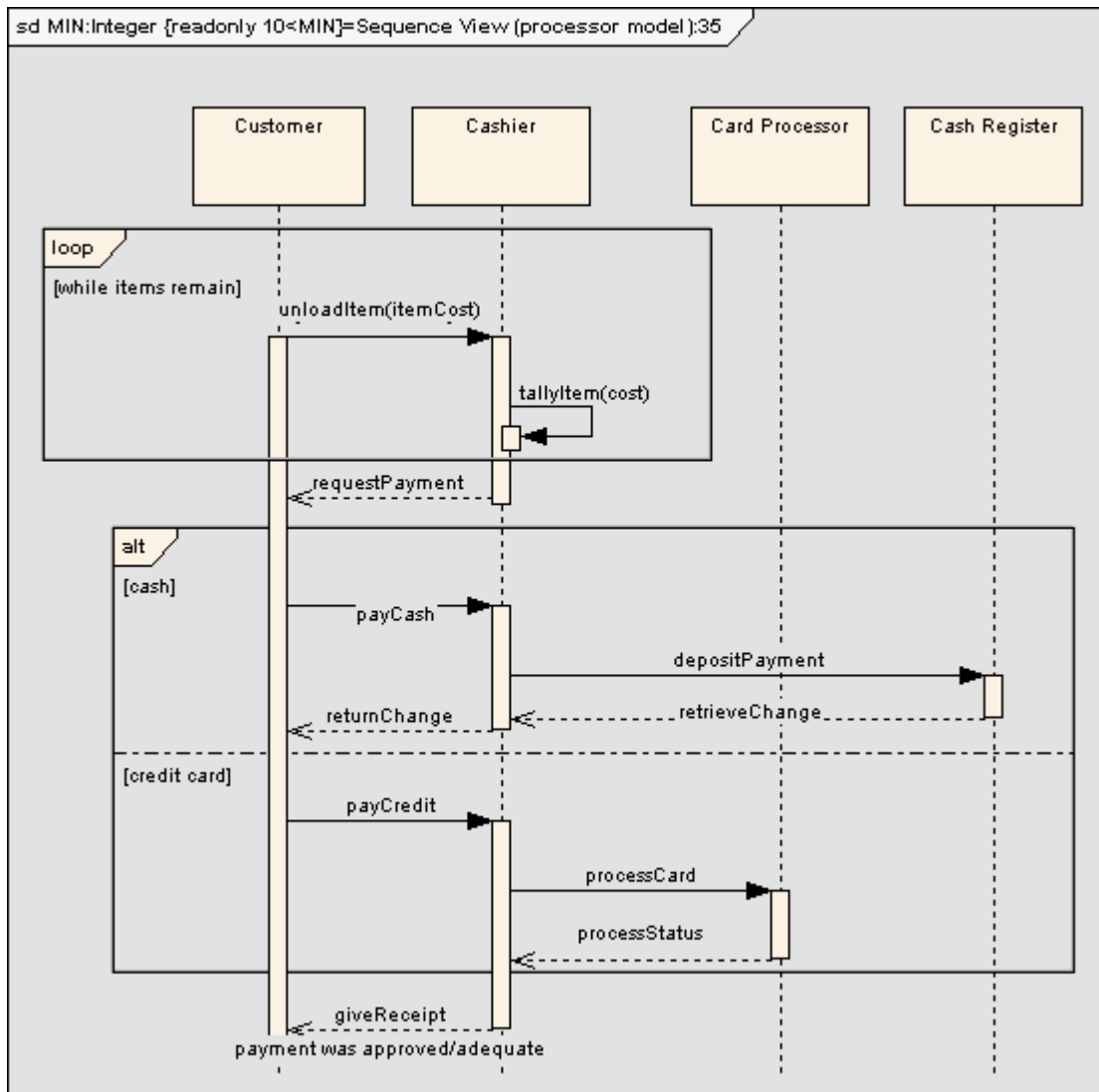
5.3.3.9 Combined Fragment

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



A *combined fragment* reflects a piece or pieces of interaction (called interaction operands) controlled by an interaction operator, whose corresponding boolean conditions are known as interaction constraints. It appears graphically as a transparent window, divided by horizontal dashed lines for each operand.

The following diagram exemplifies the use of combined fragments, with a sequence diagram modeling a simplified purchasing process. A loop fragment is utilized to iterate through an unknown amount of items for purchase, after which the cashier requests payment. At this point, two payment options are considered, and an alternative fragment is created, divided to show the two operands, cash and credit card. After the fragment completes its trace, the cashier gives a receipt to the customer, under the fulfilled condition that payment requirements were met.

**Common Usage**

- [Sequence Diagram](#)

 Fragment(s)
Further Information

- [Create a Combined Fragment](#)
- [Interaction Operators](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 409) states:

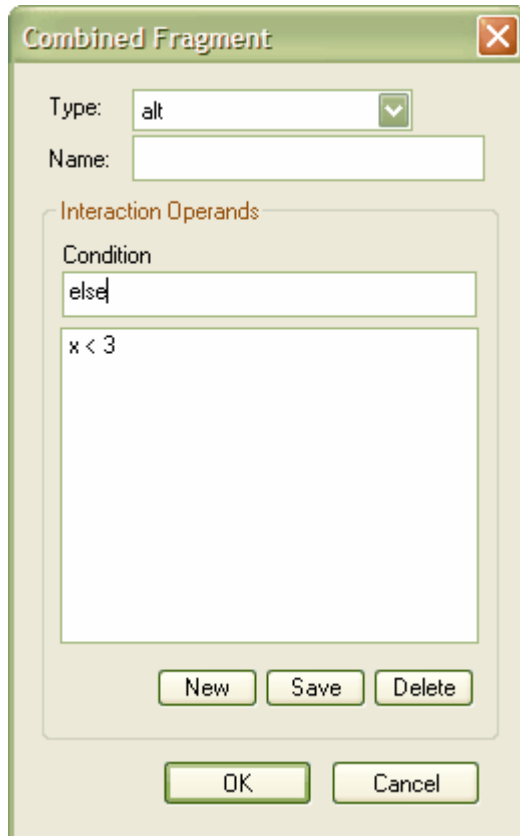
"A combined fragment defines an expression of interaction fragments. A combined fragment is defined by an interaction operator and corresponding interaction operands. Through the use of CombinedFragments the user will be able to describe a number of traces in a compact and concise manner."

5.3.3.9.1 Create a Combined Fragment

[Further Information](#)

Use the following guidelines to create a combined fragment in EA.

1. Drag the fragment element from the Interaction section of the toolbox. The following dialog will appear:



2. Specify the interaction operator in the *Type* drop-down. See the [Interaction Operators](#) topic for an explanation of the various types of combined fragments.
3. A condition or interaction constraint must be specified for each operand.
4. A rectangular frame will appear, partitioned by dashed lines into segments for each operand.
5. Adjust the frame to encompass the desired event occurrences for each operand.

Further Information

- [Combined Fragment](#)
- [Interaction Operators](#)

5.3.3.9.2 Interaction Operators

[Further Information](#)

When creating combined fragments, you must apply an appropriate interaction operator to characterize the fragment. The following table provides rough guidance on the various operators, and their corresponding

descriptions.

Interaction Operator	Description
alt	Divides up interaction fragments based on Boolean conditions.
opt	Encloses an optional fragment of interaction.
par	Indicates operands will operate in parallel.
loop	The operand will repeat a number of times, as specified by interaction constraints.
critical	Indicates a sequence that cannot be interrupted by other processing.
neg	Asserts that a fragment is invalid, and implies that all other interaction is valid.
assert	Specifies the only valid fragment to occur. Often enclosed within a consider or ignore operand.
strict	Indicates the behaviors of the operands must be processed in strict sequence.
seq	The combined fragment is weakly sequenced. This means that the ordering within operands is maintained, but the ordering between operands is undefined, so long as an event occurrence of the first operand precedes that of the second operand, if the event occurrences are on the same lifeline.
ignore	Indicates which messages should be ignored during execution, or can appear anywhere in the execution trace.
consider	Specifies which messages should be considered in the trace. This is often used to specify the resulting event occurrences with the use of an Assert operator.

Further Information

- [Combined Fragment](#)

OMG UML Specification

The OMG UML specification (UML 2.0 Superstructure, p. 410-412) states:

"The semantics of a CombinedFragment is dependent upon the interactionOperator as explained below.

"Alternatives

The interactionOperator alt designates that the CombinedFragment represents a choice of behavior. At most one of the operands will execute. The operand that executes must have an explicit or implicit guard expression that evaluates to true at this point in the interaction. An implicit true guard is implied if the operand has no guard. The set of traces that defines a choice is the union of the (guarded) traces of the operands. An operand guarded by else designates a guard that is the negation of the disjunction of all other guards in the enclosing CombinedFragment.

"Option

The interactionOperator opt designates that the CombinedFragment represents a choice of behavior where either the (sole) operand happens or nothing happens. An option is semantically equivalent to an alternative CombinedFragment where there is one operand with non-empty content and the second operand is empty.

"Break

The interactionOperator break designates that the CombinedFragment represents a breaking scenario in the sense that the operand is a scenario that is performed instead of the remainder of the enclosing InteractionFragment. Thus the break operator is a shorthand for an Alternative operator where one operand is given and the other assumed to be the rest of the enclosing InteractionFragment. Break CombinedFragments must be global relative to the enclosing InteractionFragment.

"Parallel

The interactionOperator *par* designates that the CombinedFragment represents a parallel merge between the behaviors of the operands. The eventoccurrences of the different operands can be interleaved in any way as long as the ordering imposed by each operand as such is preserved. A parallel merge defines a set of traces that describes all the ways that eventoccurrences of the operands may be interleaved without obstructing the order of the eventoccurrences within the operand.

"Weak Sequencing

The interactionOperator *seq* designates that the CombinedFragment represents a weak sequencing between the behaviors of the operands. Weak sequencing is defined by the set of traces with these properties:

1. The ordering of eventoccurrences within each of the operands are maintained in the result.
2. Eventoccurrences on different lifelines from different operands may come in any order.
3. Eventoccurrences on the same lifeline from different operands are ordered such that an eventoccurrence of the first operand comes before that of the second operand.

Thus weak sequencing reduces to a parallel merge when the operands are on disjoint sets of participants. Weak sequencing reduces to strict sequencing when the operands work on only one participant.

"Strict Sequencing

The interactionOperator *strict* designates that the CombinedFragment represents a strict sequencing between the behaviors of the operands. The semantics of the strict operation defines a strict ordering of the operands on the first level within the CombinedInteraction with operator *strict*. Therefore eventoccurrences within contained CombinedInteractions will not directly be compared with other eventoccurrences of the enclosing CombinedInteraction.

"Negative

The interactionOperator *neg* designates that the CombinedFragment represents traces that are defined to be invalid. The set of traces that defined a negative CombinedFragment is equal to the set of traces given by its (sole) operand, only that this set is a set of invalid rather than valid traces. All InteractionFragments that are different from Negative are considered positive meaning that they describe traces that are valid and should be possible.

"Critical Region

The interactionOperator *critical* designates that the CombinedFragment represents a critical region. A critical region means that the traces of the region cannot be interleaved by other Eventoccurrences (on those Lifelines covered by the region). This means that the region is treated atomically by the enclosing fragment when determining the set of valid traces. Even though enclosing CombinedFragments may imply that some Eventoccurrences may interleave into the region, such as e.g. with *par*-operator, this is prevented by defining a region. Thus the set of traces of enclosing constructs are restricted by critical regions.

"Ignore / Consider

The interactionOperator *ignore* designates that there are some message types that are not shown within this combined fragment. These message types can be considered insignificant and are intuitively ignored if they appear in a corresponding execution. Alternatively one can understand *ignore* to mean that the messages that are ignored can appear anywhere in the traces. Conversely the interactionOperator *consider* designates which messages should be considered within this CombinedFragment. This is equivalent to defining every other message to be ignored.

"Assertion

The interactionOperator *assert* designates that the CombinedFragment represents an assertion. The sequences of the operand of the assertion are the only valid continuations. All other continuations result in an invalid trace. Assertions are often combined with *Ignore* or *Consider* as shown in Figure 345.

"Loop

The interactionOperator *loop* designates that the CombinedFragment represents a loop. The loop operand will be repeated a number of times. The Guard may include a lower and an upper number of iterations of the loop as well as a Boolean expression. The semantics is such that a loop will iterate minimum the 'minint' number of times (given by the iteration expression in the guard) and at most the 'maxint' number of times. After the minimum number of iterations have executed, and the boolean expression is false the loop will terminate. The loop construct represent a recursive application of the *seq* operator where the loop operand is sequenced after the result of earlier iterations.

*"The Semantics of Gates (see also 'Gate (from Fragments)' on page 418)
The gates of a CombinedFragment represent the syntactic interface between the CombinedFragment and its surroundings, which means the interface towards other InteractionFragments. The only purpose of gates is to define the source and the target of Messages or General Order relations."*

5.3.3.10 Component

[Common Usage](#)..|..[OMG UML Specification](#)



A *component* is a modular part of a system, whose behavior is defined by its provided and required interfaces; the internal workings of the component should be invisible and its usage environment-independent. Source code files, DLLs, Java beans and other artifacts defining the system may be manifested in components.

A component may be composed of multiple classes, or components pieced together. As smaller components come together to create bigger components, the eventual system can be modeled, building-block style. By building the system in discrete components, localization of data and behavior allows for decreased dependency between classes and objects, providing a more robust and maintainable design.

Common Usage

- [Component Diagram](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 7) states:

"A modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. As such, a component serves as a type, whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics)."

5.3.3.11 Datastore

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)

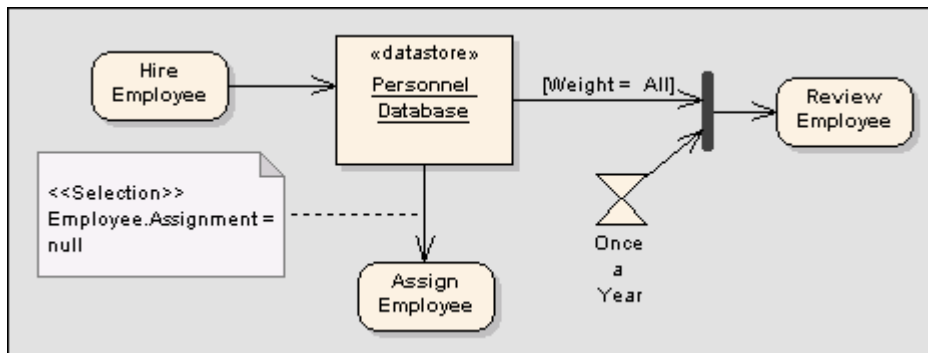


A *datastore* is an element used to define permanently stored data. A token of data that enters into a datastore is stored permanently, updating tokens for that data that already exists. A token of data that comes out of a datastore is a copy of the original data.

Use object flow connectors to link elements to datastores, as values and information are being passed between nodes. Selection and transformation behavior, together composing a sort of query, can be specified

as to the nature of data access. For instance, selection behavior determines which objects are affected by the connection to the datastore. Transformation behavior might then further specify the value of an attribute pertaining to a selected object.

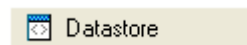
To define the behavior of access to a datastore, attach a note to the object flow connector. To do this, right-click on the object-flow and select *Attach Note or Constraint*. A dialog will indicate other flows in the diagram, to which you can select to attach the note, if the behavior applies to multiple flows. To comply with UML 2, preface behavior with the notation <<selection>> or <<transformation>>.



Refer to figure 236 (UML 2.0 Superstructure, p. 319).

Common Usage

- [Activity Diagrams](#)



Further Information

- [Activity Diagrams](#)
- [Activity](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 318) states:

"A data store node is a central buffer node for non-transient information... A data store keeps all tokens that enter it, copying them when they are chosen to move downstream. Incoming tokens containing a particular object replace any tokens in the object node containing that object."

5.3.3.12 Decision

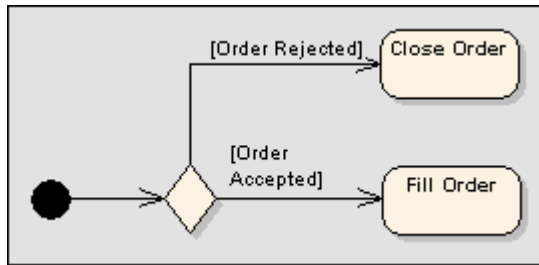
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *decision* is an element of an Activity diagram that indicates a point of conditional progression: if a condition is true, then processing continues one way, if not, then another.

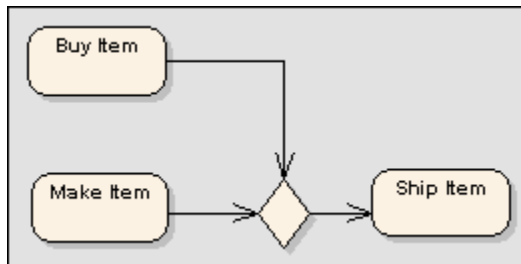
This can also be used as a merge node in that multiple alternate flows can be merged (but not synchronized) to form one flow. The following are examples of these two modes of using the decision element.

Used as a decision:



Refer to figure 238 (UML 2.0 Superstructure, p. 321).

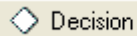
Used as a merge:



Refer to figure 266 (UML 2.0 Superstructure, p. 344).

Common Usage

- [Activity Diagrams](#)
- [Interaction Overview](#)



Further Information

- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 319 (*Decision symbol*)) states:

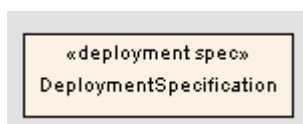
"A decision node is a control node that chooses between outgoing flows. A decision node has one incoming edge and multiple outgoing activity edges."

The OMG UML specification (*UML 2.0 Superstructure*, p. 343 (*Merge symbol*)) also states:

"A merge node is a control node that brings together multiple alternate flows. It is not used to synchronize concurrent flows but to accept one among several alternate flows...A merge node has multiple incoming edges and a single outgoing edge."

5.3.3.13 Deployment Spec

[Common Usage](#).. [OMG UML Specification](#)

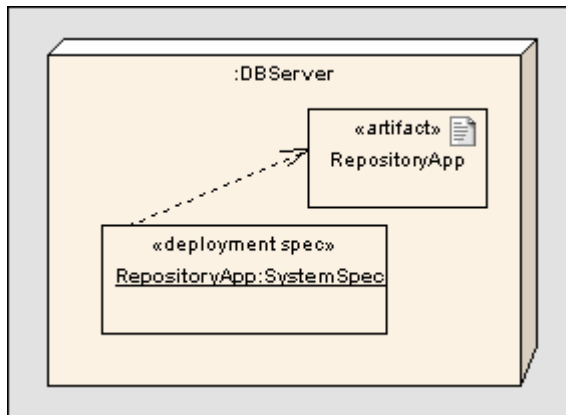


A *deployment specification (spec)* specifies parameters guiding deployment of an artifact, as is necessary with

most hardware and software technologies. A specification lists those properties that must be defined for deployment to occur. An instance of this specification specifies the values for the parameters; a single specification can be instantiated for multiple artifacts.

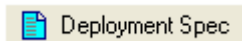
These specifications can be extended by certain component profiles. Examples of standard tagged values that a profile might add to a deployment specification are «concurrencyMode» with tagged values {thread, process, none} or «transactionMode» with tagged values {transaction, nestedTransaction, none}.

The following example depicts the artifact RepositoryApp deployed on the server node, per specifications of RepositoryApp, instantiated from the deployment specification SystemSpec.



Common Usage

- [Deployment Diagrams](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 190) states:

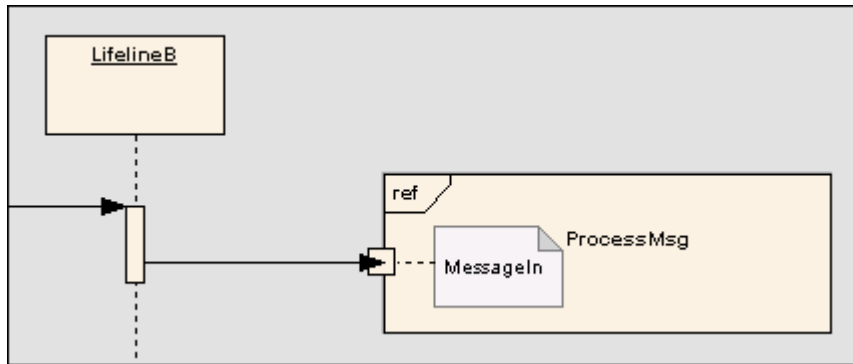
"A deployment specification specifies a set of properties which determine execution parameters of a component artifact that is deployed on a node. A deployment specification can be aimed at a specific type of container. An artifact that reifies or implements deployment specification properties is a deployment descriptor."

5.3.3.14 Diagram Gate

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



A *diagram gate* is a simple graphical way to indicate the point at which messages can be transmitted into and out of interaction fragments. A fragment might need to receive or deliver a message; internally, an ordered message reflects this need, with a gate indicated on the boundary of the fragment's frame. Any external messages 'synching' with this internal message must correspond appropriately. Gates can appear on interactions, interaction occurrences and combined fragments (to specify the expression).

**Common Usage**

- [Timing Diagrams](#)
- [Sequence Diagrams](#)

 Diagram Gate
Further Information

- [Interactions](#)
- [Interaction Occurrences](#)
- [Combined Fragments](#)

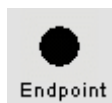
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 418) states:

"A Gate is a connection point for relating a Message outside an InteractionFragment with a Message inside the InteractionFragment... Gates are connected through Messages. A Gate is actually a representative of an EventOccurrence that is not in the same scope as the Gate. Gates play different roles: we have formal gates on Interactions, actual gates on InteractionOccurrences, expression gates on CombinedFragments."

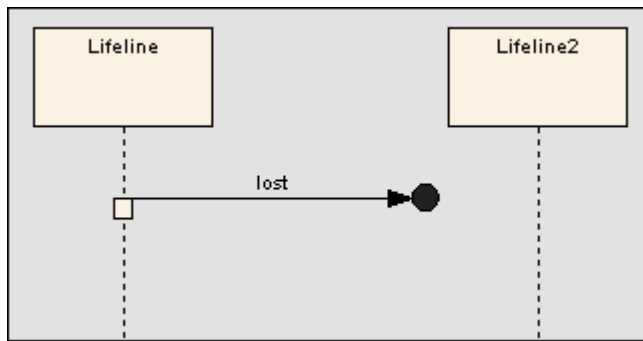
5.3.3.15 Endpoint

[Common Usage](#)...[OMG UML Specification](#)



An *endpoint* is used in interaction diagrams to reflect a lost or found message in sequence. To model this, drag an endpoint element onto the workspace. With Sequence diagrams, drag a message from the appropriate lifeline to the endpoint. With Timing diagrams, the message connecting the lifeline to the endpoint will require some timing specifications to draw the connection.

The following example depicts an lost message in a Sequence diagram.

**Common Usage**

- [Sequence Diagram](#)
- [Timing Diagram](#)

Endpoint

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 429) states:

"A lost message is a message where the sending event occurrence is known, but there is no receiving event occurrence. We interpret this to be because the message never reached its destination.

"A found message is a message where the receiving event occurrence is known, but there is no (known) sending event occurrence. We interpret this to be because the origin of the message is outside the scope of the description. This may for example be noise or other activity that we do not want to describe in detail."

5.3.3.16 Entry Point

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Entry points are used to define the beginning of a state machine. An entry point exists for each region, directing the initial concurrent state configuration.

Common Usage

- [State Machine Diagram](#)

Entry

Further Information

- [Pseudo-States](#)

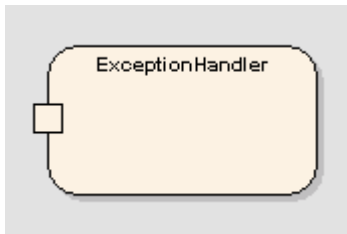
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"An entry point pseudostate is an entry point of a state machine. In each region of the state machine it has a single transition to a vertex within the same region."

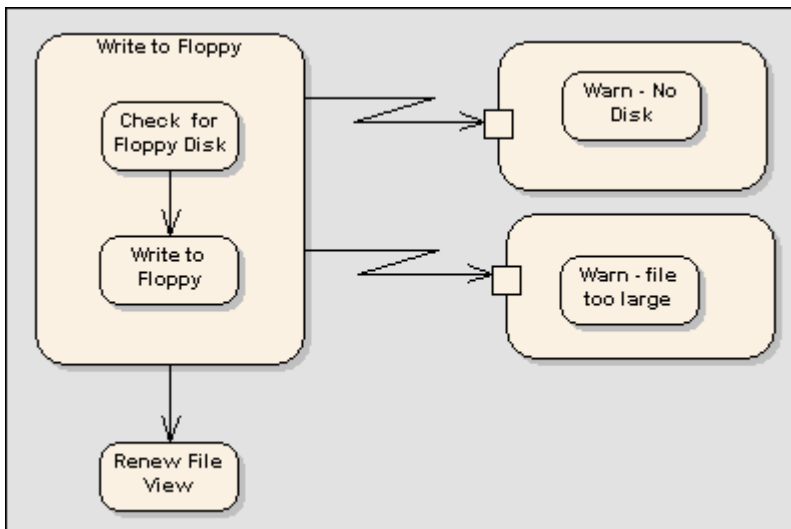
5.3.3.17 Exception

[Common Usage..](#) [Further Information..](#) [OMG UML Specification](#)




The *exception handler* element defines the group of operations to carry out when an exception occurs. The protected element may contain a set of operations and is linked to the exception handler via an interrupt flow connector. Any defined error contained within an element's parts can trigger the flow to move to an exception.

The following example shows exceptions that can occur when trying to write to a floppy disk. If the disk drive specified does not contain a disk, this causes an exception, with an action prompting the user to load a disk. Another exception will be triggered if the remaining disk space is less than the size of the file to be written, with the handling action alerting the user to the problem.



Common Usage

- [Activity Diagrams](#)

 Exception

Further Information

- [Interruptible Activity Region](#)
- [Interrupt Flow](#)
- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure, p. 8*) states:

"A special kind of signal, typically used to signal fault situations. The sender of the exception aborts execution and execution resumes with the receiver of the exception, which may be the sender itself. The receiver of an exception is determined implicitly by the interaction sequence during execution; it is not explicitly specified."

5.3.3.18 Expansion Region

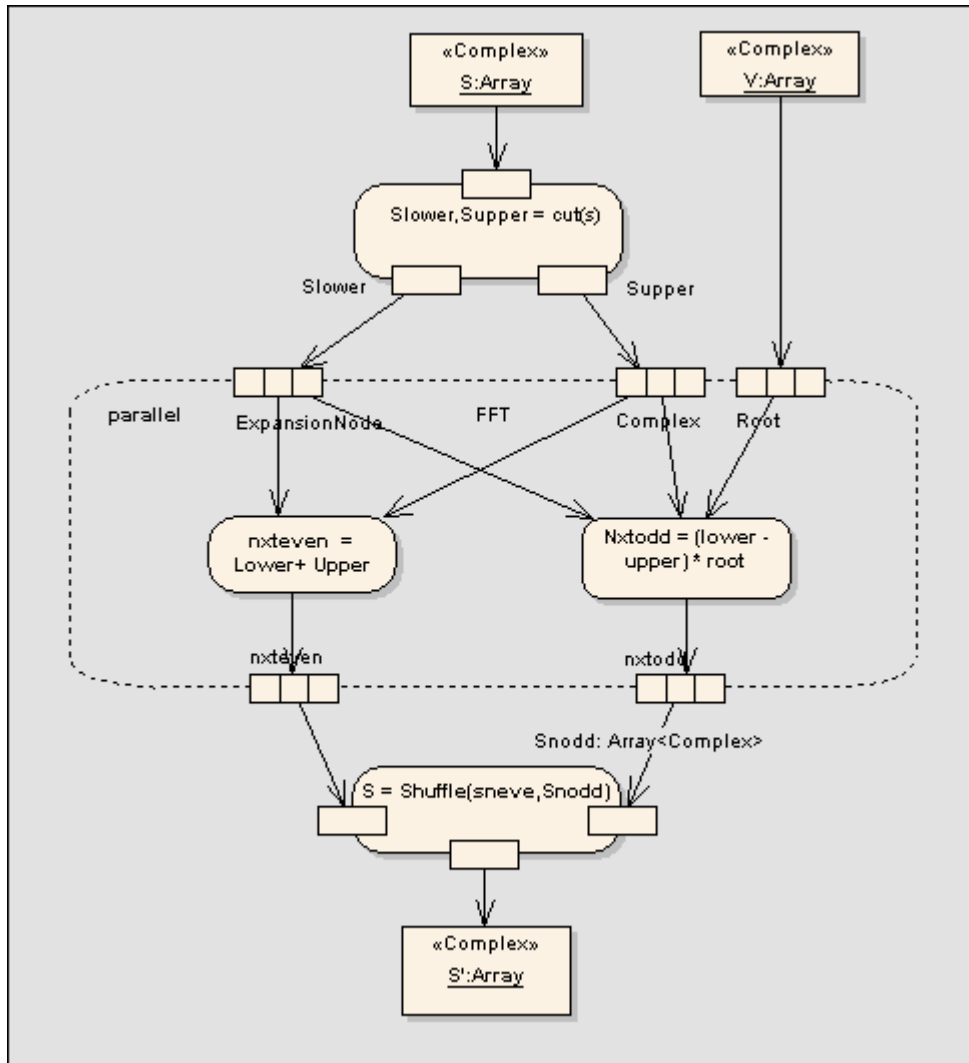
[Common Usage](#)..[Further Information](#)..[OMG UML Specification](#)



An *expansion region* surrounds a process to be imposed multiple times on the incoming data, once for every element in the input collection. If there are multiple inputs, the collection sizes must match, and the elements within each collection must be of the same type. Similarly, any outputs must be in the form of a collection matching the size of the inputs.

The concurrency of the expansion region's multiple executions can be specified as type parallel, iterative, or stream. Parallel reflects that the elements in the incoming collections can be processed at the same time or overlapping, whereas an iterative concurrency type specifies that execution must occur sequentially. A stream-type expansion region indicates that the input and output come in and exit as streams, and that the expansion region's process must have some method to support streams.

Modify the kind of expansion region by opening the *Properties* dialog of an expansion region, then under the *General* tab selecting *Kind* from the combo box.



Refer to figure 247 (UML 2.0 Superstructure, p. 330).

Common Usage

- [Activity Diagrams](#)

Region

Further Information

- [Add an Expansion Region](#)
- [Activity Diagrams](#)
- [Regions](#)

The OMG UML specification states:

The OMG UML specification (*UML 2.0 Superstructure*, p. 325) states:

"An expansion region is a structured activity region that executes multiple times corresponding to elements of an input collection."

5.3.3.18.1 Add Expansion Region

[Further Information](#)

After adding a region element to the diagram, the following prompt appears:



By default this is set to `InterruptibleActivityRegion`.

1. Select the type: *ExpansionRegion*.
2. Choose the concurrency attribute by selecting an entry from the drop down *Kind*.

Further Information

- [Expansion Region](#)

5.3.3.19 Exit Point

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



Exit points are used in submachine states and state machines to denote the point where the machine will be exited and the transition sourcing this exit point, for submachines, will be triggered. Exit points are a type of pseudo-state used in the state machine diagram.

Common Usage

- [State Machine Diagram](#)



Further Information

- [Pseudo-States](#)

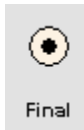
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"An exit point pseudostate is an exit point of a state machine. Entering an exit point within any region of the state machine referenced by a submachine state implies the exit of this submachine state and the triggering of the transition that has this exit point as source in the state machine enclosing the submachinestate."

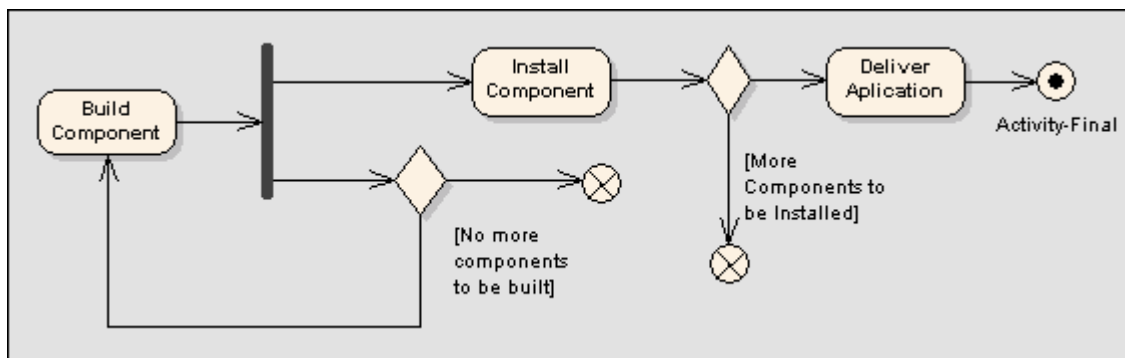
5.3.3.20 Final

[Common Usage](#)..|.. [Further Information](#)..|.. [OMG UML Specification](#)



There are two nodes used to define a *final* state in an activity, both defined in UML 2.0 as of type "final node". The final element, shown above, indicates the completion of an activity - upon reaching the final, all execution in the activity diagram is aborted. The other type of final node, [flow final](#), depicts an exit from the system but has no effect on other executing flows in the activity.

The following example illustrates the development of an application. The process comes to a flow final node when there are no more components to be built; note that the fork element indicates a concurrent process with the building of new components and installation of completed components. The flow final terminates only the sub-process building components. Similarly, only those tokens entering the decision branch, for the installation of further components, terminate with the connecting flow final. It is only after the Deliver Application activity is completed, after the control flow reaches the final node, that all flows stop.



Refer to figure 251 (UML 2.0 Superstructure, p. 332).

Common Usage

- [Activity Diagram](#)



Further Information

- [Activity Diagrams](#)
- [Flow Final](#)

OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 298) states:

"An activity may have more than one activity final node. The first one reached stops all flows in the activity."

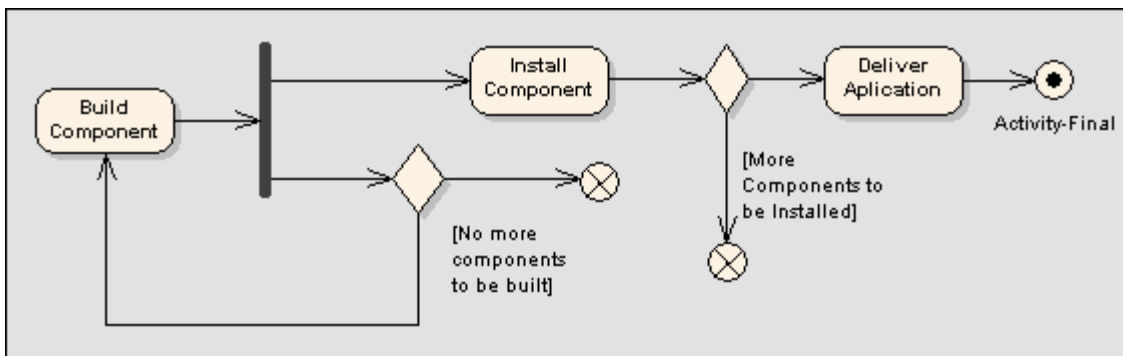
5.3.3.21 Flow Final

[Common Usage](#) .. [Further Information](#) .. [OMG UML Specification](#)



There are two nodes used to define a *final* state in an activity, both defined in UML 2.0 as of type "final node". The *flow final* element depicts an exit from the system as opposed to the activity final which represents the completion of the activity. Only the flow entering the flow final node exits the activity; other flows continue undisturbed.

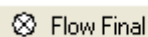
The following example illustrates the development of an application. The process comes to a flow final node when there are no more components to be built; note that the fork element indicates a concurrent process with the building of new components and installation of completed components. The flow final terminates only the sub-process building components. Similarly, only those tokens entering the decision branch, for the installation of further components, terminate with the connecting flow final. It is only after the Deliver Application activity is completed, after the control flow reaches the final node, that all flows stop.



Refer to figure 251 (UML 2.0 Superstructure, p. 332).

Common Usage

- [Activity Diagrams](#)



Further Information

- [Activity Diagrams](#)
- [Activity Final](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 333) states:

"A flow final destroys all tokens that arrive at it. It has no effect on other flows in the activity."

5.3.3.22 Fork/Join

[Common Usage](#) .. [Further Information](#) .. [OMG UML Specification](#)



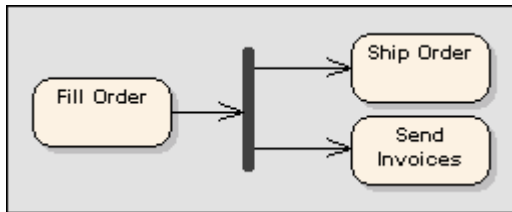
The *fork/join* elements have different modes of use. These are as follows:

- To fork or split the flow into a number of concurrent flows.
- To join the flow of a number of concurrent flows.
- To both join and fork a number of incoming flows to a number of outgoing flows.

With respect to State Machine diagrams, forks and joins are utilized as pseudo-states. Other pseudo-states include history states, entry points and exit points. Forks are used to split an incoming transition into concurrent multiple transitions leading to different target states. Joins are used to merge concurrent multiple transitions into a single transition leading to a single target. They are semantic inverses. To learn more about these individual elements, consult their topics, listed [below](#).

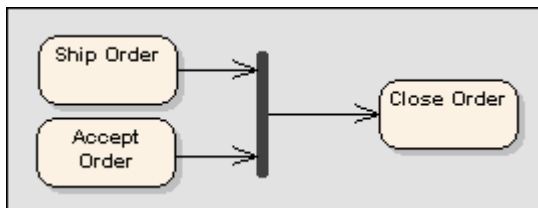
Some examples of fork/join nodes include:

Fork or split the flow into a number of concurrent flows:



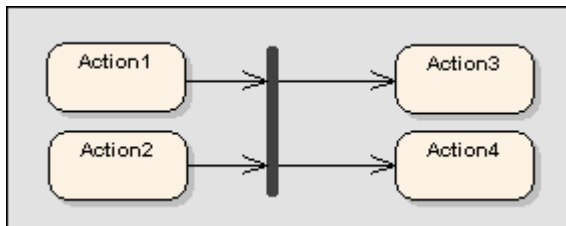
Refer to figure 255 (UML 2.0 Superstructure, p. 335).

Join the flow of a number of concurrent flows:



Refer to figure 263 (UML 2.0 Superstructure, p. 340).

Join and fork a number of incoming flows to a number of outgoing flows:



Common Usage

- [Activity Diagrams](#)
- [State Machine Diagram](#)



Further Information

- [Fork](#)
- [Join](#)
- [Pseudo-States](#)

OMG UML Specification

Fork

The OMG UML specification (*UML 2.0 Superstructure*, p. 333) states:

"A fork node is a control node that splits a flow into multiple concurrent flows... A fork node has one incoming edge and multiple outgoing edges."

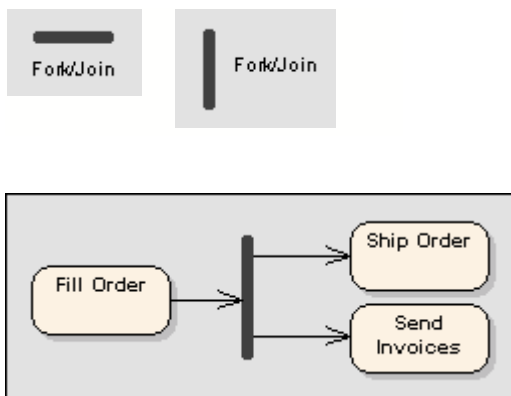
Join

The OMG UML specification (*UML 2.0 Superstructure*, p. 338-339) states:

"A join node is a control node that synchronizes multiple flows...A join node has multiple incoming edges and one outgoing edge."

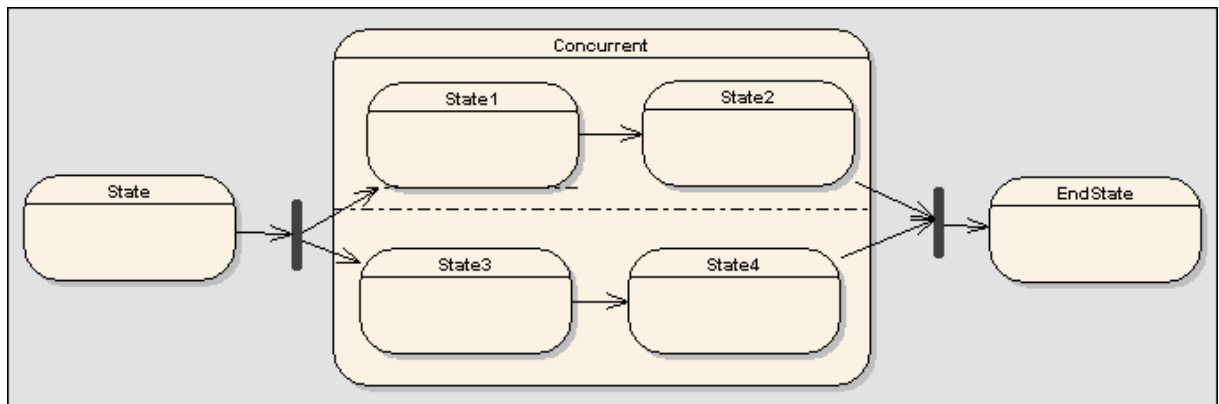
5.3.3.22.1 Fork

[Further Information...OMG UML Specification](#)



Refer to figure 255 (UML 2.0 Superstructure, p. 335).

With respect to state machine diagrams, a *fork* pseudo-state signifies that its incoming transition will come from a single state, and it will have multiple outgoing transitions. These transitions must occur concurrently, requiring the use of concurrent regions, as depicted below in the composite state. Unlike choice or junction pseudo-states, forks may not have triggers or guards. The following diagram demonstrates a fork pseudo-state dividing into two concurrent regions, which then return to the EndState via the join.



Further Information

- [Pseudo-States](#)
- [Fork/Join](#)
- [Regions](#)

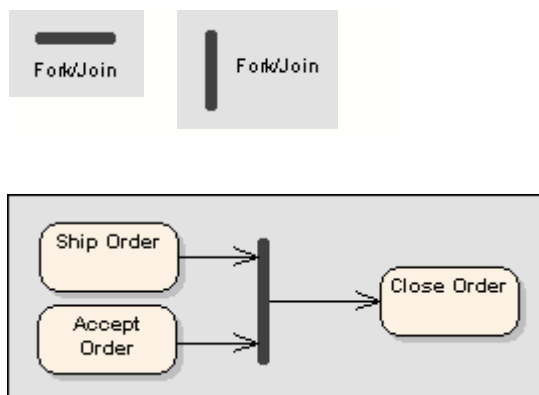
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"fork vertices serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices (i.e. vertices in different regions of a composite state). The segments outgoing from a fork vertex must not have guards or triggers."

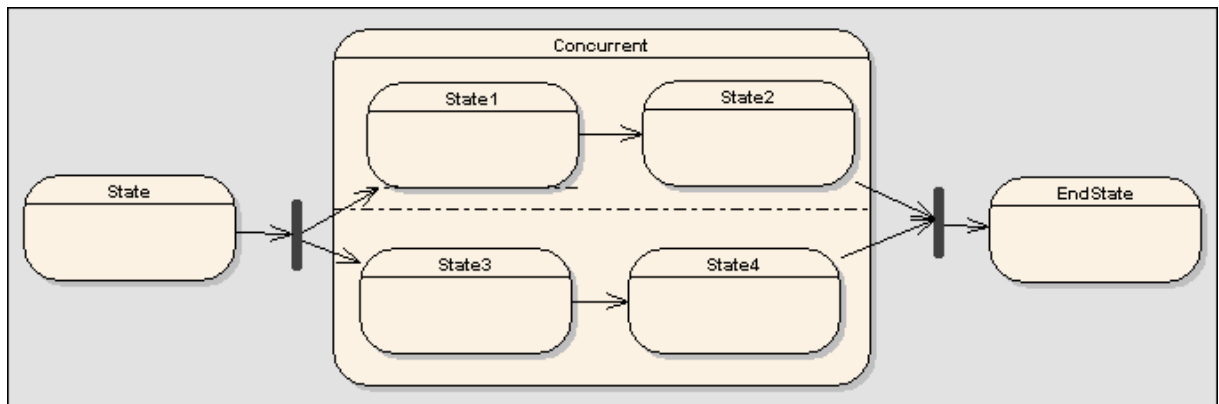
5.3.3.22.2 Join

[Further Information...](#) [OMG UML Specification](#)



Refer to figure 263 (UML 2.0 Superstructure, p. 340).

The *join* element is utilized by the activity and state machine diagrams. The above example illustrates a join transition between activities. With respect to state machine diagrams, a join pseudo-state indicates multiple states concurrently transitioning into the join and onto a single state. Unlike choice or junction pseudo-states, joins may not have triggers or guards. The following diagram demonstrates a fork pseudo-state dividing into two concurrent regions, which then return to the EndState via the join.



Further Information

- [Pseudo-States](#)
- [Fork/Join](#)
- [Regions](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"join vertices serve to merge several transitions emanating from source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers."

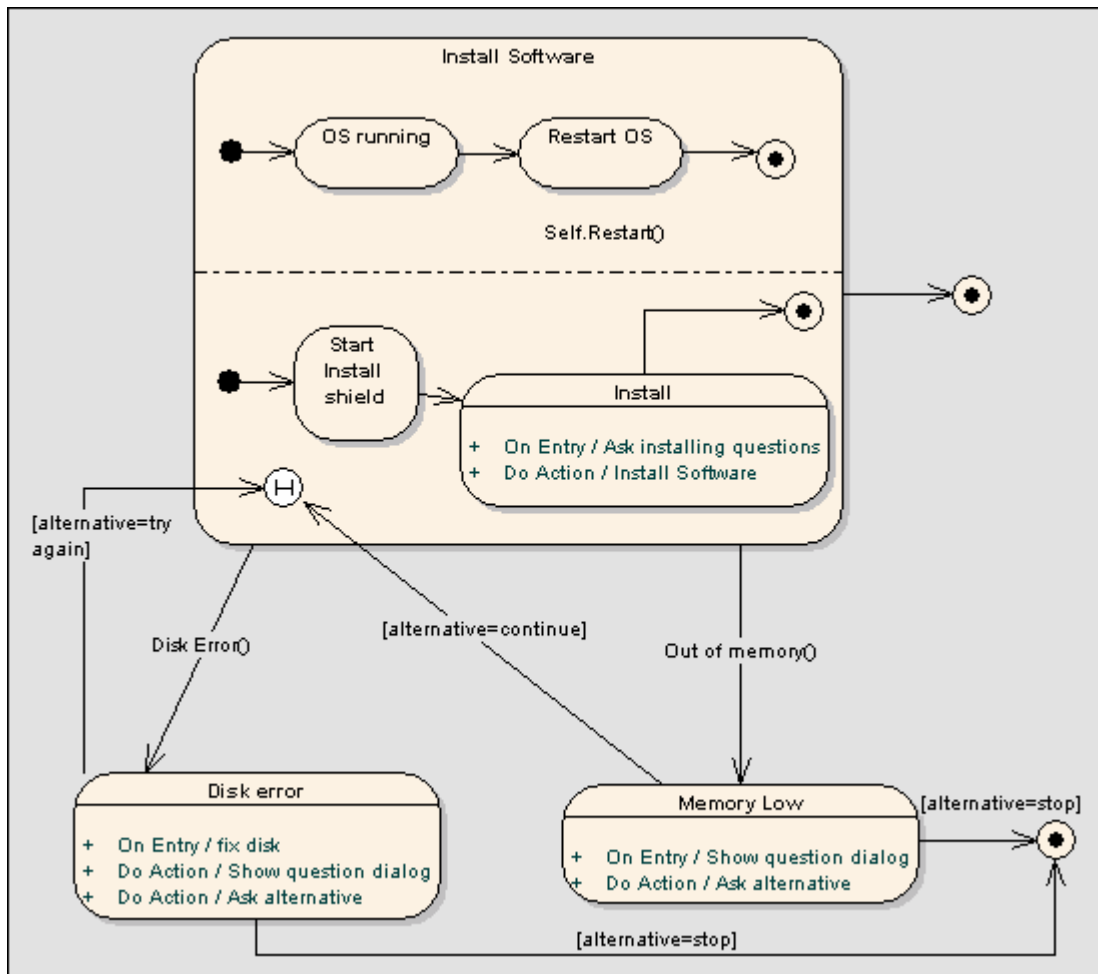
5.3.3.23 History

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)




There are two types of *history* pseudo-states defined in UML, shallow and deep history. A shallow history sub-state is used to represent the most recently active sub-state of a composite state; this pseudo-state does not recurse into this sub-state's active configuration, should one exist. A single connector can be used to depict the default shallow history state, in case the composite state has never been entered.

A deep history sub-state, in contrast, reflects the most recent active configuration of the composite state. This includes active sub-states of all regions, and recurses into those sub-states' active sub-states, should they exist. At most one deep history and one shallow history can dwell within a composite state.

**Common Usage**

- [State Machine Diagram](#)

 History State
Further Information

- [Pseudo-States](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 470) states:

"deepHistory represents the most recent active configuration of the composite state that directly contains this pseudostate; e.g. the state configuration that was active when the composite state was last exited. A composite state can have at most one deep history vertex. At most one transition may originate from the history connector to the default deep history state. This transition is taken in case the composite state had never been active before. Entry actions of states entered on the path to the state represented by a deep history are performed. shallowHistory represents the most recent active substate of its containing state (but not the substates of that substate). A composite state can have at most one shallow history vertex. A transition coming into the shallow history vertex is equivalent to a transition coming into the most recent active substate of a state. At most one transition may originate from the history connector to the default shallow history state. This transition is taken in case the composite state had never been active before. Entry actions of states entered on the path to the state represented by a shallow history are performed."

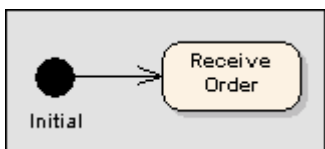
5.3.3.24 *Initial*

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



The *initial* element is used by the Activity and State Machine diagrams. In Activity diagrams, it defines the start of a flow when an activity is invoked. With State Machines, the initial element is a [pseudo-state](#) used to denote the default state of a composite state; there can be one initial vertex in each region of the composite state.

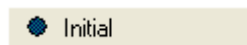
This simple example shows the start of a flow to receive an order.



Refer to figure 257 (UML 2.0 Superstructure, p. 336).

Common Usage

- [State Machine Diagram](#)
- [Activity Diagram](#)



Further Information

- [Pseudo-States](#)
- [Final](#)

OMG UML Specification

The OMG UML specification ([UML 2.0 Superstructure](#), p. 470) states:

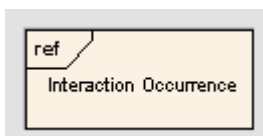
"An initial pseudostate represents a default vertex that is the source for a single transition to the default state of a composite state. There can be at most one initial vertex in a region. The initial transition may have an action."

The OMG UML specification ([UML 2.0 Superstructure](#), p. 335) states:

"An initial node is a control node at which flow starts when the activity is invoked."

5.3.3.25 *Interaction Occurrence*

[Common Usage](#)..|..[OMG UML Specification](#)



An *interaction occurrence* is a reference to an existing interaction element. Interaction occurrences are visually represented by a frame, with "ref" in the frame's title space. The diagram name is indicated in the frame contents. To create an interaction occurrence, simply drag an interaction diagram onto an open interaction

diagram's workspace. A dialog will open, providing configuration options.

Common Usage

- [Sequence Diagrams](#)

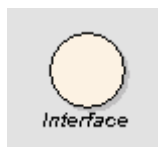
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 423) states:

"An InteractionOccurrence refers to an Interaction. The InteractionOccurrence is a shorthand for copying the contents of the referred Interaction where the InteractionOccurrence is. To be accurate the copying must take into account substituting parameters with arguments and connect the formal gates with the actual ones. It is common to want to share portions of an interaction between several other interactions. An InteractionOccurrence allows multiple interactions to reference an interaction that represents a common portion of their specification."

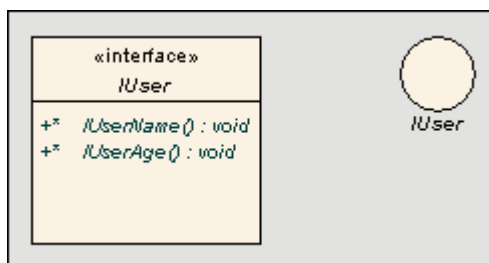
5.3.3.26 Interface

[Common Usage](#) | [OMG UML Specification](#)



An *interface* is a specification of behavior that implementers agree to meet. It is a contract. By implementing an interface, classes are guaranteed to support a required behavior, which allows the system to treat non-related elements in the same way - ie. through the common interface.

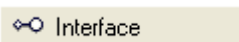
Interfaces may be drawn in a similar method to a class, with operations specified, as shown below. They may also be drawn as a circle with no explicit operations detailed. Use the right-click context menu option *Use Circle Notation* to switch between styles. When drawn as a circle, realization links to the circle form of notation are drawn without target arrows.



Note: An interface cannot be instantiated (ie. you cannot create an object from an Interface). You must create a class that 'implements' the interface specification, and in the class body place operations for each of the Interface operations. You can then instantiate the class.

Common Usage

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



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 114) states:

"An interface is a kind of classifier that represents a declaration of a set of coherent public features and obligations. In a sense, an interface specifies a kind of contract which must be fulfilled by any instance of a classifier that realizes the interface. The obligations that may be associated with an interface are in the form of various kinds of constraints (such as pre- and postconditions) or protocol specifications, which may impose ordering restrictions on interactions through the interface. Since interfaces are declarations, they are not directly instantiable. Instead, an interface specification is realized by an instance of a classifier, such as a class, which means that it presents a public facade that conforms to the interface specification. Note that a given classifier may realize more than one interface and that an interface may be realized by a number of different classifiers."

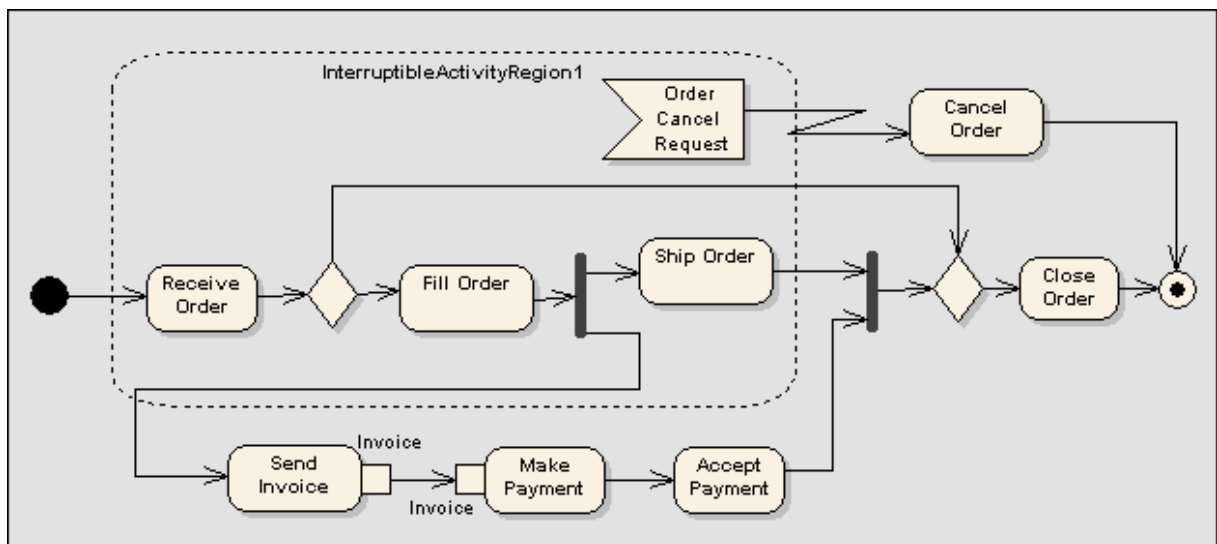
5.3.3.27 Interruptible Activity Region

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *interruptible activity region* surrounds a group of activity elements, all affected by certain interrupts in such a way that all tokens passing within the region are terminated should the interruption(s) be raised. Any processing occurring within the bounds of an interruptible activity region is terminated when a flow is instigated across an interrupt flow to an external element.


The example below illustrates that an order cancellation will kill any processing of the order at the receipt, filling or shipping stage.



Refer to figure 260 (UML 2.0 Superstructure, p. 338).

Common Usage

- [Activity Diagram](#)

 Region

Further Information

- [Add an Interruptible Activity Region](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 337) states:

"An interruptible region contains activity nodes. When a token leaves an interruptible region via edges designated by the region as interrupting edges, all tokens and behaviors in the region are terminated."

5.3.3.27.1 Add Interruptible Activity Region[Further Information](#)

After adding a region element to an activity diagram, the following prompt appears:



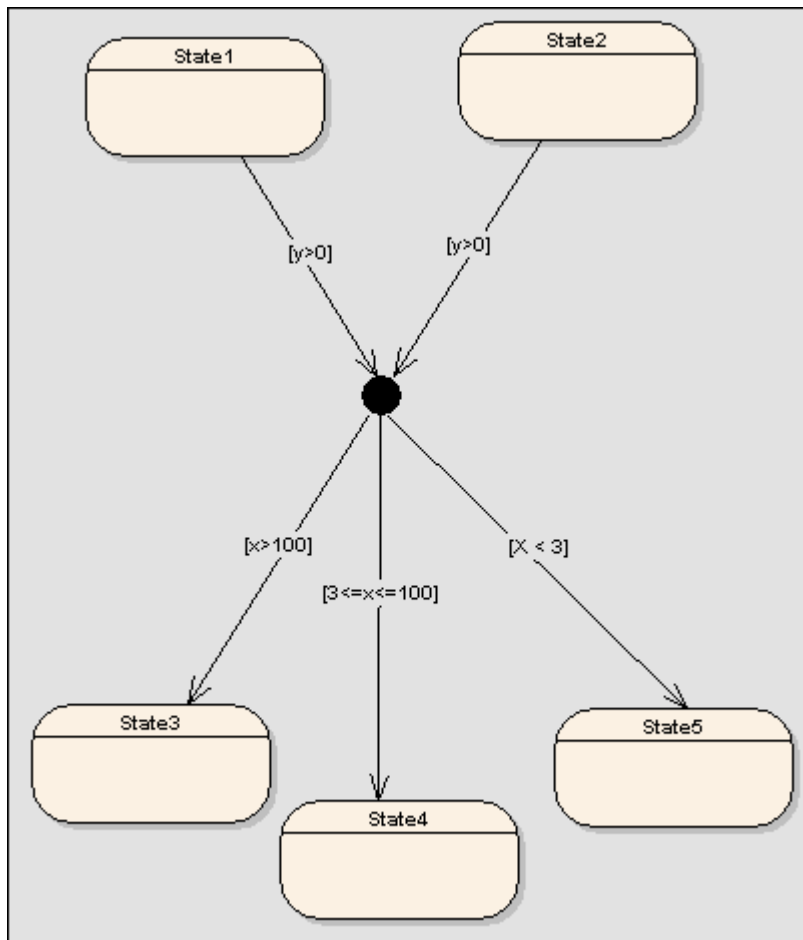
By default this is set to InterruptibleActivityRegion and the *Kind* option is disabled. Press **OK**.

Further Information


- [Interruptible Region](#)

5.3.3.28 Junction[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)

Junction pseudo-states are used to design complex transitional paths. A junction can be used to combine or merge multiple paths into a shared transition path. Alternatively, a junction can split an incoming path into multiple paths, similar to a fork pseudostate. Unlike forks or joins, junctions can apply guards to each incoming or outgoing transition, such that if the guard expression is false, the transition is disabled. The following example illustrates how guards can be applied to transitions coming into or out of a junction pseudo-state.

**Common Usage**

- [State Machine Diagram](#)

 Junction
Further Information

- [Pseudo-States](#)

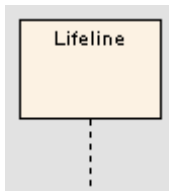
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"junction vertices are semantic-free vertices that are used to chain together multiple transitions. They are used to construct compound transition paths between states. For example, a junction can be used to converge multiple incoming transitions into a single outgoing transition representing a shared transition path (this is known as an merge). Conversely, they can be used to split an incoming transition into multiple outgoing transition segments with different guard conditions. This realizes a static conditional branch. (In the latter case, outgoing transitions whose guard conditions evaluate to false are disabled. A predefined guard denoted "else" may be defined for at most one outgoing transition. This transition is enabled if all the guards labeling the other transitions are false.) Static conditional branches are distinct from dynamic conditional branches that are realized by choice vertices."

5.3.3.29 Lifeline

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



A *lifeline* is an individual participant in an interaction (ie. lifelines cannot have multiplicity). A lifeline represents a distinct connectable element. To specify that representation within Enterprise Architect, right-click the lifeline and select **Set Instance Classifier**. A dialog will appear containing a selectable list of all project classifiers. Lifelines are available in Sequence and Timing diagrams, and although the representation differs between the two, the meaning of a lifeline is the same.

Common Usage

- [Sequence Diagrams](#)
- [Timing Diagrams](#)



Further Information

- [State Lifeline](#)
- [Value Lifeline](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p.427) states:

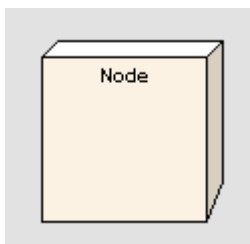
"A lifeline represents an individual participant in the Interaction. While Parts and StructuralFeatures may have multiplicity greater than 1, Lifelines represent only one interacting entity.

"Lifeline is a specialization of NamedElement.

"If the referenced ConnectableElement is multivalued (i.e. has a multiplicity > 1), then the Lifeline may have an expression (the 'selector') that specifies which particular part is represented by this Lifeline. If the selector is omitted this means that an arbitrary representative of the multivalued ConnectableElement is chosen."

5.3.3.30 Node

[Common Usage](#)..|..[OMG UML Specification](#)




A *node* is a physical piece of equipment on which the system will be deployed - for example a workgroup server or workstation. A node usually hosts components and other executable pieces of code, which again may be linked to particular processes or execution spaces. Typical nodes are client workstations, application

servers, mainframes, routers, terminal servers, etc.

Nodes are used in deployment diagrams to model the deployment of a system, and to illustrate the physical allocation of implemented artifacts.

Common Usage

- [Deployment Diagram](#)

 Node

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 195) states:

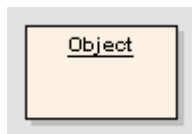
"In the metamodel, a Node is a subclass of Class. It is associated with a Deployment of an Artifact. It is also associated with a set of Elements that are deployed on it. This is a derived association in that these PackageableElements are involved in a Manifestation of an Artifact that is deployed on the Node. Nodes may have an internal structure defined in terms of parts and connectors associated with them for advanced modeling applications."

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) also states:

"A classifier that represents a run-time computational resource, which generally has at least memory and often processing capability. Run-time objects and components may reside on nodes."

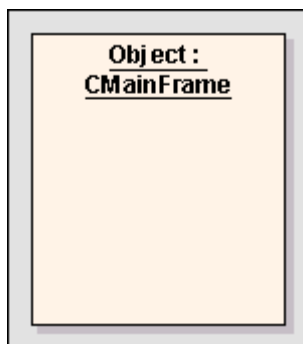
5.3.3.31 Object

[Common Usage](#).. [Further Information](#).. [OMG UML Specification](#)



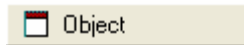
An *object* is an *instance* of a class at run time. For example the car with the license plate "AAA-001" is an instance of the general class of cars (with a license plate number attribute). Objects are often used in analysis to represent the numerous artifacts and items that exist in any business - pieces of paper, faxes, information, etc.

Early in analysis, objects can be used to quickly capture all the things that are of relevance within the system domain. As the model progresses these analysis objects are refined into generic classes from which instances may be derived to represent common business items. Once classes are defined, *objects* may be typed - that is they may have a classifier set that indicates their base type - see [Set Instance Classifier](#).



Common Usage

- [Object Diagram](#)
- [Composite Structure Diagram](#)
- [Communication Diagram](#)

**Further Information**

- [Setting the Instance Classifier](#)
- [Setting the Run-time State](#)

OMG UML Specification

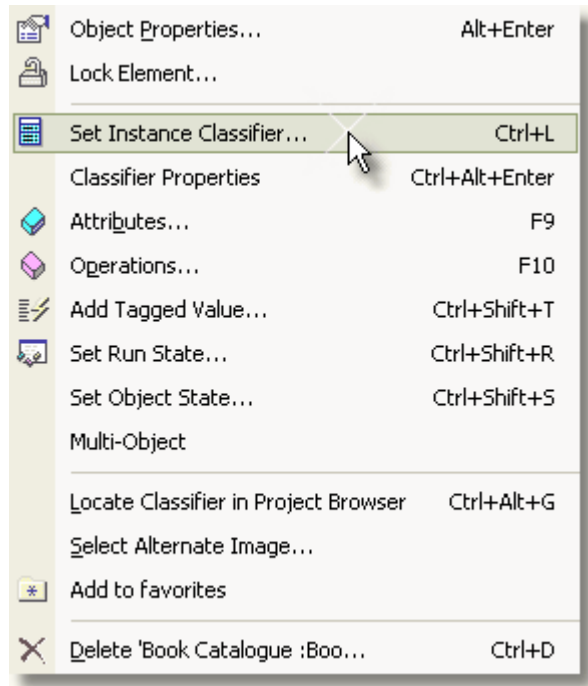
The OMG UML specification states:

"An object represents a particular instance of a class. It has identity and attribute values. A similar notation also represents a role within a collaboration because roles have instance-like characteristics."

5.3.3.31.1 Instance Classifier**[Further Information](#)**

To set the object base type or 'classifier', follow these steps:

1. Select an object in the diagram view.
2. Right click to view the context menu.



3. Select the **Set Instance Classifier** menu item.
4. From the list of available classes, select the required type.

Further Information

- [Object](#)

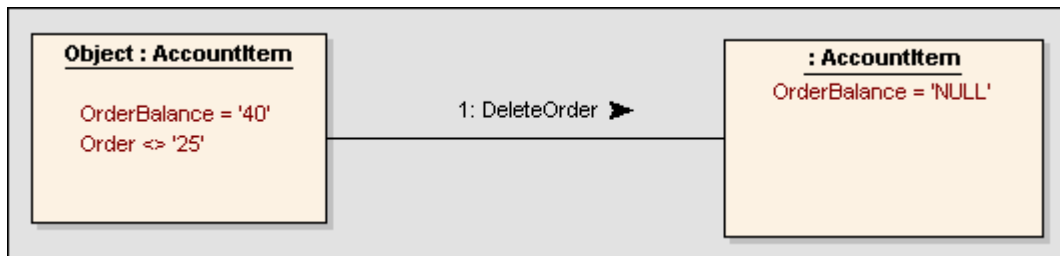
5.3.3.31.2 Run-time State

[Further Information](#)

At run-time, an object instance can have specific values for its attributes, or exist in a particular state. To model the varying behavior of objects at run-time, utilize instance values and run-time states.

Typically there is interest in the run-time behavior of objects that already have a classifier set. You can select from the classifier's attribute list and apply specific values for your object instance. If the classifier has a child state machine, its states will propagate to a list, where the run-time state for the object can be defined. To do this, refer to the topics below.

The following example defines run-time values for the listed variables, which are attributes of the instances' classifier 'AccountItem'.



Further Information

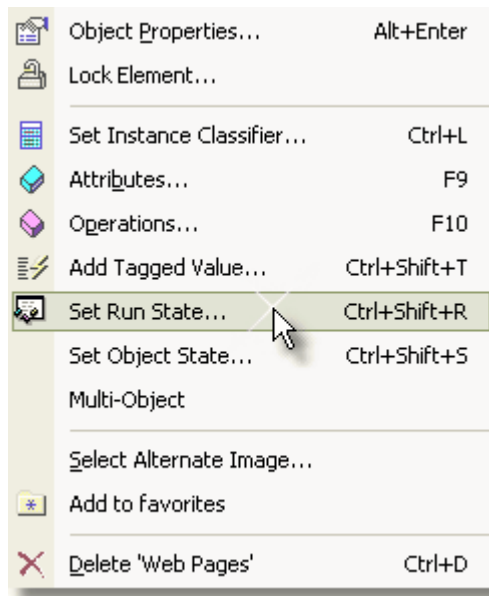
- [Object](#)
- [Define an Instance Variable](#)
- [Remove a Defined Variable](#)
- [Define a Run-time State](#)

5.3.3.31.2.1 Define a Run-time Variable

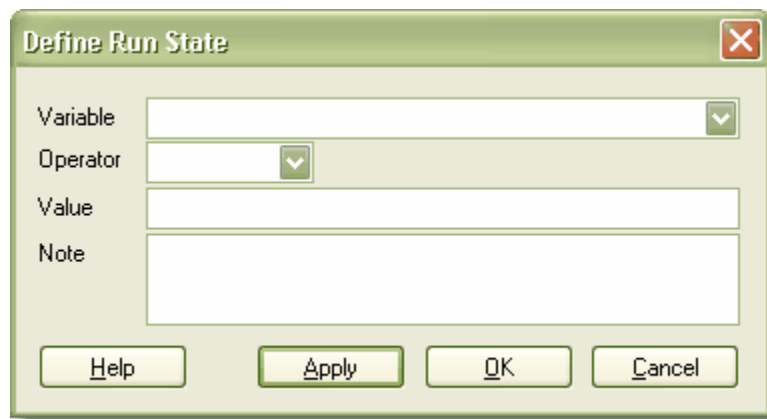
[Further Information](#)

To add new instance variables to your object by using the *Define Run State* dialog, follow the steps below:

1. Right-click on the object to bring up the object context menu.



2. If Instance Variables are supported you can select the *Set Run State* option (or press *Ctrl+Shift+R*). The *Run State* dialog will appear.



3. Select the variable from the *Variable* drop down list -OR- type in the name of a new variable.
4. Set the *Operator*, the *Value* and optionally enter some *Notes*.
5. Press *OK* to save.

Further Information

- [Runtime State](#)
- [Remove a Defined Variable](#)
- [Define a Run-time State](#)

5.3.3.31.2.2 Remove a Defined Variable

[Further Information](#)

To delete a run-time variable:

1. Right click on an object to bring up the object context menu.
2. Select *Set Run State* to bring up the *Run State* dialog.
3. Select the variable to delete from the *Variable* drop down list.
4. Clear the *Value* field.
5. Press *OK*.

Further Information

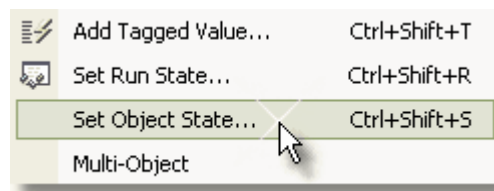
- [Runtime State](#)
- [Define a Run-time Variable](#)
- [Define a Run-time State](#)

5.3.3.31.2.3 Define a Run-time State

[Further Information](#)

To set the run-time state for a class instance, follow the steps below:

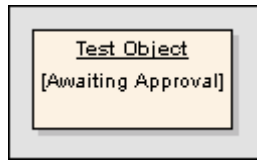
1. Right click on an object and select the *Set Object State* menu item.



2. The *Set Object Runtime State* dialog is displayed. Insert the required state - eg. 'Awaiting Approval'. If the associated classifier has a child state machine, those states will propagate into this drop-down list.



3. Press *OK* to apply the state. The object now shows the run-time state in square brackets below the object name (see below).

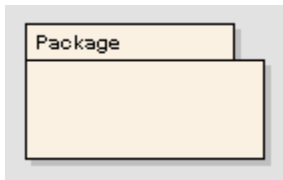


Further Information

- [Runtime State](#)
- [Define a Run-time Variable](#)
- [Remove a Defined Variable](#)

5.3.3.32 Package

[Common Usage](#) | [OMG UML Specification](#)



A *package* is a namespace as well as an element that can be contained in other package's namespaces. A package can own or merge with other packages, and its elements can be imported into a package's namespace. In addition to using packages in the Project Browser to organize your project contents, these packages can also be dragged onto a diagram workspace for structural or relational depictions, including package imports or merges.

Common Usage

- [Package Diagrams](#)
- [Class Diagrams](#)



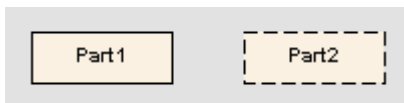
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 99) states:

"A package is a namespace for its members, and may contain other packages. Only packageable elements can be owned members of a package. By virtue of being a namespace, a package can import either individual members of other packages, or all the members of other packages. In addition a package can be merged with other packages."

5.3.3.33 Part

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



Parts are run-time instances of classes or interfaces. Multiplicity can be specified for a part, using the notation $[x\{\dots y\}]$, where x specifies the initial or set amount of instances when the composite structure is created, and y

indicates the maximum amount of instances at any time. Parts are used to express composite structures, or modeling patterns that can be invoked by various objects to accomplish a specific purpose. When illustrating the composition of structures, parts can be embedded as properties of other parts. When embedded as [properties](#), parts can be bordered by a solid outline, indicating the surrounding part owns the part by composition. Alternatively, a dashed outline indicates that the property is referenced and used by the surrounding part, but not composed within it.

Common Usage



- [Composite Structure Diagram](#)

Further Information

- [Properties](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

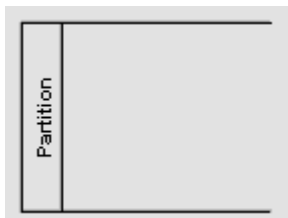
"An element representing a set of instances that are owned by a containing classifier instance or role of a classifier. (See role.) Parts may be joined by attached connectors and specify configurations of linked instances to be created within an instance of the containing classifier."

The OMG UML specification (*UML 2.0 Superstructure*, p. 14) also states:

A role is "the named set of features defined over a collection of entities participating in a particular context. Collaboration: The named set of behaviors possessed by a class or part participating in a particular context. Part: a subset of a particular class which exhibits a subset of features possessed by the class Associations: A synonym for association end often referring to a subset of classifier instances that are participating in the association."

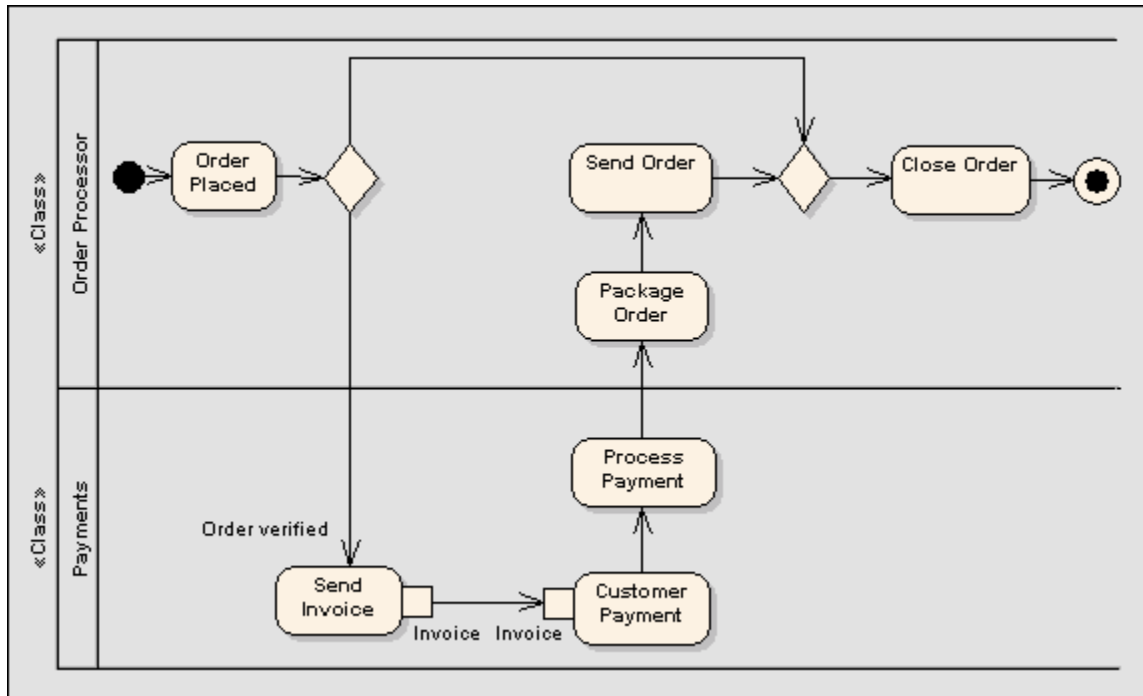
5.3.3.34 Partition

[Common Usage](#)..[Further Information](#)..[OMG UML Specification](#)



Activity *partitions* are used to logically organize an Activity diagram. They do not affect the token flow of an Activity diagram, but help structure the view or parts of it.

This example depicts the partitioning between the classes Payments and Order Processor.

**Common Usage**

- [Activity Diagram](#)

Partition

Further Information

- [Activities](#)
- [Activity Partition](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

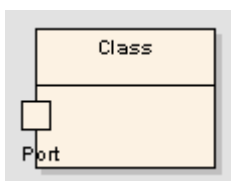
"A grouping of any set of model elements based on a set of criteria. 1. activity diagram: A grouping of activity nodes and edges. Partitions divide the nodes and edges to constrain and show a view of the contained nodes. Partitions can share contents. They often correspond to organizational units in a business model. They may be used to allocate characteristics or resources among the nodes of an activity."

The OMG UML specification (*UML 2.0 Superstructure*, p. 307) also states:

"An activity partition is a kind of activity group for identifying actions that have some characteristic in common."

5.3.3.35 Port

[Common Usage](#).. [Further Information](#).. [OMG UML Specification](#)



Ports define the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted by using the expose interface toolbox element. Any connector to a port must provide the required interface, if defined. Ports can appear on either a contained part, a class, or on the boundary of a composite structure.

A Port is a "typed" structural feature or property of its containing classifier.

Common Usage



- [Class Diagrams](#)
- [Object Diagrams](#)
- [Composite Structure Diagrams](#)

Further Information

- [Adding a Port to an Element](#)
- [Managing Inherited and Redefined Ports](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 167) states:

"A port is a structural feature of a classifier that specifies a distinct interaction point between that classifier and its environment or between the (behavior of the) classifier and its internal parts. Ports are connected to properties of the classifier by connectors through which requests can be made to invoke the behavioral features of a classifier. A Port may specify the services a classifier provides (offers) to its environment as well as the services that a classifier expects (requires) of its environment."

5.3.3.35.1 Adding a Port to an Element

[Further Information](#)

You can add a new Port to an element by:

1. Clicking on the Port symbol in Composite section of the UML toolbox and then dragging to or re-clicking on the target host element. This creates an untyped, simple port on the boundary, near where the mouse cursor was pointing.
2. Use the context menu of a suitable Class, Part or Composite object to *Insert Embedded Element* - select Port and a new Port is added at the mouse cursor position.
3. Drag a suitable classifier from the *Project View* onto a Class or Part. EA will prompt to add a typed Port or Part at the mouse cursor position. The new Port will be typed by the original dragged classifier.
4. Use the *Embedded Elements* dialog to add a new Port to the currently selected element.

Further Information

- [Port](#)
- [Managing Inherited and Redefined Ports](#)

5.3.3.35.2 Managing Inherited and Redefined Ports

[Further Information](#)

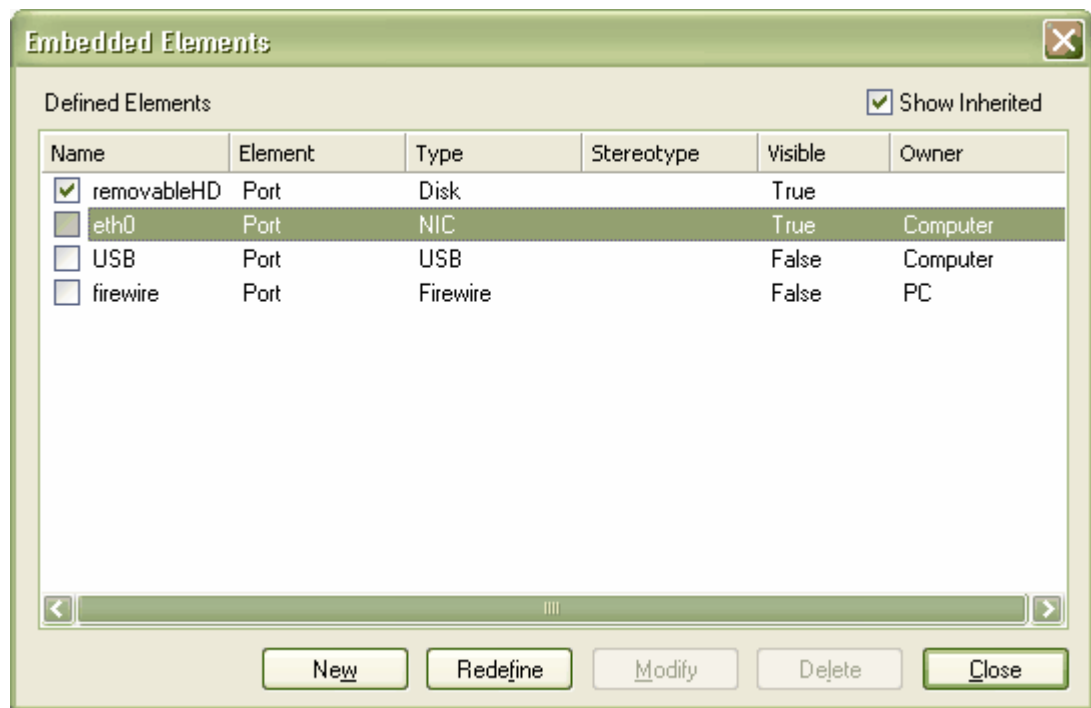
A Port is a **redefinable** and **re-useable** property of a composite classifier. So, like attributes, any class will inherit zero or more Ports from its parent and realized interfaces. If you have an inheritance hierarchy with Ports defined in the parent classes, when you open the *Embedded Elements* dialog, you will see the inherited

Ports and their named owners.

It is possible to expose for design purposes, an inherited Port (ie. the child class is re-using the parent Port). In this case, EA will create a clone of the re-used Port and mark it as read-only in the child class. This is convenient for modeling Port interactions in child classes where the Ports are defined in the Parents.

It is also possible to redefine a Port in a child class, such that the name is the same, but that the child is a modifiable clone of the original. This is useful where a child class will place additional restrictions or behavior on the Port. The *Embedded Elements* dialog allows you to highlight an inherited Port and mark it as "redefined", this create a new Port on the Child class which is editable, but still logically related to the initial Port.

The *Embedded Elements* diagram below illustrates Port inheritance. The Port "removableHD" is owned by the Child class. The Ports "eth0" and "USB" are owned by the "Computer" class. The "firewire" Port has been added to PC. If any of the inherited Ports are made visible, they are considered re-use Ports and will appear on the child in read-only format. By using the Redefine function, the inherited Port can be copied down and made writeable.



Further Information

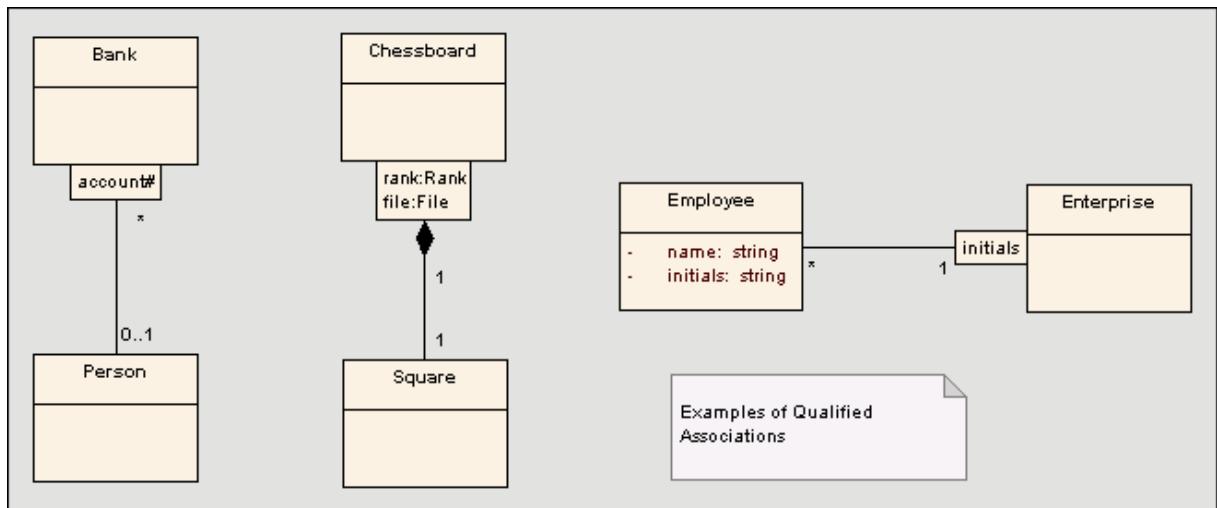
- [Port](#)
- [Adding a Port to an Element](#)

5.3.3.36 Qualifiers

OMG UML Specification

A *qualifier* is a property of an association which limits the nature of the relationship between two classifiers or objects.

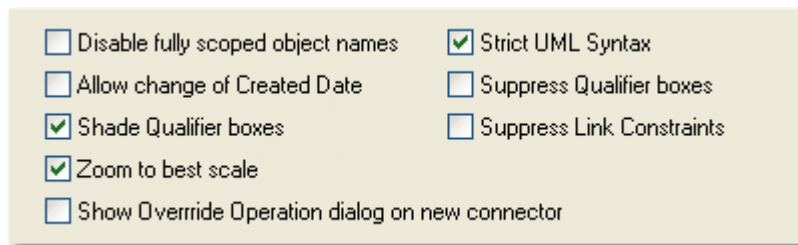
Some examples of qualified associations are given in this diagram:



Qualifiers are set in the **Source** and **Target Role** tabs of the **Association Properties** dialog.

Note: Separate multiple qualifiers with a semi-colon - each qualifier will then appear on a separate line. For example in the diagram above, the qualifier "rank:Rank;file:File" has been rendered in two lines, with a line break at the ; character.

Note: You can enable or disable a mild shading on the qualifier rectangles in the **Tools | Options | Diagram** page.



Note: You can enable/disable qualifier rectangles in the **Tools | Options | Diagram** page. If disabled, the old style text qualifiers are used. It is not recommended that you disable qualifiers as they are an integral part of the UML.

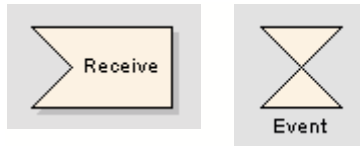
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 93) states:

"A qualifier declares a partition of the set of associated instances with respect to an instance at the qualified end (the qualified instance is at the end to which the qualifier is attached). A qualifier instance comprises one value for each qualifier attribute. Given a qualified object and a qualifier instance, the number of objects at the other end of the association is constrained by the declared multiplicity. In the common case in which the multiplicity is 0..1, the qualifier value is unique with respect to the qualified object, and designates at most one associated object. In the general case of multiplicity 0..*, the set of associated instances is partitioned into subsets, each selected by a given qualifier instance. In the case of multiplicity 1 or 0..1, the qualifier has both semantic and implementation consequences. In the case of multiplicity 0..*, it has no real semantic consequences but suggests an implementation that facilitates easy access of sets of associated instances linked by a given qualifier value."

5.3.3.37 Receive

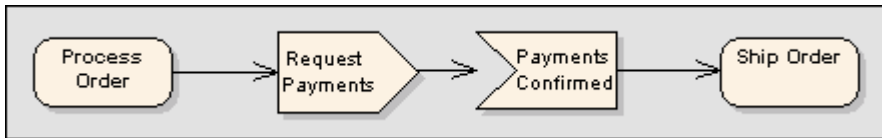
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *receive* element is used to define the acceptance or receipt of a request. Movement from a receive element occurs only once receipt is fulfilled according to its specification. The receive element comes in two forms:

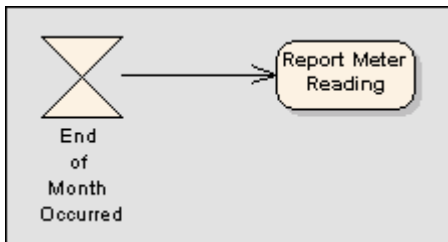
- Accept event action element
- Accept time event action element

The following example reflects a payment process on an order. Upon receiving the payment, the payment is confirmed and the flow continues to ship the order.



Refer to figure 158 (UML 2.0 Superstructure, p. 218).

To depict an accept time event, use the standard receive element from the toolbox. Right-click this element, and select *Accept Time Event*. The following example shows the hourglass-shaped accept time event action:



Refer to figure 159 (UML 2.0 Superstructure, p. 219).

Common Usage

- [Activity Diagrams](#)

Further Information

- [Send](#)
- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 218) states:

"AcceptEventAction is an action that waits for the occurrence of an event meeting specified conditions."

5.3.3.38 Region

[Common Usage](#).. [Further Information](#)



There are two types of *regions* supported. These are as follows:

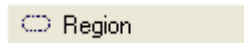
- [Expansion region](#)
- [Interruptible activity region](#)

After adding a region element to an activity diagram, the following prompt appears, from where the region type is selected.



Common Usage

- [Activity Diagram](#)

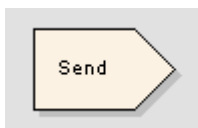


Further Information

- [Expansion Region](#)
- [Interruptible Activity Region](#)

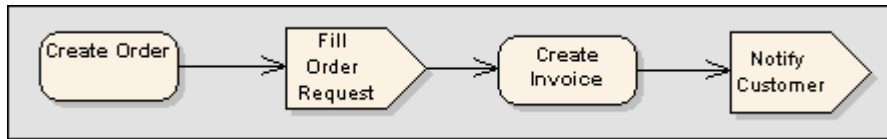
5.3.3.39 Send

[Common Usage](#).. [Further Information](#).. [OMG UML Specification](#)



The *send* element is used to depict the action of sending a signal.


The following example shows an order being processed, where a signal is sent to fill the order processed, and, upon creation of that order, a notification is sent to the customer.



Refer to figure 158 (UML 2.0 Superstructure, p. 218).

Common Usage

- [Activity Diagrams](#)

 Send

Further Information

- [Receive](#)
- [Activity Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 255) states:

"SendObjectAction is an action that transmits an object to the target object, where it may invoke behavior such as the firing of state machine transitions or the execution of an activity. The value of the object is available to the execution of invoked behaviors. The requestor continues execution immediately. Any reply message is ignored and is not transmitted to the requestor."

5.3.3.40 State

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)




A state represents a situation where some invariant condition holds; this condition can be static, ie. waiting for an event, or dynamic, ie. performing a set of activities. State modeling is usually related to classes, and describes the allowable states a class or element may be in and the transitions that allow the element to move there. There are three types of states: simple states, composite states and submachine states.

Furthermore, there are pseudo-states, resembling some aspect of a state, but with a pre-defined implication. Pseudo-states are used to model complex transitional paths, and classify common state machine behavior.

Common Usage

- [State Machine Diagram](#)

 State

Further Information

- [Composite State](#)
- [Submachine State](#)
- [Pseudo-States](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 477) states:

"A state models a situation during which some (usually implicit) invariant condition holds. The invariant may represent a static situation such as an object waiting for some external event to occur. However, it

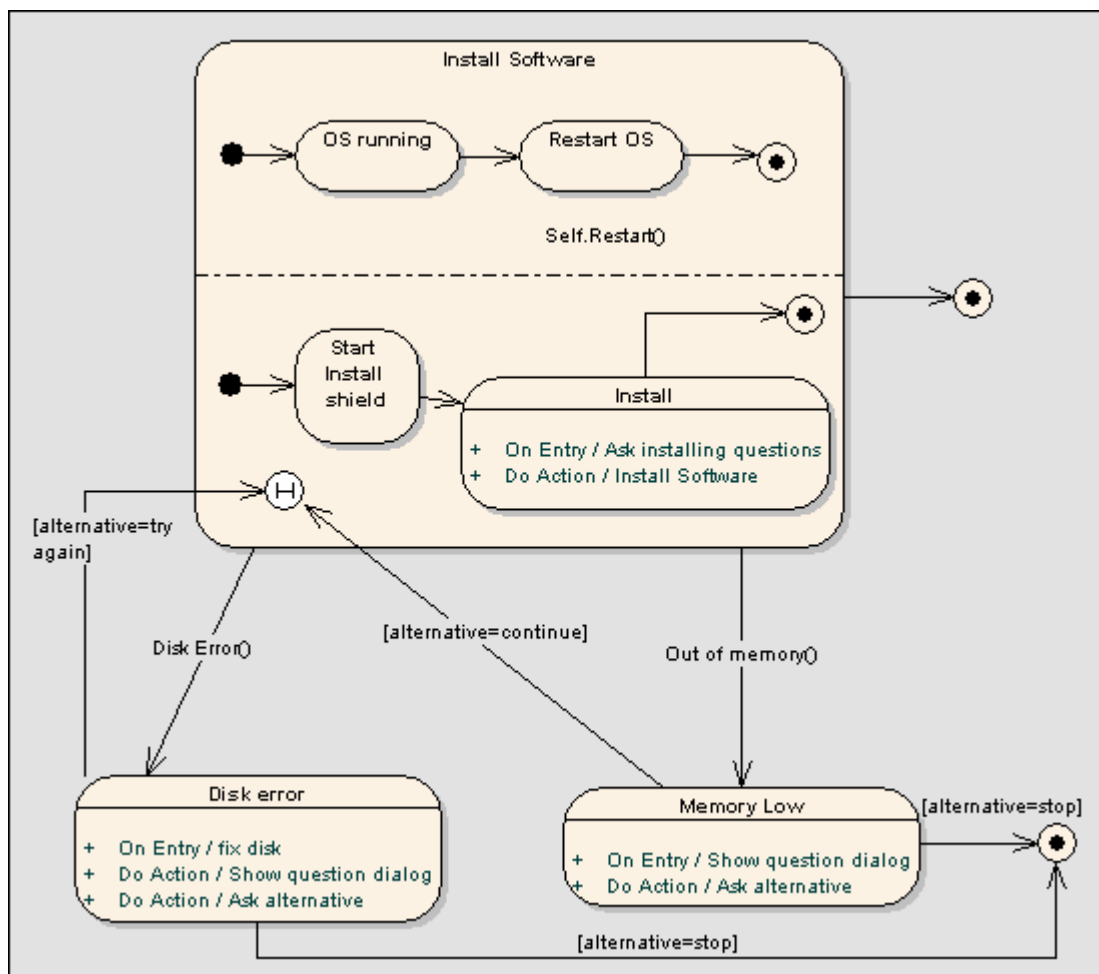
can also model dynamic conditions such as the process of performing some activity (i.e., the model element under consideration enters the state when the activity commences and leaves it as soon as the activity is completed)."

5.3.3.40.1 Composite State

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)

Composite states are semantically equivalent to submachine states, but they cannot be reused as can submachine states. They are composed inline the state machine, by expanding a state, adding regions if applicable, and designing the appropriate composite state with internal states referred to as sub-states.

Composite states can be orthogonal, if regions are created. If a composite state is orthogonal, its entry denotes that a single sub-state is concurrently active in all regions. The hierarchical nesting of composite states, coupled with region usage, generates a situation of multiple states concurrently active; this situation is referred to as the active state configuration.



Common Usage

- [State Machine Diagram](#)

Further Information

- [Regions](#)
- [Submachine State](#)
- [State](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 478) states:

"A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions. A given state may only be decomposed in one of these two ways.

"Any state enclosed within a region of a composite state is called a substate of that composite state. It is called a direct substate when it is not contained by any other state; otherwise it is referred to as an indirect substate.

"Each region of a composite state may have an initial pseudostate and a final state. A transition to the enclosing state represents a transition to the initial pseudostate in each region. A newly-created object takes its topmost default transitions, originating from the topmost initial pseudostates of each region.

"A transition to a final state represents the completion of activity in the enclosing region. Completion of activity in all orthogonal regions represents completion of activity by the enclosing state and triggers a completion event on the enclosing state. Completion of the topmost regions of an object corresponds to its termination."

5.3.3.41 State Lifeline

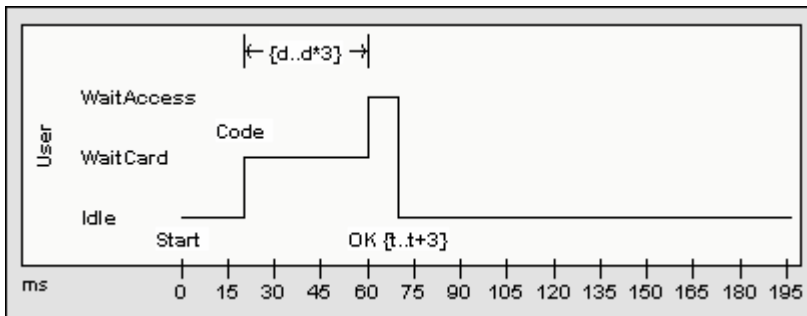
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *lifeline* is the path an object takes across a measure of time, as indicated by the x-axis.

A *state lifeline* follows discrete transitions between states, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations.

An example of a state lifeline is given in this diagram:



Refer to figure 350 (UML 2.0 Superstructure, p. 452).

A state lifeline consists of a set of transition points. Each transition point can be defined with the following properties:

At time	Specifies the starting time for a change of state.
Transition to	Indicates the state to which the lifeline will change.
Event	Describes the occurring event.
Timing constraints	Refers to the time taken for a state to change within a lifeline, or the time taken to transmit a message (ie. $t..t+3$).
Timing observations	Provides information on the time of a state change or sent message.
Duration constraints	Pertains to a lifeline's period at a particular state. The constraint could be instigated by a change of state within a lifeline, or that lifeline's receipt of a message.
Duration observations	Indicates the interval of a lifeline at a particular state, begun from a change in state or message receipt.

An example of this from the diagram above would be that the **OK** transition point has the properties:

At Time	70ms
Transition to	200ms
Event	OK
Timing constraints	$t..t+3$
Timing observations	—
Duration constraints	—
Duration observations	—

Common Usage

- [Timing Diagram](#)

 State Lifeline

Further Information

- [Value Lifeline](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 451) states:

"This is the state of the classifier or attribute, or some testable condition, such as an discrete enumerable value. See also 'StateInvariant (from BasicInteractions)' on page 433.

"It is also permissible to let the state-dimension be continuous as well as discrete. This is illustrative for scenarios where certain entities undergo continuous state changes, such as temperature or density."

5.3.3.42 State/Continuation

[Common Usage](#)..|..[Further Information](#).



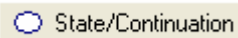
The *state/continuation* symbol serves two different purposes for interaction diagrams, as state invariants and continuations.

Common Usage

- [Sequence Diagrams](#)

Further Information

- [State Invariants](#)
- [Continuations](#)

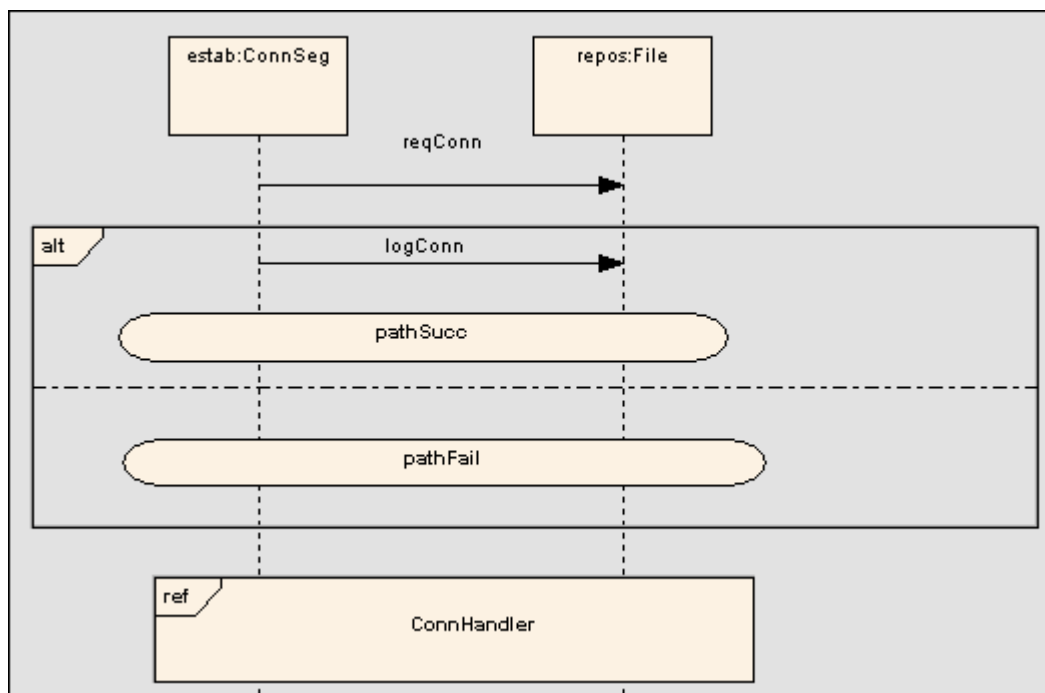


5.3.3.42.1 Continuation

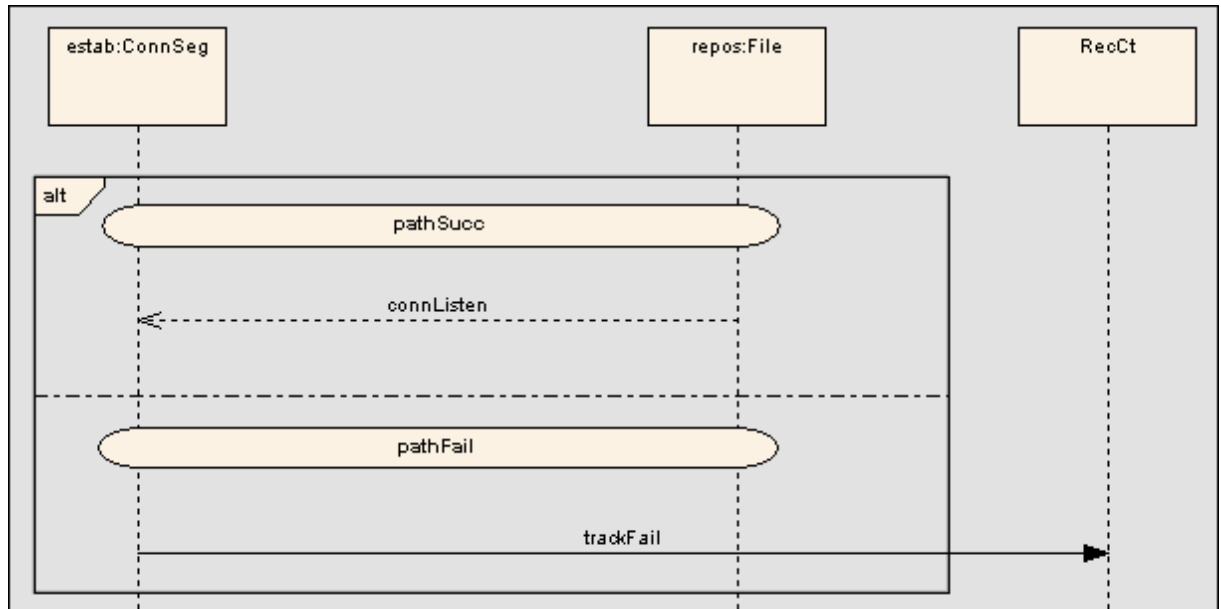
[Further Information](#)..|..[OMG UML Specification](#)

A *continuation* is:

- Used in seq and alt combined fragments, to indicate the branches of continuation an operand follows.
- To indicate a continuation, end an operand with a continuation, and indicate the continuation branch with matching continuation (same name) preceding the interaction fragment.
- For the following continuation example, an alt combined fragment has continuations, pathSucc and pathFail. These continuations are located within the interaction occurrence ConnHandler, which has subsequent events based on the continuation.



Below is the interaction referenced by the InteractionOccurrence.



Further Information

- [State/Continuation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 414) states:

"A Continuation is a syntactic way to define continuations of different branches of an Alternative CombinedFragment. Continuations is intuitively similar to labels representing intermediate points in a flow of control."

The OMG UML specification (*UML 2.0 Superstructure*, p. 415) states:

"Continuations have semantics only in connection with Alternative CombinedFragments and (weak) sequencing. If an InteractionOperand of an Alternative CombinedFragment ends in a Continuation with name (say) X, only InteractionFragments starting with the Continuation X (or no continuation at all) can be appended."

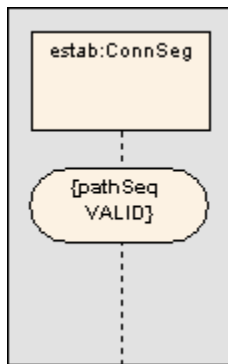
5.3.3.42.2 State Invariant

[Further Information](#).. [OMG UML Specification](#)

A *state invariant* is:

- A condition applied to a lifeline, which must be fulfilled for the lifeline to exist.

A state invariant is exemplified below.



Further Information

- [State/Continuation](#)

OMG UML Specification

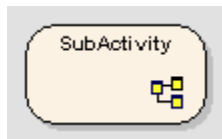
The OMG UML specification (*UML 2.0 Superstructure*, p. 433) states:

"A *StateInvariant* is a constraint on the state of a *Lifeline*. In this case we mean by "state" also the values of eventual attributes of the *Lifeline*.

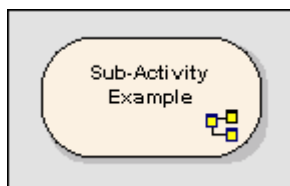
"A *StateInvariant* is an *InteractionFragment* and it is placed on a *Lifeline*."

5.3.3.43 SubActivity

[Common Usage..](#)



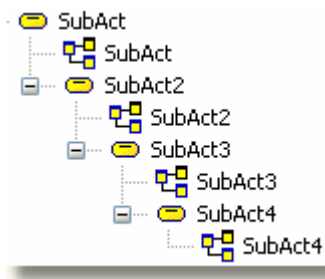
You can add a *subactivity* element to an Activity diagram. A subactivity element is a pointer to a child Activity diagram.




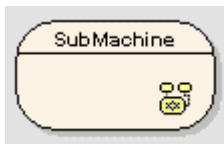
To create a subactivity, click on the subactivity element in the Activity group in the toolbox, then click on the diagram. Subactivity elements have a small yellow diagram on the bottom right-hand corner.

To access the Activity diagram represented by the subactivity element, double click the subactivity element and the diagram will open.

When you create nested subactivity elements, they are shown as nested in the Project Browser - see the example below.

**Common Usage**

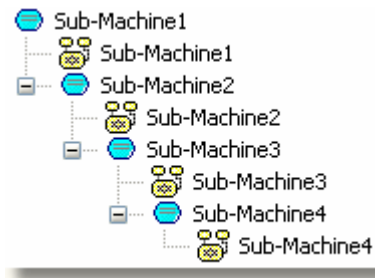
- [Activity Diagrams](#)

 Subactivity
5.3.3.44 SubMachine**Common Usage**


You can add a *sub-machine* element to a State Machine diagram. A sub-machine element is a pointer to a child State Machine diagram.

To create a sub-machine, click on the sub-machine element in the State group in the toolbox, then click on the diagram. Sub-machine elements have a small yellow diagram on the bottom right-hand corner. To access the State Machine diagram represented by the sub-machine element, double click the sub-machine element and the diagram will open.

When you create nested sub-machine elements, they are shown as nested in the Project Browser - see the example below.

**Common Usage**

- [State Machine Diagram](#)

 Sub-Machine

5.3.3.45 *Synch*

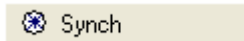
[Common Usage](#)..|..[OMG UML Specification](#)



A *synch* state is useful for indicating that concurrent paths of a state machine will be synchronized. After bringing the paths to a synch state, the emerging transition will indicate unison.

Common Usage

- [State Machine Diagram](#)



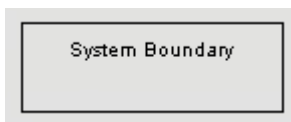
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 16) states:

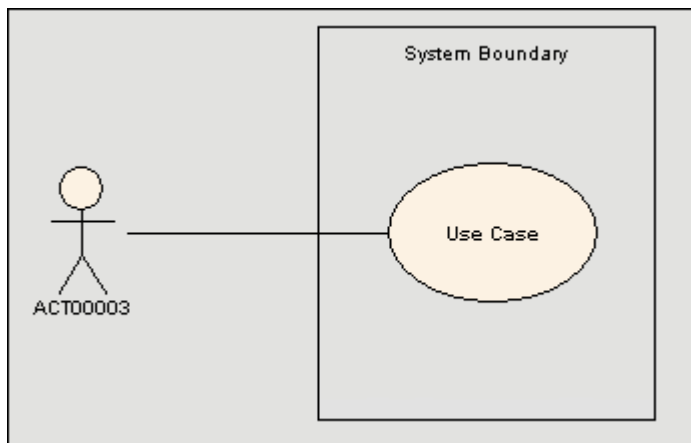
"A vertex in a state machine used for synchronizing the concurrent regions of a state machine."

5.3.3.46 *System Boundary*

[Common Usage](#)..|..[OMG UML Specification](#)

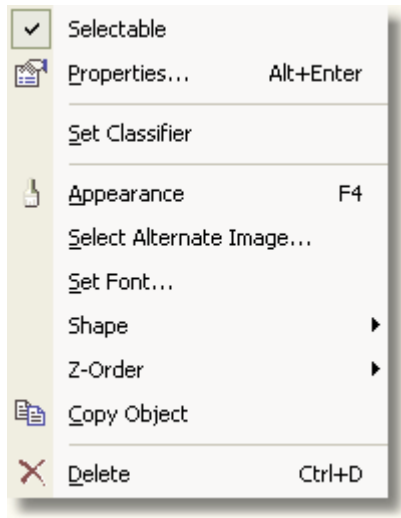


A *system boundary* element signifies a classifier, such as a class, component or sub-system, to which the enclosed use cases are applied. By depicting a boundary, its referenced classifier does not reflect ownership of the embodied use cases, but instead indicates usage.



A boundary element may be marked as 'Selectable' or not. When not selectable, you can click within the boundary space without activating or selecting the boundary itself. This is useful when you have many

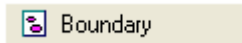
elements within the boundary and selection of them is being made difficult by the boundary itself.



Note: A boundary may have an associated image which it will display instead of its default format ... use the *Select Alternate Image...* item to select an image.

Common Usage

- [Use Case](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 520) states:

"If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier."

5.3.3.47 Terminate

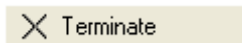
[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



The *terminate* pseudostate indicates that upon entry of its pseudostate, the state machine's execution will end.

Common Usage

- [State Machine Diagram](#)



Further Information

- [Pseudo-States](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 471) states:

"Entering a terminate pseudostate implies that the execution of this state machine by means of its context object is terminated."

5.3.3.48 Use Case

[Common Usage](#) | [OMG UML Specification](#)

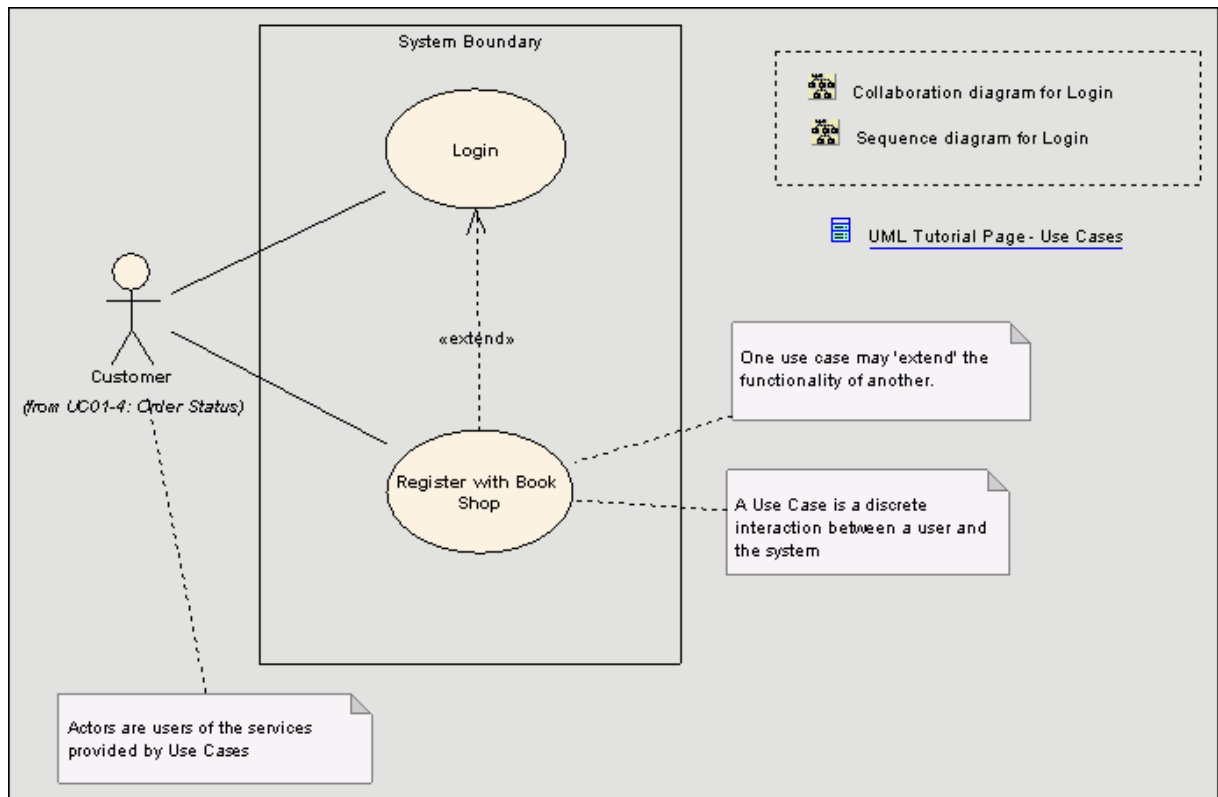


A *use case* is a UML modeling element that describes how a user of the proposed system will interact with the system to perform a discrete unit of work. It describes and signifies a single interaction over time that has meaning for the end user (person, machine or other system), and is required to leave the system in a complete state: either the interaction completed or was rolled back to the initial state.

- A use case typically has requirements and constraints that describe the essential features and rules under which it operates.
- A use case may have an associated Sequence diagram illustrating behavior over time - who does what and to whom, when.
- A use case typically has scenarios associated with it that describe the work flow over time that produces the end result. Alternate work flows (to capture exceptions, etc.) are also allowed.

Tip: Use a Use Case diagram and model to build up the functional requirements and implementation details of the system.

Below is an example Use Case model:



Common Usage

- [Use Case Diagram](#)

○ Use Case

Further Information

- [Use Case Extension Points](#)
- [Using Rectangle Notation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

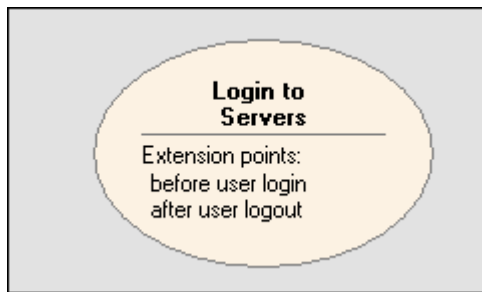
"The specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system."

5.3.3.48.1 Use Case Extension Points

Further Information

Use *extension points* to specify the point of an extended use case where an extending use case's behavior should be inserted. The specification text can be informal or precise to define the location of the extension point.

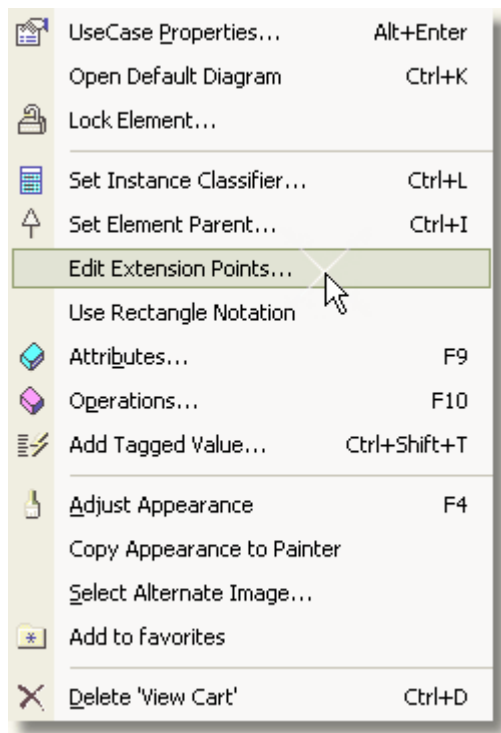
Note that conditions to apply that extending use case and the extension point to use should be attached as a note to the extend relationship.



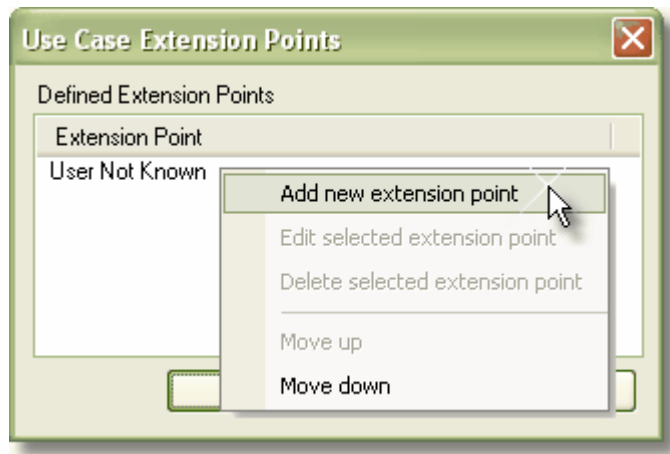
Extension Points

To work with extension points, follow these steps:

1. Right click on the use case element to view the context menu.
2. Select *Edit Extension Points*.



3. The *Extension Points dialog* will appear - listing defined points for that use case.
4. Right click in the list to access the create, edit, delete and move functions.



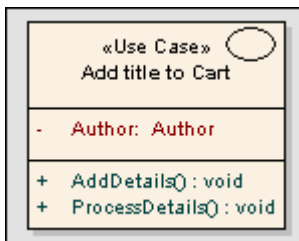
Further Information

- [Use Case](#)

5.3.3.48.2 Using Rectangle Notation

[Further Information](#)

You can display a use case using rectangle notation - this displays the use case in a rectangle, with an oval in the top right-hand corner. Any attributes, operations, constraints, etc. belonging to the use case are shown, in the same style as a class.



To show a use case using rectangle notation, right click on the use case object on the diagram and select *Use Rectangle Notation*. This setting will only apply to the selected use case, and can be toggled on and off.

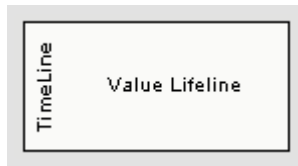
Tip: Actor elements can also be displayed using rectangle notation.

Further Information

- [Use Case](#)

5.3.3.49 Value Lifeline

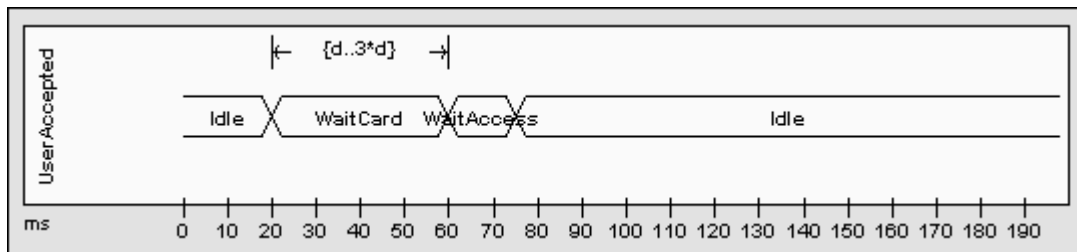
[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



A *lifeline* is the path an object takes across a measure of time, indicated by the x-axis.

The *value lifeline* shows the lifeline's state across the diagram, within parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

An example of a value lifeline is shown in the below diagram:



Refer to figure 351 (UML 2.0 Superstructure, p. 453).

The value lifeline consists of a set of transition points. Each transition point can be defined with the following properties:

At time	Specifies the starting time for a change of state.
Transition to	Indicates the state to which the lifeline will change.
Event	Describes the occurring event.
Timing constraints	Refers to the time taken for a state to change within a lifeline, or the time taken to transmit a message.
Timing observations	Provides information on the time of a state change or sent message.
Duration constraints	Pertains to a lifeline's period at a particular state. The constraint could be instigated by a change of state within a lifeline, or that lifeline's receipt of a message.
Duration observations	Indicates the interval of a lifeline at a particular state, begun from a change in state or message receipt.

An example of this from the diagram above would be that the "OK" transition point has the properties:

At Time	20ms
Transition to	60ms
Event	WaitCard
Timing constraints	—
Timing observations	—
Duration constraints	d..3*d
Duration observations	—

Common Usage

- [Timing Diagram](#)

A small rectangular icon with a light beige background and a thin black border. Inside the rectangle, on the left, is a small blue square containing a white icon of a value lifeline (a vertical line with a horizontal bar at the top). To the right of the icon, the text "Value Lifeline" is written in a black, sans-serif font.**Further Information**

- [State Lifeline](#)
- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 451) states:

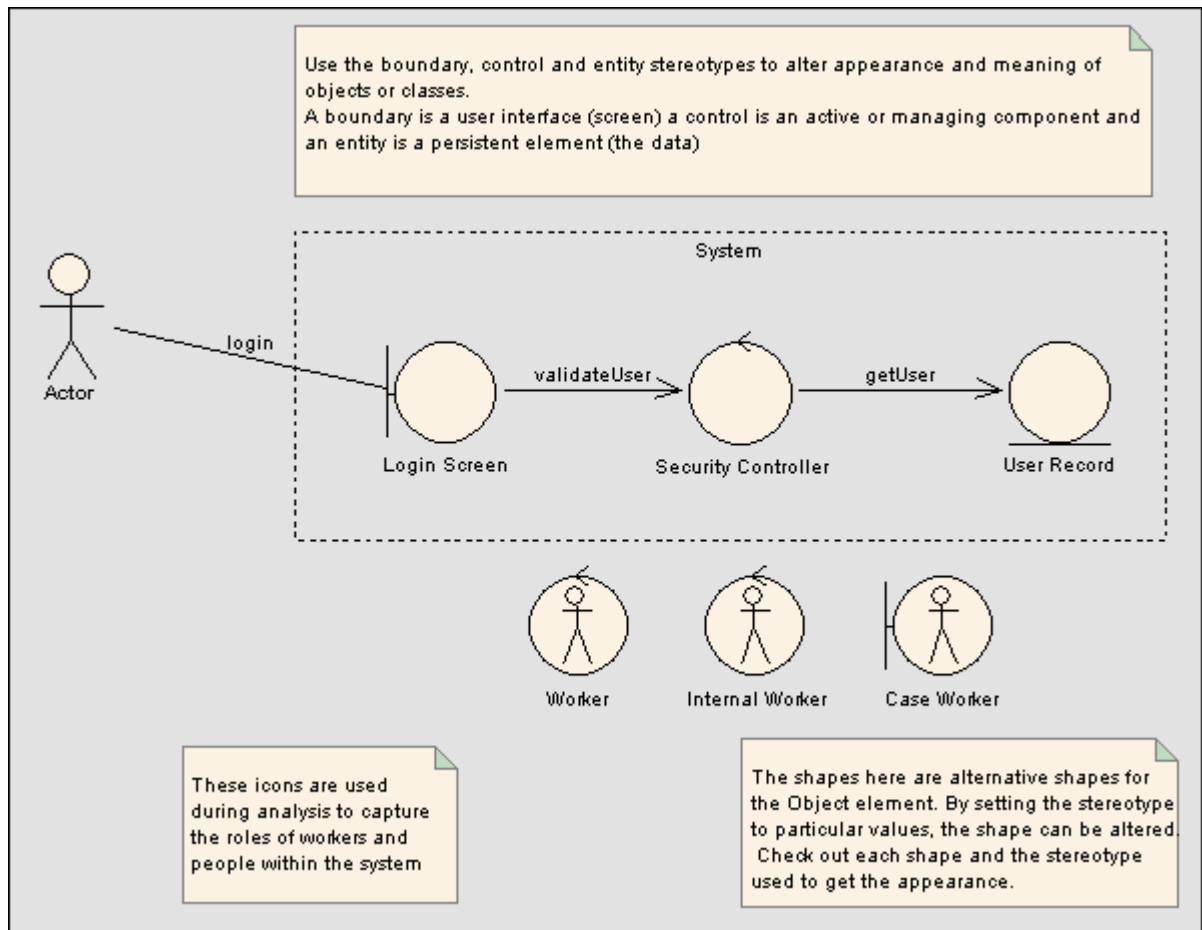
"Shows the value of the connectable element as a function of time. Value is explicitly denoted as text. Crossing reflects the event where the value changed."

5.3.4 Inbuilt and Extended Stereotypes

There are many other elements in UML which you can also work with in Enterprise Architect. This topic gives a brief introduction to some of these elements.

5.3.4.1 Analysis Stereotypes

EA has some built in stereotypes which you can assign to an element during analysis. The effect of these stereotypes is to display a different icon than the normal element icon, providing a visual key to the element purpose. The diagram below illustrates the main types of inbuilt icons for elements:

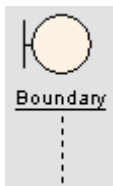


The stereotypes used are:

- *boundary* - for a system boundary (eg. a Login screen)
- *control* - to specify an element is a controller of some process (as in Model View Controller pattern)
- *entity* - the element is a persistent or data element
- *worker*, *caseworker* and *internal worker* - denotes specific roles in the analysis based on RUP and Robustness analysis guidelines

5.3.4.2 Boundary

Common Usage




A *boundary* is a stereotyped class that models some system boundary - typically a user interface screen. It is used in the conceptual phase to capture users interacting with the system at a screen level (or some other boundary interface type). It is often used in Sequence and Robustness (Analysis) diagrams. It is the View in

the Model-View-Controller pattern.

Tip: Use boundary elements in analysis to capture user interactions, screen flows and element interactions (or 'collaborations').

Common Usage

- [Sequence Diagram](#)
- [Analysis Diagram](#)
- [Robustness Diagram](#)

 Boundary

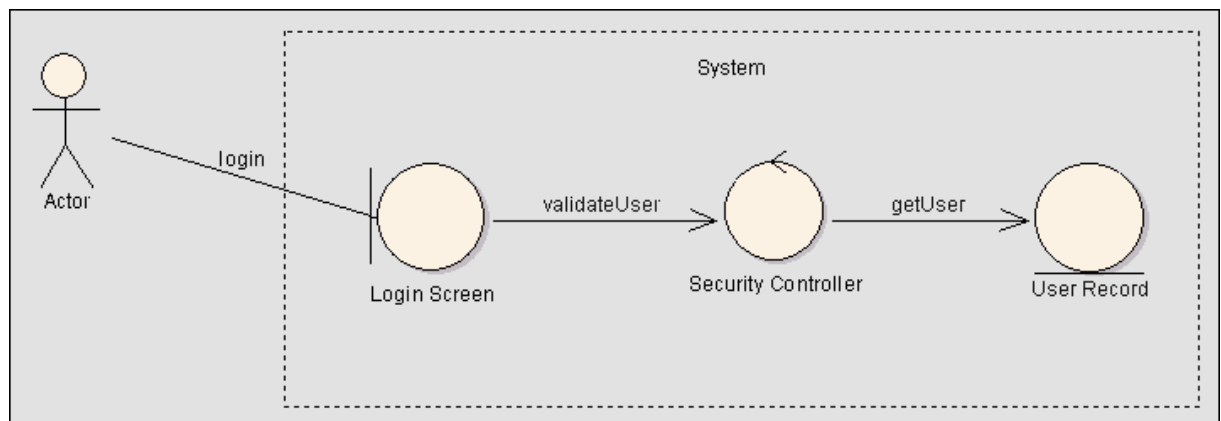
5.3.4.2.1 Create a Boundary

Further Information

To create a boundary element, follow the steps below:

1. Insert a new class.
2. Open the *Properties* dialog.
3. Set the *Stereotype* to 'boundary'.
4. Save changes.

The illustration below shows an actor interacting with a Login screen.

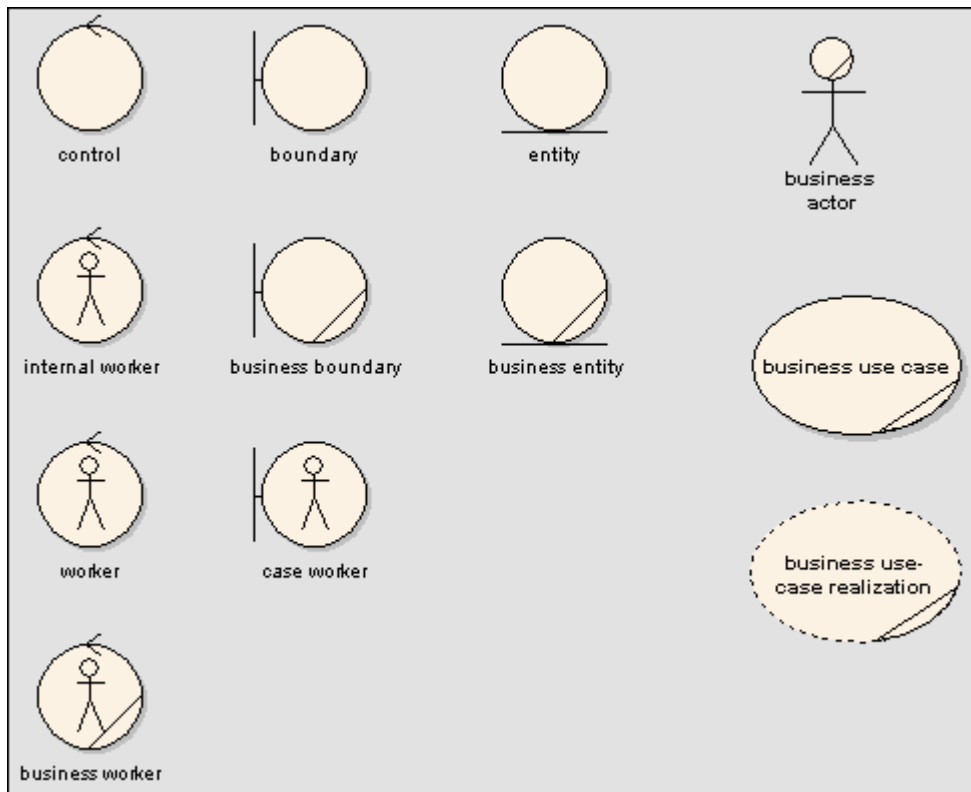


Further Information

- [Boundary](#)

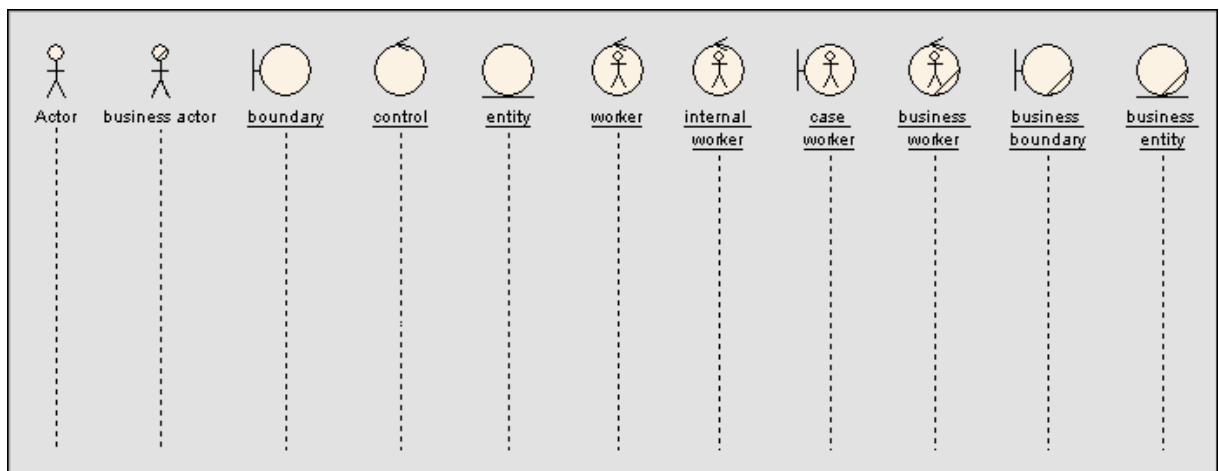
5.3.4.3 Business Modeling Stereotypes 1

This diagram shows the range of inbuilt stereotyped icons for business modeling. These include stereotypes for classes and objects - as well as one of actors and two for use cases. The name given to each graphical element in the diagram is that of the stereotype to apply.



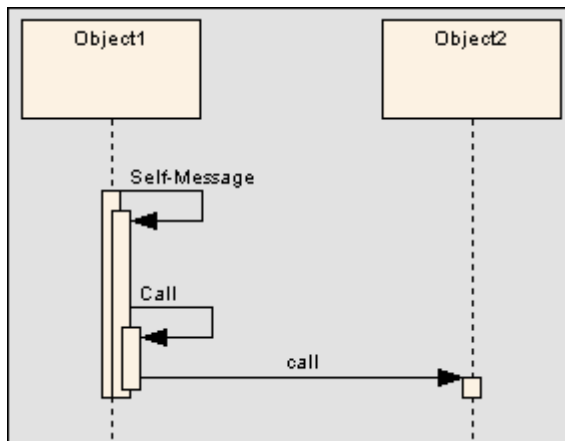
5.3.4.4 Business Modeling Stereotypes 2

This diagram shows the inbuilt stereotype icons for business modeling with sequence diagrams. The name given to each graphical element in the diagram is that of the stereotype to apply.



5.3.4.5 Call

[Common Usage](#)...[Further Information](#)



A *call* is a type of message element that extends the level of activation from the previous message. All self-messages create a new activation level, but this focus of control usually ends with the next message (unless [activation levels](#) are manually adjusted). Self-message calls, as depicted above by the first call, indicate a nested invocation; new activation levels are added with each call. Unlike a regular message between elements, a call between elements continues the existing activation in the source element, implying that the call was initiated within the previous message's activation scope.

Common Usage

- [Sequence Diagram](#)



Further Information

- [Message \(Sequence Diagram\)](#)
- [Sequence Element Activation](#)
- [Lifeline Activation Levels](#)

5.3.4.6 Composite Elements

Enterprise Architect supports *composite elements* for classes, objects, use cases etc. A composite element is a pointer to a child diagram.

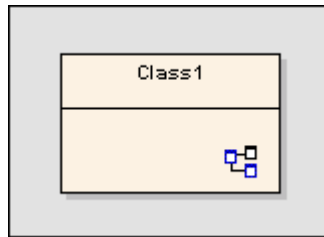
Creating a Composite Element

Composite elements can be set from the element context menu. Follow the instructions below:

1. Create the element you wish to set as a composite element.
2. Right click on the element in the diagram and select *Set as Composite Element* from the context menu.

Note: If *Set as Composite Element* does not appear in the element context menu, the option is not available for the type of element you have selected.

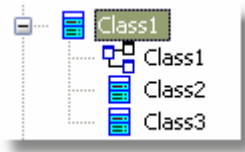
The element will appear as follows:



Note the small icon in the bottom right hand corner representing that this is now a composite element.

3. Double click the composite element to access the child diagram that it points to.

The composite element and its child diagram are represented in the Project Browser as follows:



Note that Class2 and Class3 are elements in the child diagram.

The Automation Interface

Automation support is available for composite elements - Element has an Elements collection and a Diagrams collection.

5.3.4.7 Control

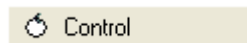
Common Usage



A *control* is a stereotyped class that represents a controlling entity or manager. A control organizes and schedules other activities and elements. It is the controller of the Model-View-Controller pattern.

Common Usage

- [Sequence Diagram](#)



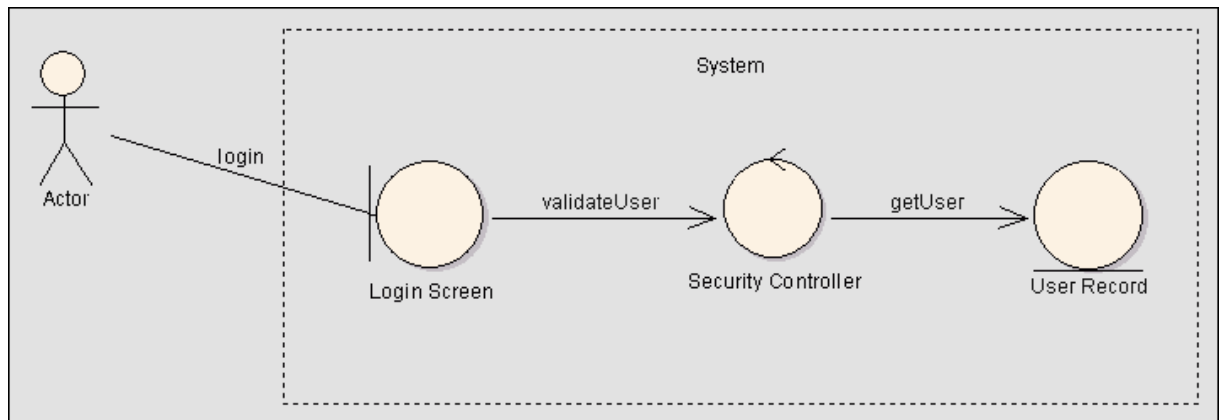
5.3.4.7.1 Create a Control Element

Further Information

To create a control element, follow the steps below:

1. Insert a new class.
2. Bring up the *Properties* dialog.
3. Set the *Stereotype* to 'control'.
4. Save changes.

The appearance will change to that illustrated in the diagram below (for the security controller element):



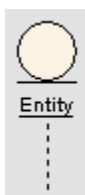
Note: The Model-View-Controller (MVC) pattern is a design pattern for building a wide range of applications that have a user interface, business or application logic and persistent data.

Further Information

- [Control](#)

5.3.4.8 Entity

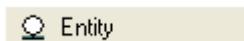
Common Usage



An *entity* is a store or persistence mechanism that captures the information or knowledge in a system. It is the Model in the Model-View-Controller pattern.

Common Usage

- [Sequence Diagram](#)



5.3.4.8.1 Create an Entity

[Further Information](#)

To create an entity, follow the steps below:

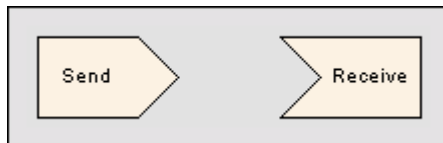
1. Insert a class from the UML toolbox.
2. Open the *Properties* dialog.
3. Set the element *Stereotype* to 'entity'.
4. Save changes.

Further Information

- [Entity](#)

5.3.4.9 Event

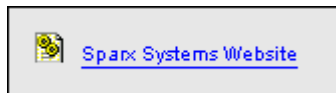
The UML includes two elements that are used to model events. The first element is the *receive event* and is depicted as a rectangle with a recessed 'V' on the left side. This element indicates that an event occurs in the system due to some external or internal stimulus. Typically this will invoke further activities and processing.



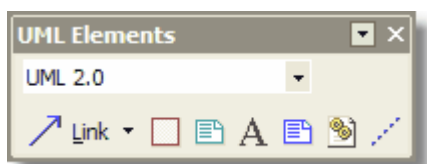
The second element is the *send event*. This element models the generation of a stimulus in the system and the passing of that stimulus to other elements within the system or external to the system.

Send and receive events can be added from the Analysis and Activity sections of the UML toolbox.

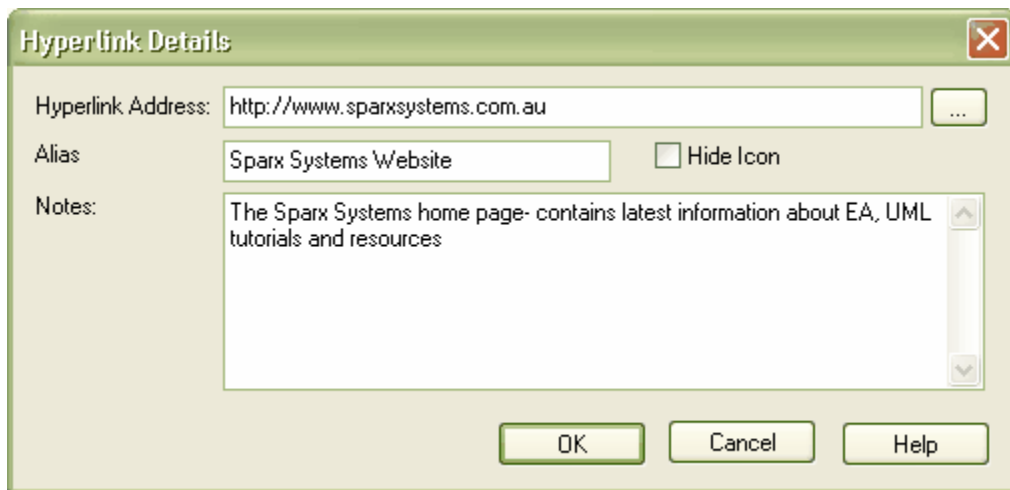
5.3.4.10 Hyperlinks



You can place a *hyperlink* element onto a diagram. This element is a type of text element - but one that may contain a pointer to a file or web address. When double clicked on, EA will attempt to execute the related file or address. You can connect diagrams to associated files, web pages and even other EA model files. You can add a hyperlink using the *Elements* toolbar

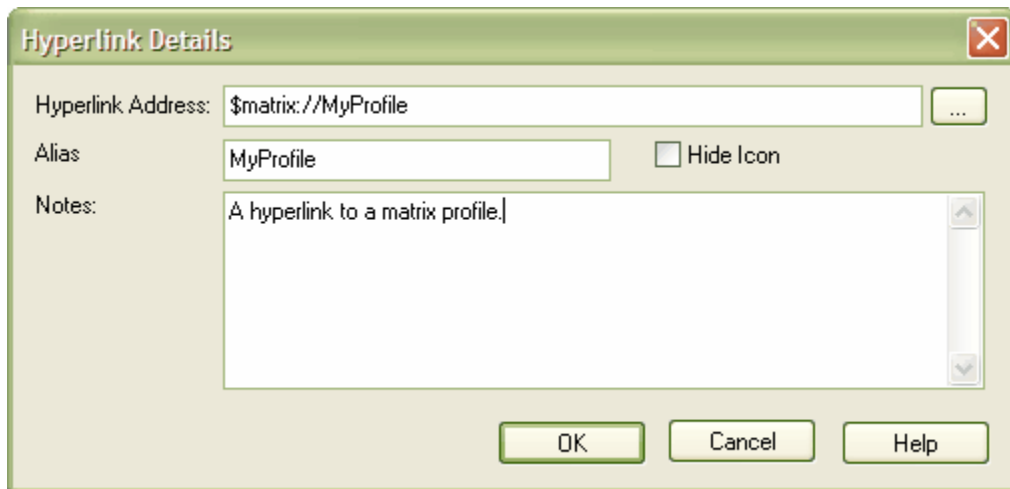


Configuring the hyperlink:



Creating Hyperlinks to Matrix Profiles

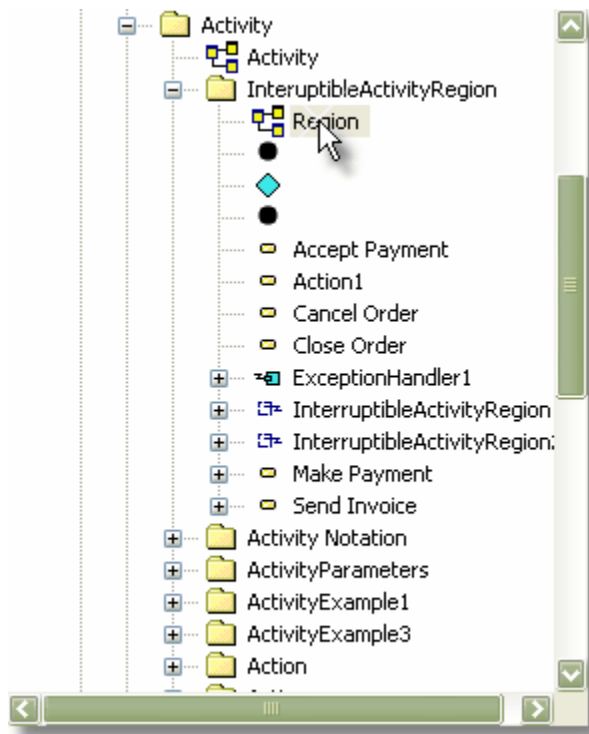
Hyperlinks may also be added with a Matrix Profile as its target from a hyperlink object in a diagram. To achieve this Use \$matrix:// as the target prefix followed by the name of the profile (e.g. \$matrix://MyProfile).



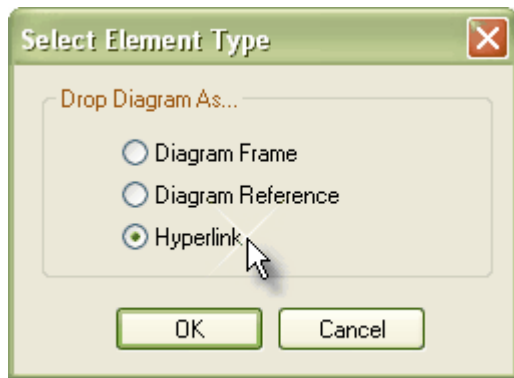
Creating Hyperlinks between diagrams

You can create hyperlinks between diagrams by following the steps detailed below:

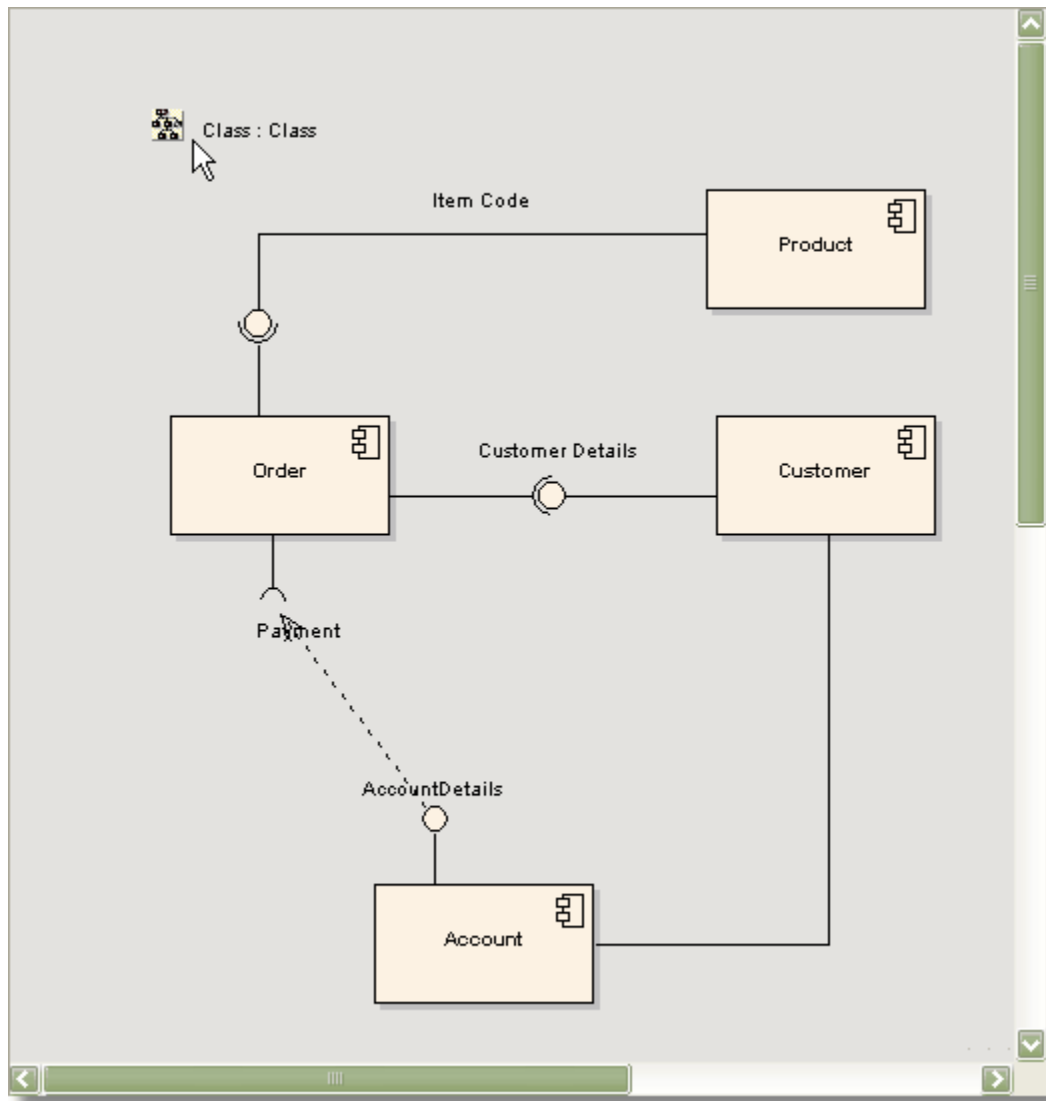
1. Open the diagram that you want to display a hyperlink to another diagram. From the project view select and drag the diagram you want to create a hyperlink.



- Next, drag the diagram on to the current diagram and select the option *Hyperlink*.

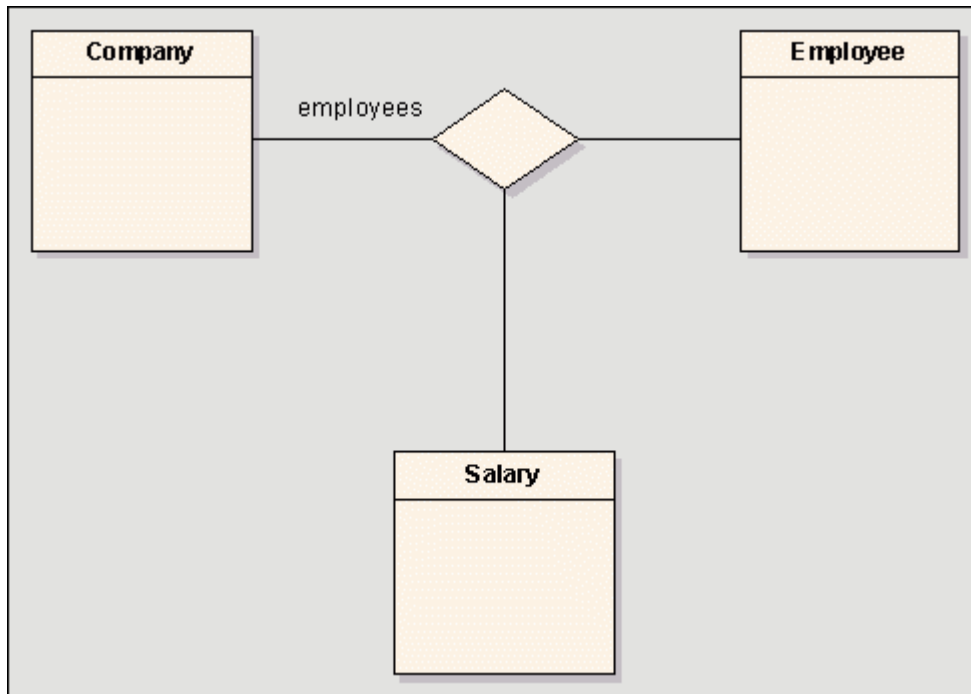


- The final hyperlinked diagram should look like the diagram below:



Note: If the hyperlinks are appearing as sub activities go to the [Tools](#) | [Options](#) | [Diagram](#) | [Behavior](#) and uncheck the [Use Automatic SubActivities](#) check box.

5.3.4.11 N-Ary Association



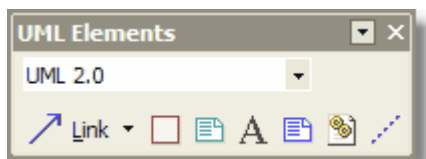
An *n*-Ary association element is used to model complex relationships between three (or more) elements. It is not a commonly employed device, but can be used to good effect where there is a dependant relationship between more than two elements.

In the example above there is a relationship between a Company and an Employee and a Salary.

5.3.4.12 Other Elements

In addition to the standard UML elements, you can also add notes, boundaries, text, diagram properties and hyperlinks to your diagram.

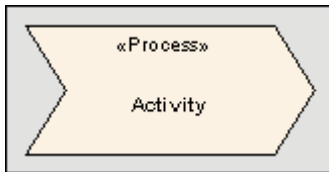
To add these, use the buttons in the [shortcut toolbar](#). The first inserts a system boundary, the second a note and the third a text element.



The dotted line is a note link, and can be used to link notes to particular elements.

These elements are simple comments and delimiters and have no structural role in the model. They are very useful in explaining and grouping other elements.

5.3.4.13 Process

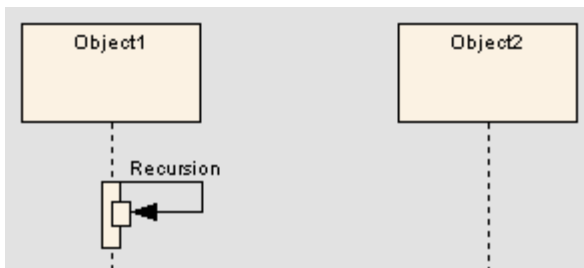


A *process* is a stereotyped activity element which expresses the concept of a business process. Typically this involves inputs, outputs, work flows, goals and connections with other processes.

Business processes typically range across many parts of the organization and span one or more systems.

5.3.4.14 Recursion

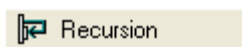
[Common Usage...](#) [Further Information](#)



A *recursion* is a type of message used in sequence diagrams to indicate a recursive function.

Common Usage

- [Sequence Diagram](#)



Further Information

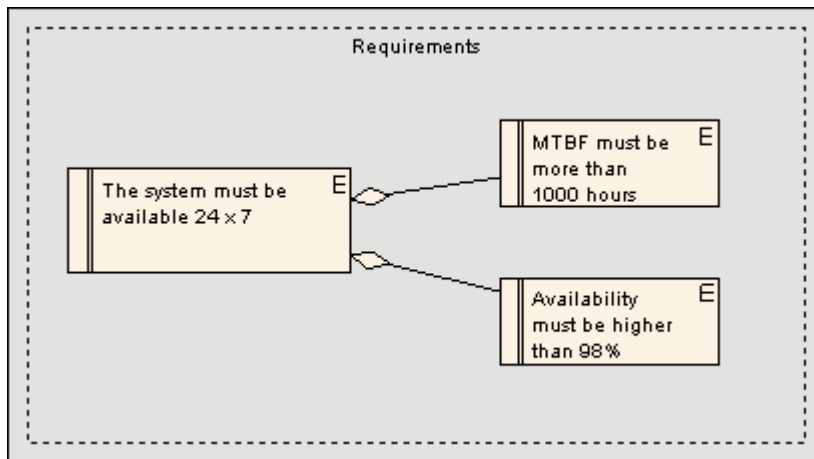
- [Message \(Sequence Diagram\)](#)

5.3.4.15 Requirements

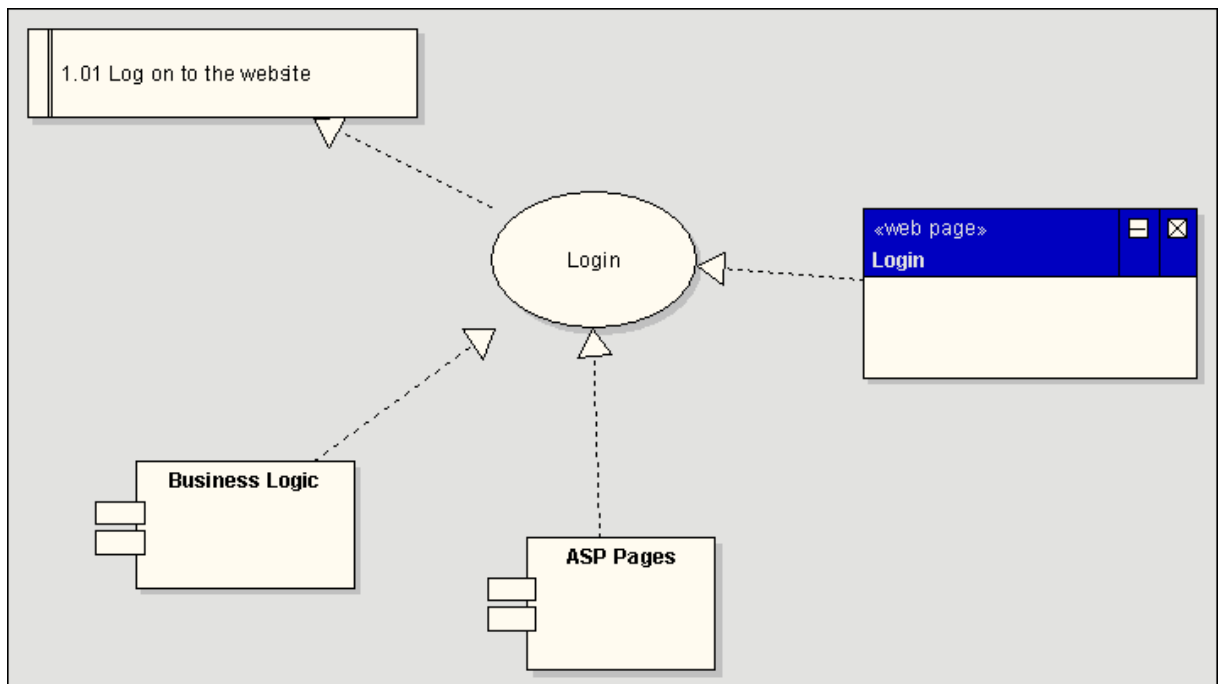
As an analysis step, often it is desirable to capture simple system *requirements*. These will eventually be realized by use cases.

In the initial requirement gathering phase, cataloging requirements may be achieved using the custom requirement extension.

Requirements may also be aggregated to create a hierarchy.



The diagram below illustrates how this might be done. A requirement that a user can log into a website is implemented (dependency link) by the 'Login' use case, which in turn is implemented by the Business Logic, ASP pages and Login Web Page. Using this approach, you can model quite detailed and complex dependencies and implementation relationships with ease.



5.3.4.16 Screen

A *screen* is a stereotyped class representing a User Interface screen. It is for prototyping and screen flow purposes only. By using UML features such as requirements, constraints and scenarios against user interface elements, you can build up a solid and detailed understanding of user interface behavior without having to use code. This becomes an excellent means of establishing the precise behavior the system will have from a user perspective, and in conjunction with the use case model, defines exactly how a user will get work done.

Web Pages may also be prototyped and specified rigorously using EA's custom interface extensions.

The example diagram below illustrates some features of EA's screen modeling extensions that support web page prototyping. By adding requirements, rules, scenarios and notes to each element, a detailed model is built up of the form or web page, without having to resort to GUI builders or HTML.

Note: EA displays a selection of stereotyped UI Controls with special icons - for example a control stereotyped as a <<List>> will display with a vertical scroll bar.

Stereotypes and their Effects

The following details the stereotypes you can add and the effect they produce:

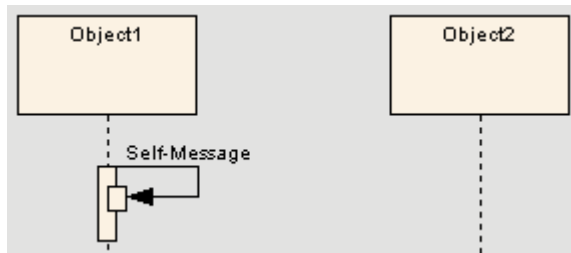
- Vertical Scroll Bar for: list, treelist, report, listview
- Simple rectangle for: textbox, editbox, text, edit, input, date, time
- Shaded rectangle for: form, panel, dialog, dialogue
- Dark rectangle for: button, push button, action
- Combo-box type for: combobox, combo, dropdown, drop list
- Check box for: check, checkbox, check box, tick
- Radio button for: radio, group, option

The screenshot shows a window titled "Payment Screen". It contains the following elements:

- A scrollable area labeled "Order Details".
- A button labeled "OrderTotal".
- Another scrollable area labeled "Ship Details".
- Three text input fields: "CreditCard#", "ExpiryDate", and "Name OnCard".
- Two buttons: "Submit" and "Cancel".
- Two buttons at the bottom: "Back" and "View Cart".

5.3.4.17 Self-Message

[Common Usage](#).. [Further Information](#)



A *self-message* reflects a new process or method invoked within the calling lifeline's operation. It is a specification of a message.

Common Usage

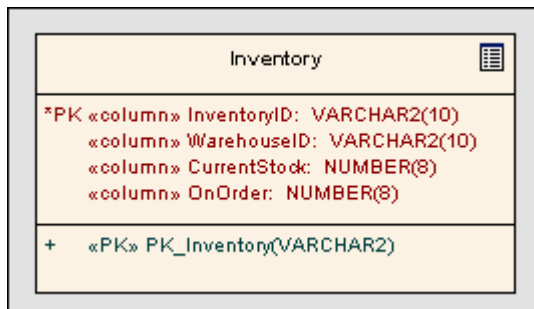
- [Sequence Diagram](#)



Further Information

- [Message](#)

5.3.4.18 Table



A *table* is a stereotyped class. It is drawn with a small table icon in the upper right corner. A table element has a special properties dialog with settings for database type and the ability to set column information and data related operations such as triggers and indexes. When setting up a table, make sure you set the default database type for that table - otherwise you will not have any data types to choose from when creating columns.

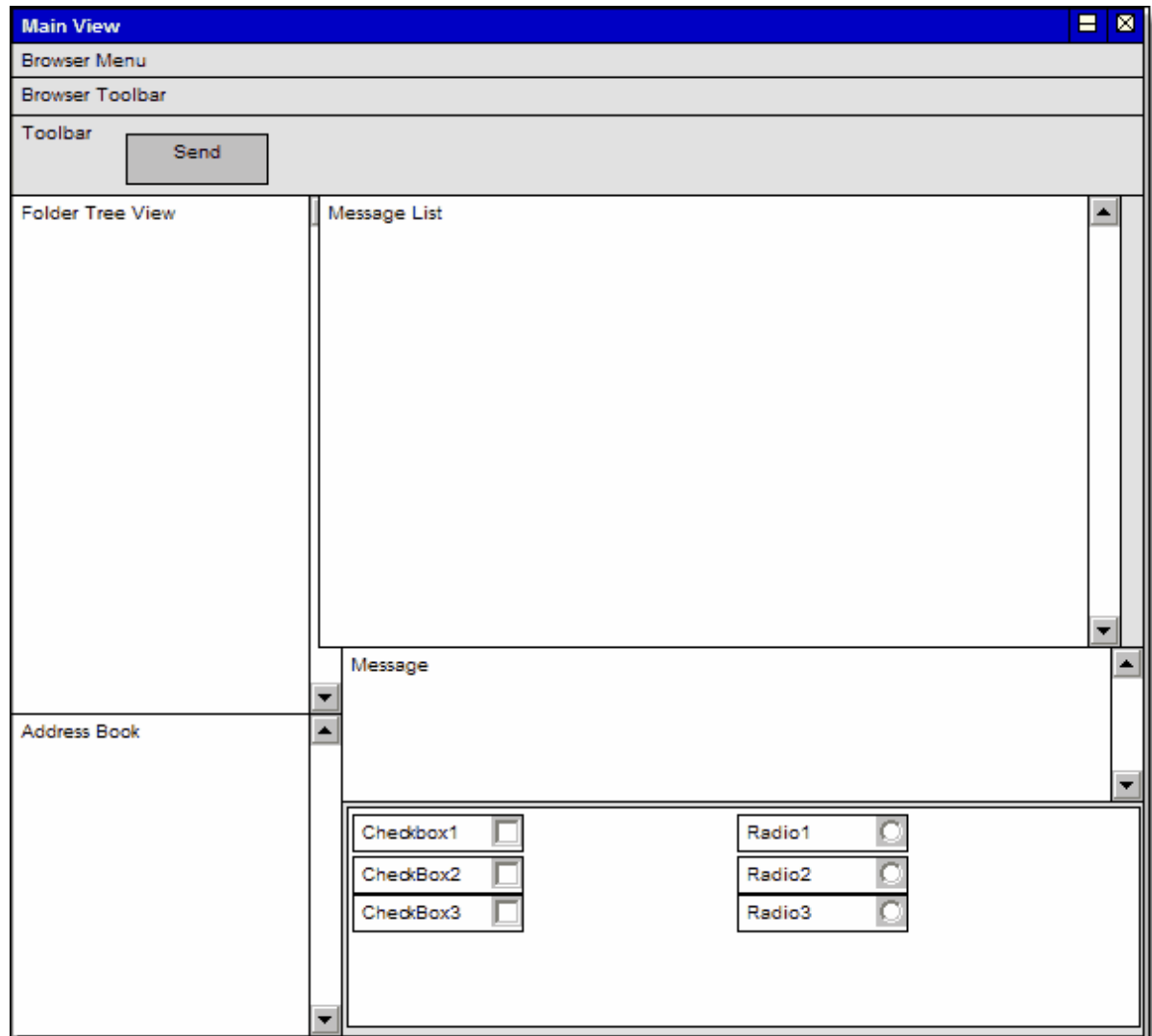
Note: PK and FK tags indicate primary and foreign key respectively.

5.3.4.19 UI Element

A *UI element* is a stereotyped class representing any user interface control element (eg. edit box). It is for capturing screen layouts and requirements. User interface controls may be drawn in a variety of ways depending on the element stereotype:

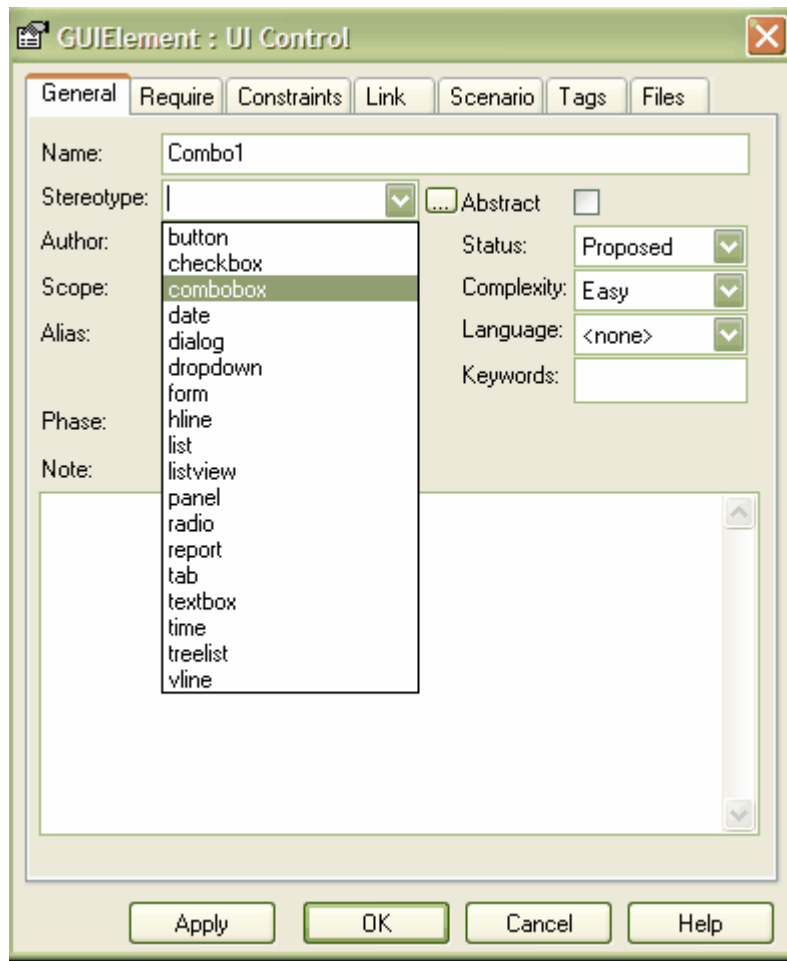
- Vertical Scroll Bar for: list, treelist, report, listview
- Simple rectangle for: textbox, editbox, text, edit, input, date, time

- Shaded rectangle for: form, panel, dialog, dialogue
- Dark rectangle for: button, push button, action
- Combo-box type for: combobox, combo, dropdown, drop list
- Check box for: check, checkbox, check box, tick
- Radio button for: radio, group, option



Set the UI Element stereotype to one of the words in the above list to change its representation use the following instructions:

1. Create a Custom diagram and then from the *UML toolbox* select the *Custom* tools
2. Drag an UI Control from the UML toolbox onto the diagram.
3. Give the UI Control element the appropriate stereotype from the *Stereotype* dropdown list in the GUIElement : UI Control dialog.



4. Give the element an appropriate name and then press the *Apply* button.

5.3.4.20 Web Stereotypes

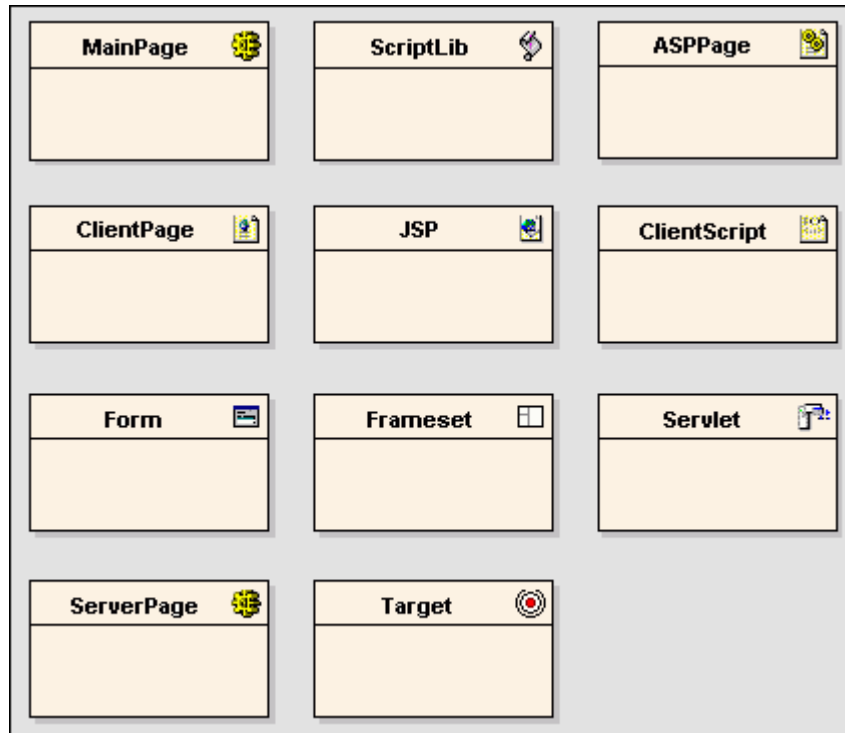
Enterprise Architect supports a number of stereotypes for web page modeling. These supported types will display with a graphical icon instead of the usual <<stereotype>> format. These stereotypes are only supported for elements of type 'Class'. The image below indicates the various graphical icons and their associated stereotypes.

Set a Web Icon

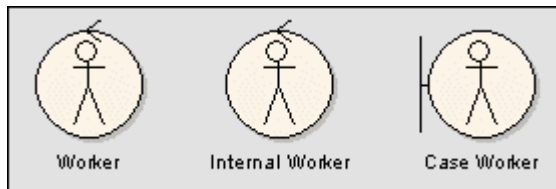
To set a web icon, follow the steps below:

1. Create a new class and place in a diagram.
2. Bring up the *Class properties* dialog.
3. From the drop down *Stereotype* combo, select the required stereotype (will be similar to examples below).
4. Press *OK*.

If you have selected a supported stereotype, the class will display as in one of the examples below:



5.3.4.21 Worker



Some additional stereotyped classes are available for performing business process modeling. These stereotypes are used to model the workers within and external to the system.

The additional stereotypes are

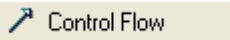
- *Case worker* (class with stereotype 'case worker')
- *Internal worker* (class with stereotype 'internal worker')
- *Worker* (class with stereotype 'worker')

5.4 UML Connections

What is a Connection?

A connection is a *logical or functional relationship between model elements*. There are several different connection types, each having a particular purpose and syntax. Enterprise Architect supports all of the UML connections as well as some custom ones of its own. Together with the [UML Elements](#), these form the basis of UML models. For more insight into using these connectors, consult the appropriate topic by

clicking on the table below.

Behavioral Diagram Connectors**Activity Diagrams**

Control Flow



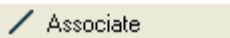
Dependency



Interrupt Flow



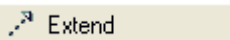
Object Flow

Use Case Diagrams

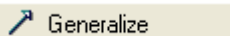
Associate



Dependency



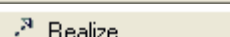
Extend



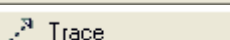
Generalize



Include



Realize



Trace



Use

State Machine Diagrams

Transition



Object Flow

Timing Diagrams

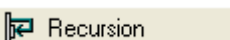
Message

Sequence Diagrams

Message



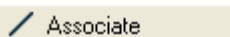
Self-Message



Recursion



Call

Communication Diagrams

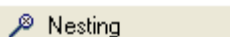
Associate



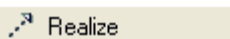
Dependency



Message



Nesting



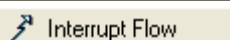
Realize

Interaction Overview Diagrams

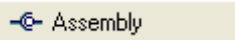
Control Flow



Dependency



Interrupt Flow

Structural Diagram Connectors**Composite Structure Diagrams**

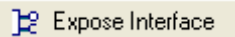
Assembly



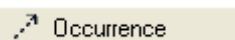
Connector



Dependency



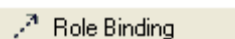
Expose Interface



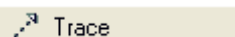
Occurrence



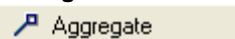
Represents



Role Binding



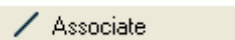
Trace

Package, Class and Object Diagrams

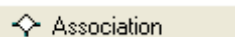
Aggregate



Assembly



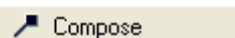
Associate



Association



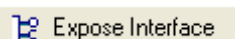
Association Class



Compose



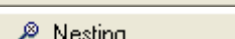
Dependency



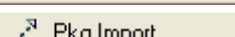
Expose Interface



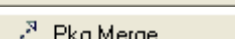
Generalize



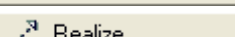
Nesting



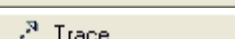
Pkg Import



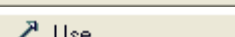
Pkg Merge



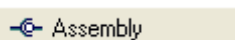
Realize



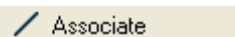
Trace



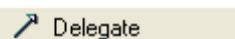
Use

Component Diagrams

Assembly



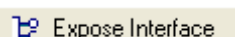
Associate



Delegate



Dependency



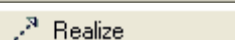
Expose Interface



Generalize



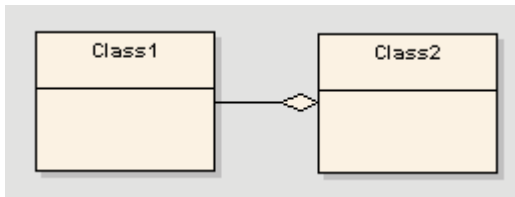
Manifest



Realize

5.4.1 Aggregate

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



An *aggregation* relationship shows that an element *contains* or is *composed of* other elements. Used in class models to show how more complex elements are built from a collection of simpler elements (eg. a car from wheels, tires, motor, etc.). A stronger form of aggregation, known as *composite aggregation*, is used to indicate ownership of the whole over its parts. After drawing an aggregation association, its form can be changed; consult the instructions under *Further Information*.

Common Usage

 Aggregate

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

Further Information

- [Change the Form of an Aggregation Link](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 5) states:

"A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part."

The OMG UML specification (*UML 2.0 Superstructure*, p. 82) states:

"Composite aggregation is a strong form of aggregation that requires a part instance be included in at most one composite at a time. If a composite is deleted, all of its parts are normally deleted with it."

5.4.1.1 Change Aggregation Link Form

[Further Information](#)

In Enterprise Architect, the default aggregation relationship is the weak form of the relationship, represented by a hollow diamond. To change the form of an aggregation link from weak to strong, follow the steps below.

1. Right click on an aggregation link to bring up the context menu.
2. Select **Set Aggregation to Composite**.
3. The diamond will now be shown as filled.

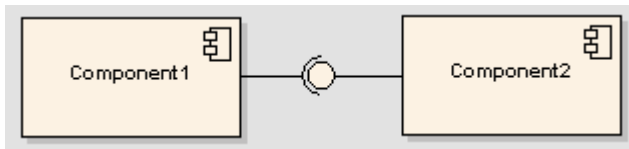
Note: If the link is already a Strong (Composition) link, the context menu option will change to **Set Aggregation to Shared**.

Further Information

- [Aggregate](#)

5.4.2 Assembly

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



As shown above, the *assembly* connector bridges a component's required interface (Component1) with the provided interface of another component (Component2).

Common Usage

- [Component Diagram](#)

 Assembly

Further Information

- [Interface](#)
- [Expose Interface](#)

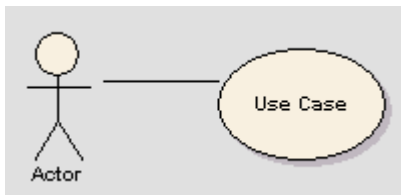
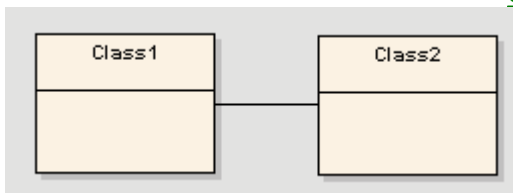
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 143) states:

"An assembly connector is a connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port."

5.4.3 Associate

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



An *association* implies two model elements have a relationship - usually implemented as an instance variable in one class. This connector may include named roles at each end, multiplicity, direction and constraints. Association is the general relationship type between elements. For more than two elements, a [diagonal representation](#) toolbox element can be used as well.

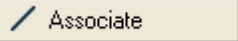
When code is generated for class diagrams, associations become instance variables in the target class.

With deployment diagrams, an association indicates a communication path between nodes, through which

nodes are able to transmit signals and messages.

Common Usage

- [Class Diagram](#)
- [Package Diagram](#)
- [Object Diagram](#)
- [Communication Diagram](#)
- [Deployment Diagram](#)



Further Information

- [Diagonal Representation](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 81) states:

"An association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type. When a property is owned by an association it represents a non-navigable end of the association. In this case the property does not appear in the namespace of any of the associated classifiers. When a property at an end of an association is owned by one of the associated classifiers it represents a navigable end of the association. In this case the property is also an attribute of the associated classifier. Only binary associations may have navigable ends."

5.4.3.1 Diagonal Representation

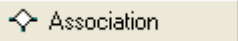
[Common Usage](#) | [Further Information](#)



Diagonal representation can be used to graphically ease ternary or multiply associated elements.

Common Usage

- [Class Diagram](#)

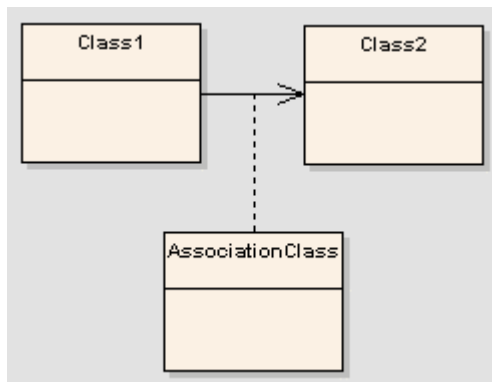


Further Information

- [Associate](#)

5.4.4 Association Class

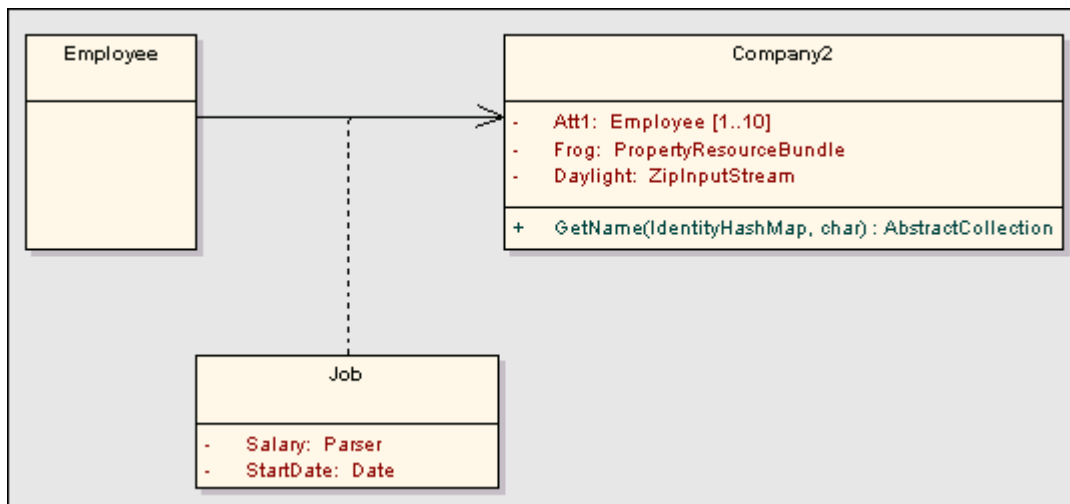
[Common Usage](#) | [OMG UML Specification](#)



An *association class* connection is a UML construct that allows an association connector to have attributes and operations (features). This results in a hybrid relation with the characteristics of a linkage and a class. When an association class link is added, EA creates a class as well. This is automatically linked to the association. When you delete the association, the class is deleted also. If you hide the association, the class is hidden also. An association class is used to model particular types of connections in UML (see the [UML Specification](#) for more details).

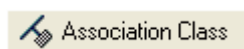
To add an association class - in the UML toolbox select Structure, find the *Association Class* icon and click once. Click on the source object in the diagram and hold the mouse down while you drag the line to the target element - release the mouse button. EA will draw the link and add the class. You will be prompted to add the class name - note that the name of the class and the link are the same.

The following diagram illustrates an association class between model elements. Note the dotted line from the class to the association. This line is not movable or able to be deleted.



Common Usage

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



Further Information

- [Link a New Class to an Existing Association](#)

OMG UML Specification

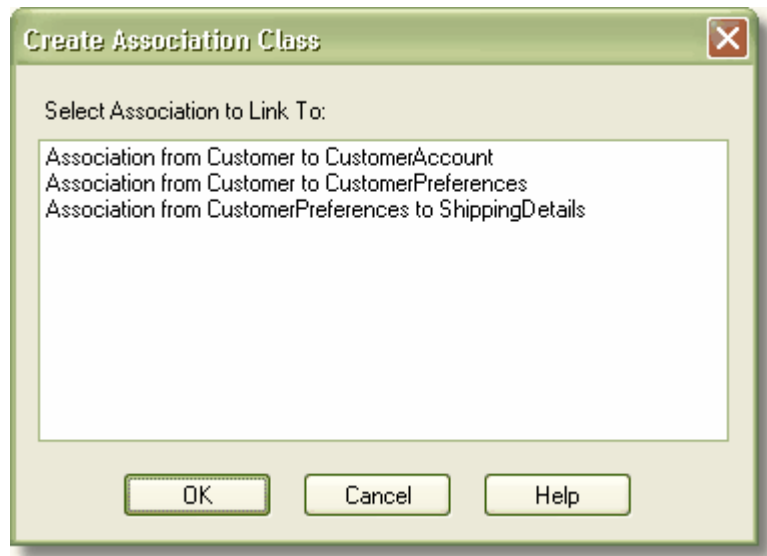
The OMG UML specification (*UML 2.0 Superstructure*, p. 118) states:

"A model element that has both association and class properties. An AssociationClass can be seen as an association that also has class properties, or as a class that also has association properties. It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not to any of the classifiers."

5.4.4.1 Link a New Class to an Existing Association[Further Information](#)

To link a new class to an existing association, follow the steps below:

1. Create a class in the diagram workspace containing the association to link.
2. Right click on the new class for the context menu.
3. Select *Link Class to Association*.



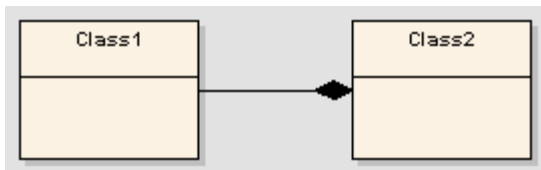
3. In the *Link class to association* dialog, check the link to connect to.
4. Press *OK*.

Further Information

- [Association Class](#)

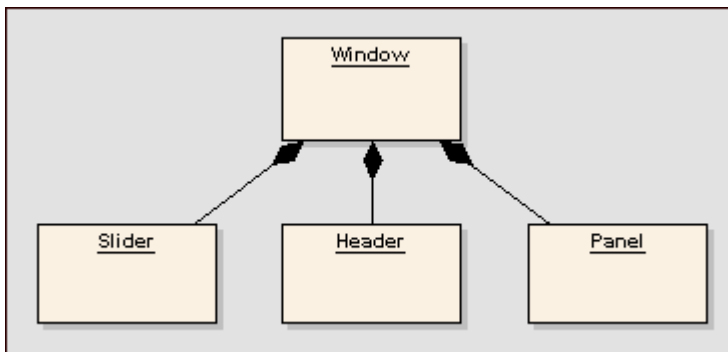
5.4.5 Compose

[Common Usage](#) | [OMG UML Specification](#)



A *composite aggregation* is used to depict an element which is made up of smaller components. A component - or part instance - can be included in a maximum of one composition at a time. If a composition is deleted, usually all of its parts are deleted with it; however a part can be individually removed from a composition without having to delete the entire composition. Compositions are transitive, asymmetric relationships and can be recursive.

See example below.



Common Usage

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

 Compose

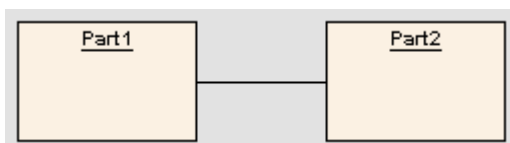
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 7) states:

"A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive."

5.4.6 Connector

[Common Usage](#) | [OMG UML Specification](#)

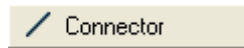


Connectors illustrate communication links between parts to fulfill the structure's purpose. Each connector end

is distinct, controlling the communication pertaining to its connecting element. These elements can define constraints specifying this behavior. Connectors can have multiplicity.

Common Usage

- [Composite Structure Diagram](#)



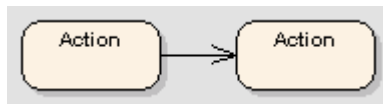
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 163) states:

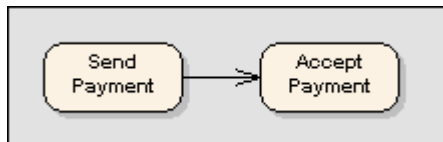
"Specifies a link that enables communication between two or more instances. This link may be an instance of an association, or it may represent the possibility of the instances being able to communicate because their identities are known by virtue of being passed in as parameters, held in variables, created during the execution of a behavior, or because the communicating instances are the same instance. The link may be realized by something as simple as a pointer or by something as complex as a network connection. In contrast to associations, which specify links between any instance of the associated classifiers, connectors specify links between instances playing the connected parts only."

5.4.7 Control Flow

[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



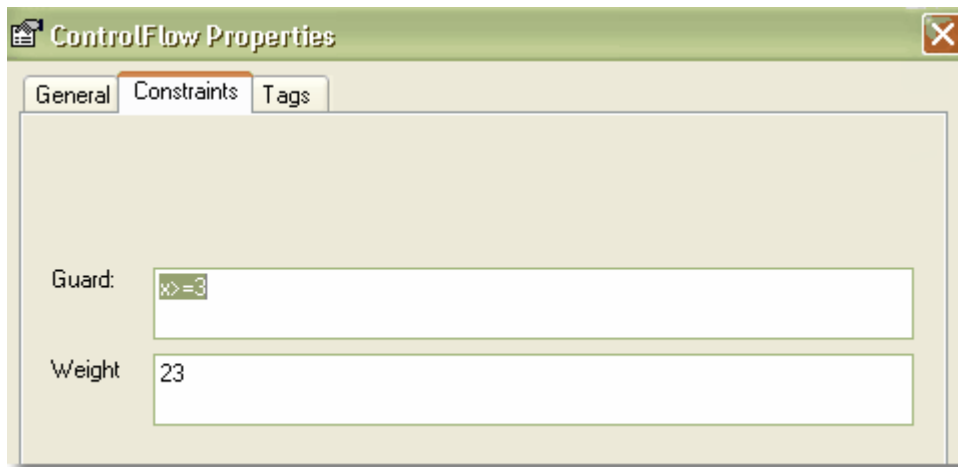
The *control flow* is a connector linking two nodes in an Activity diagram. Control flow connectors bridge the flow between activity nodes, by directing the flow to the target node once the source node's activity is completed.



Control Flows and Object Flows may define a Guard and a Weight condition.

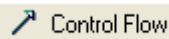
A **Guard** defines a condition that must be true before control will pass along that activity edge. A practical example of this is where two or more activity edges (control flows) exit from a Decision element. Each flow should have a Guard condition which is exclusive of each other and defines which edge will be taken under what conditions. The ControlFlow properties dialog allows you to set-up Guard conditions on Control Flows and on Object Flows.

A **Weight** defines the number of tokens that may flow along a Control or Object Flow connection when that edge is traversed. Weight may also be defined on the ControlFlow and ObjectFlow properties dialog.



Common Usage

- [Activity Diagrams](#)



Further Information

- [Actions](#)
- [Activity Diagrams](#)

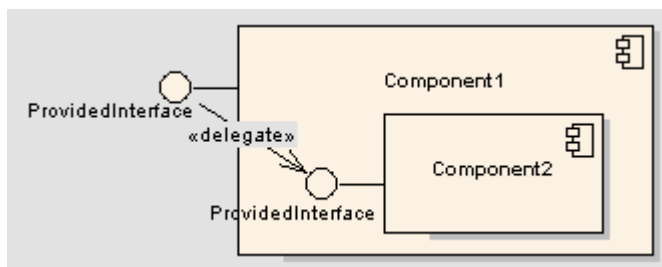
The OMG UML specification states:

The OMG UML specification (*UML 2.0 Superstructure*, p. 315) states:

"A control flow is an edge starts an activity node after the previous one is finished."

5.4.8 Delegate

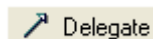
[Common Usage](#) | [OMG UML Specification](#)



A *delegate* connector defines the internal assembly of a component's external ports and interfaces. Using a delegate connector wires the internal workings of the system to the outside world, by a delegation of the external interfaces' connections.

Common Usage

- [Component Diagram](#)



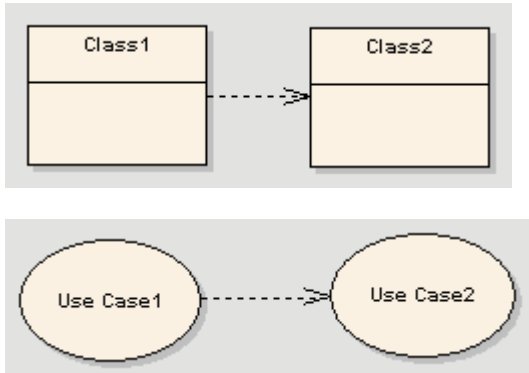
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 143) states:

"A *delegation connector* is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component's parts. It represents the forwarding of signals (operation requests and events) : a signal that arrives at a port that has a delegation connector to a part or to another port will be passed on to that target for handling."

5.4.9 Dependency

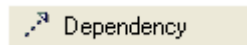
[Common Usage](#)...[OMG UML Specification](#)



Dependency relationships are used to model a wide range of dependent relationships between model elements - and even between models themselves. The Dependencies package as defined in UML 2.0 has many derivatives, such as Realization, Instantiation and Usage. Once you create a dependency you can further refine its meaning by applying a specialized stereotype.

Common Usage

- [Use Case Diagram](#)
- [Structural Diagrams](#)
- [Activity Diagram](#)



Further Information

- [Applying a Stereotype](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 108) states:

"A *dependency* is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s)."

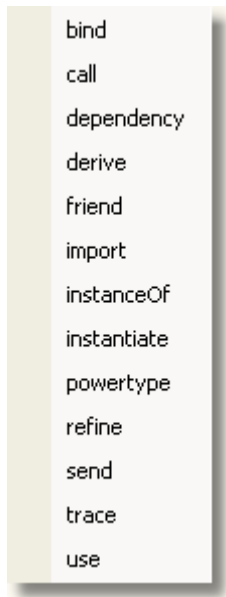
5.4.9.1 Applying a Stereotype

[Further Information](#)

To apply a stereotype to a dependency relationship, follow the steps below:

1. Select the dependency relationship to change.

2. Right click on the link, and from the context menu, select *Dependency Stereotypes*.
3. From the sub-menu select the required Stereotype:

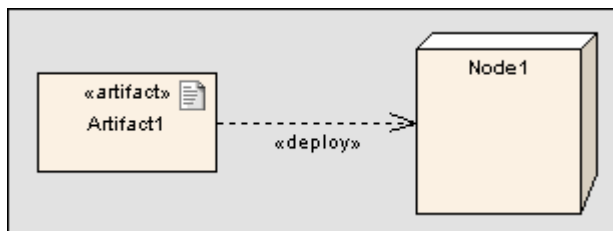


Further Information

- [Dependency](#)

5.4.10 Deployment

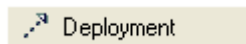
[Common Usage](#) | [OMG UML Specification](#)



A *deployment* is a type of dependency relationship that indicates the deployment of an artifact onto a node or executable target. A deployment can be made at type and instance levels. At the type level, a deployment would be made for every instance of the node. Deployment can also be specified for an instance of a node, so that a node's instances can have varied deployed artifacts. With composite structures modeled with nodes defined as parts, parts can also serve as targets of a deployment relationship.

Common Usage

- [Deployment Diagram](#)



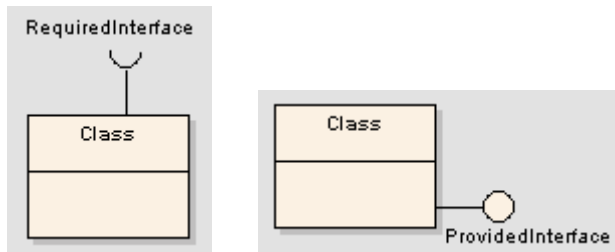
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 187) states:

"A *deployment* is the allocation of an artifact or artifact instance to a deployment target."

5.4.11 Expose Interface

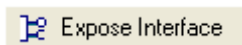
[Common Usage](#)..|..[Further Information](#).



The *expose interface* toolbox element is a graphical way to depict the required and supplied interfaces of a component, class, or part.

Common Usage

- [Component Diagram](#)
- [Class Diagram](#)
- [Composite Structure Diagram](#)

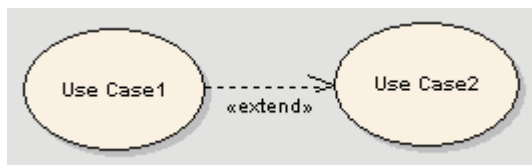


Further Information

- [Interface](#)
- [Assembly](#)

5.4.12 Extend

[Common Usage](#)..|..[OMG UML Specification](#)



An *extend* connection is used to indicate an element extends the behavior of another. Extensions are used in use case models to indicate one use case (optionally) extends the behavior of another. An extending use case often expresses alternate flows.

Common Usage

- [Use Case Diagram](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 515) states:

"This relationship specifies that the behavior of a use case may be extended by the behavior of another (usually supplementary) use case. The extension takes place at one or more specific extension points defined in the extended use case. Note, however, that the extended use case is defined independently of the extending use case and is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending use case defines a set of modular behavior increments that augment an execution"

of the extended use case under specific conditions. Note that the same extending use case can extend more than one use case. Furthermore, an extending use case may itself be extended."

5.4.13 Generalize

[Common Usage](#) | [OMG UML Specification](#)



A *generalization* is used to indicate inheritance. Drawn from the specific classifier to a general classifier, the generalize implication is that the source inherits the target's characteristics.

Common Usage

 Generalize

- [Class Diagram](#)
- [Component Diagram](#)
- [Object Diagram](#)
- [Package Diagram](#)
- [Use Case Diagram](#)

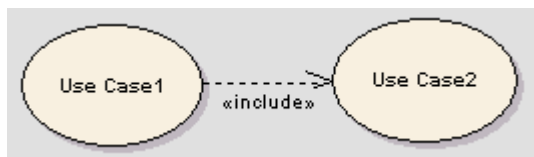
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 66) states:

"A *generalization* is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier."

5.4.14 Include

[Common Usage](#) | [OMG UML Specification](#)



An *include* connection indicates that the source element includes the functionality of the target element.

Include connections are used in use case models to reflect that one use case includes the behavior of another. Use an include relationship to avoid having the same subset of behavior in many use cases; this is similar to [delegation](#) used in class models.

Common Usage

- [Use Case Diagram](#)



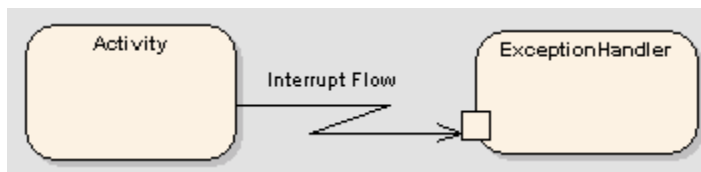
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 518) states:

"Include is a directed relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The including use case may only depend on the result (value) of the included use case. This value is obtained as a result of the execution of the included use case."

5.4.15 Interrupt Flow

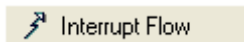
[Common Usage](#)...[OMG UML Specification](#)



The *interrupt flow* is a toolbox element used to define the two UML concepts of connectors for [Exception Handler](#) and [Interruptible Activity Region](#). An interrupt flow is also known as an activity edge.

Common Usage

- [Activity Diagram](#)



Further Information

- [Activity Diagram](#)
- [Exception handler](#)
- [Interruptible Activity Region](#)

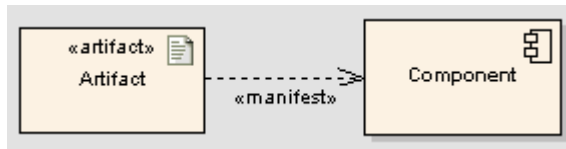
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 293) states:

"An activity edge is an abstract class for directed connections between two activity nodes."

5.4.16 Manifest

[Common Usage](#)..|..[OMG UML Specification](#)



A *manifest* relationship indicates that the artifact source embodies the target model element. [Stereotypes](#) can be added to Enterprise Architect to classify the type of manifestation of the model element.

Common Usage

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



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 195) states:

"An artifact embodies or manifests a number of model elements. The artifact owns the manifestations, each representing the utilization of a packageable element. Specific profiles are expected to stereotype the manifestation relationship to indicate particular forms of manifestation, e.g. <<tool generated>> and <<custom code>> might be two manifestations for different classes embodied in an artifact."

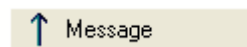
5.4.17 Message

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)

Messages indicate a flow of information or transition of control between elements. Messages can be used by all interaction diagrams except the Interaction Overview diagram, to reflect system behavior. If between classes or classifier instances, the associated list of operations will be available to specify the event.

Common Usage

- [Sequence Diagram](#)
- [Communication Diagram](#)
- [Timing Diagram](#)



Further Information

- [Messages for Timing Diagrams](#)
- [Messages for Sequence Diagrams](#)
- [Messages for Communication Diagrams](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 428) states:

"A Message defines a particular communication between Lifelines of an Interaction."

"A Message is a NamedElement that defines one specific kind of communication in an Interaction. A communication can be e.g. raising a signal, invoking an Operation, creating or destroying an Instance. The Message specifies not only the kind of communication given by the dispatching ExecutionOccurrence, but also the sender and the receiver."

"A Message associates normally two EventOccurrences - one sending EventOccurrence and one receiving EventOccurrence."

Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.4.17.1 Message (Communication Diagram)

[Common Usage](#) | [Further Information](#)

A message in a Communication diagram is equivalent in meaning to a message in a Sequence diagram. It implies that one object uses the services of another object, or sends a message to that object. Communication messages in Enterprise Architect are always associated with an association link between object instances. Always create the association link first - then add messages to the link.

Messages may be dragged into a suitable position by clicking and dragging on the message text.

Communication messages should be ordered to reflect the sequencing of the diagram. The numbering scheme should reflect the nesting of each event. A sample sequencing scheme could be 1, 2, 2.1, 2.2, 3. This would indicate events 2.1 and 2.2 occur within an operation initiated by event 2.

If the target object is a class or has its instance classifier set, the drop-down list of possible message names will include the exposed operations for the base type.

Common Usage

- [Communication Diagrams](#)

Further Information

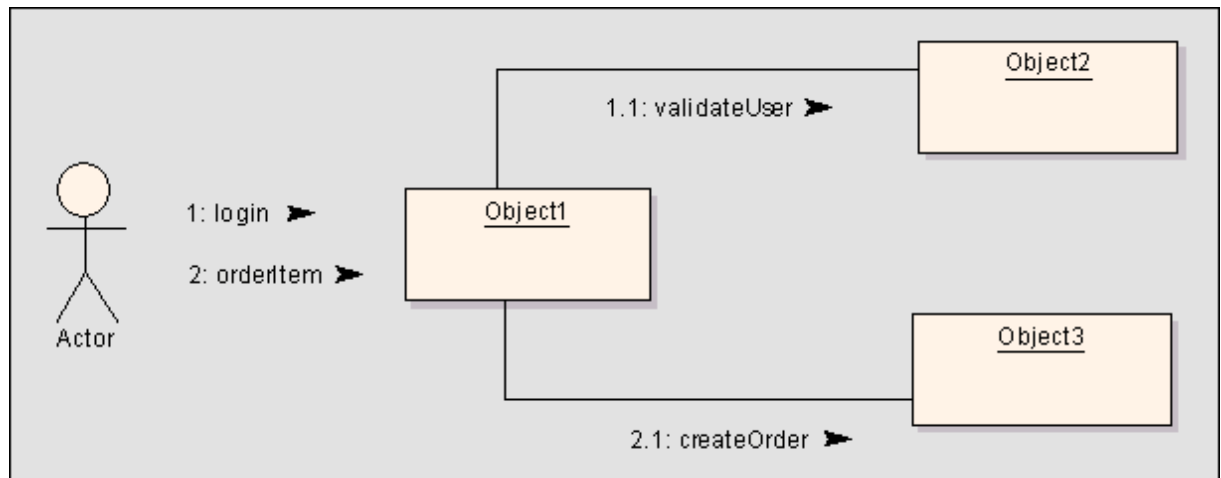
- [Create a Communication Message](#)
- [Ordering Communication Messages](#)

5.4.17.1.1 Create a Communication Message

[Further Information](#)

To create a communication message, follow the steps below:

1. Open a diagram (any type - except Sequence).
2. Add the required objects.
3. Add an association relationship between all objects that will communicate.
4. Right click on an association to view the context menu.
5. Select the option to add a message from one object to another.
6. When the *Message* dialog appears, fill in a *Name* and any other required details.
7. Press *OK*. The message is added - linked to the association and object instances.
8. Move to the required position.



Further Information

- [Message \(Communication\)](#)

5.4.17.1.2 Re-Order Messages

Further Information

When constructing your Communication diagram, it is frequently necessary to re-order the sequence of messages and to create or delete message 'groups'. There are two dialogs that help you perform these tasks - the *Message Properties* dialog and the *Sequence Communications* dialog.

Organizing Message Groups

If you have several messages that are in the form 1.2, 1.2, 1.3, 1.4 etc., but would like to start a new numbering group on the third message eg. 1.1, 1.2, 2.1, 2.2, 2.3 etc., indicate message 3 is a 'Start Group' message by using the following instructions.

1. View the communication message context menu by right clicking on a message.
2. Select *Communication Properties...* to open the *Message properties* dialog.
3. To make the selected message the start of a new group, check the box called *Start New Group*.
4. Press *OK* to save changes.

Message Properties

Signature

Message:

Show Inherited Methods

Parameters:

Return: Is Return

Sequence Expression

Condition:

Is Iteration Start New Group

Constraint:

Control Flow Type:

Synch: Lifecycle:

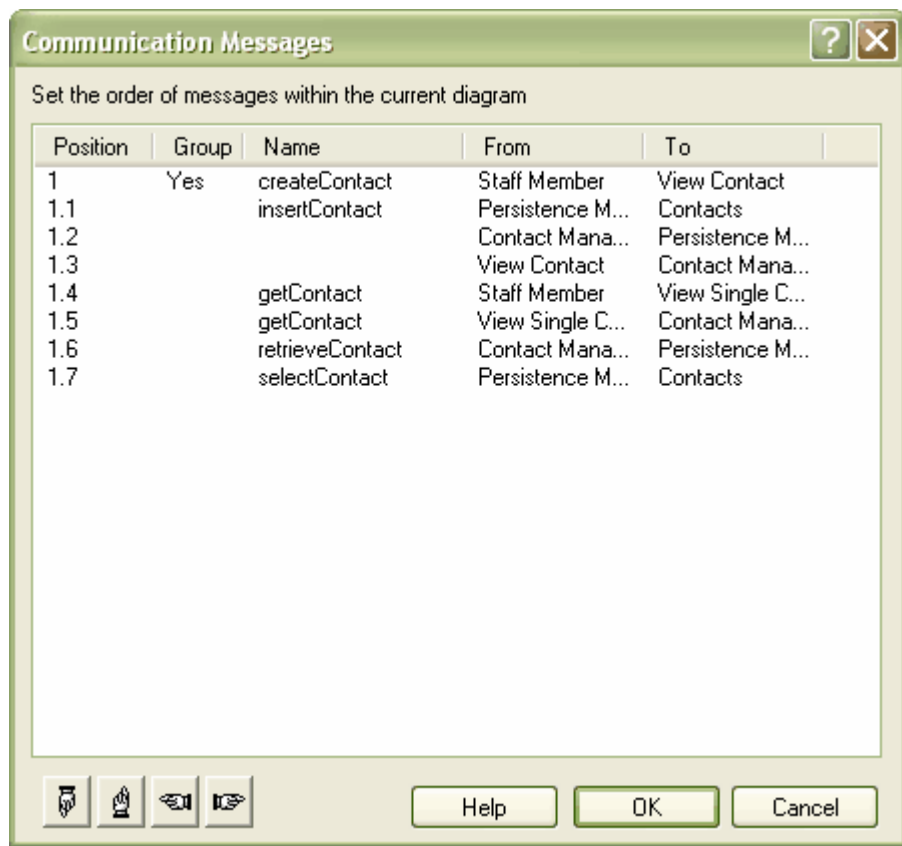
Frequency:

Notes:

Sequencing Messages

To re-order messages, follow the steps below:

1. Select *Diagram | Sequence Communications* from the main menu to open the *Communication Messages* sequencing dialog.
2. Select the message you want to move, and move up or down the sequence using the hand icons. Repeat until the sequence matches your requirements.
3. Press **OK** to apply changes.



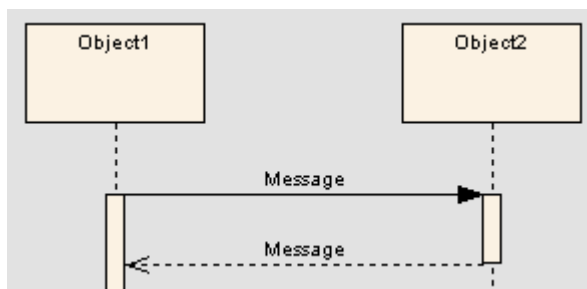
Further Information

- [Message \(Communication\)](#)

Note: Communication diagrams were known as Collaboration diagrams in UML 1.4.

5.4.17.2 Message (Sequence Diagram)

[Common Usage](#)...[Further Information](#)



Sequence diagrams depict work flow or activity over time using messages passed from element to element. These messages correspond in the software model to class operations and behavior. They are semantically similar to the messages passed between elements in a Communication diagram.

Double click on a sequence message to view the *Message properties* dialog. If the destination class has defined operations, these will appear in the message name drop down list to be selected.

You can also set:

- Conditions: indicates what must be true for message to be sent
- Return value
- Synchronization
- Frequency
- Creation (Lifecycle) - set to New for element creation, Delete for element destruction

Common Usage

- [Sequence Diagram](#)

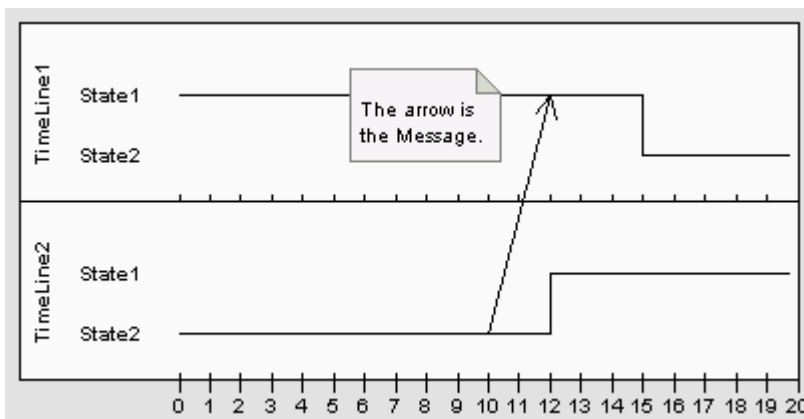
→ Message

Further Information

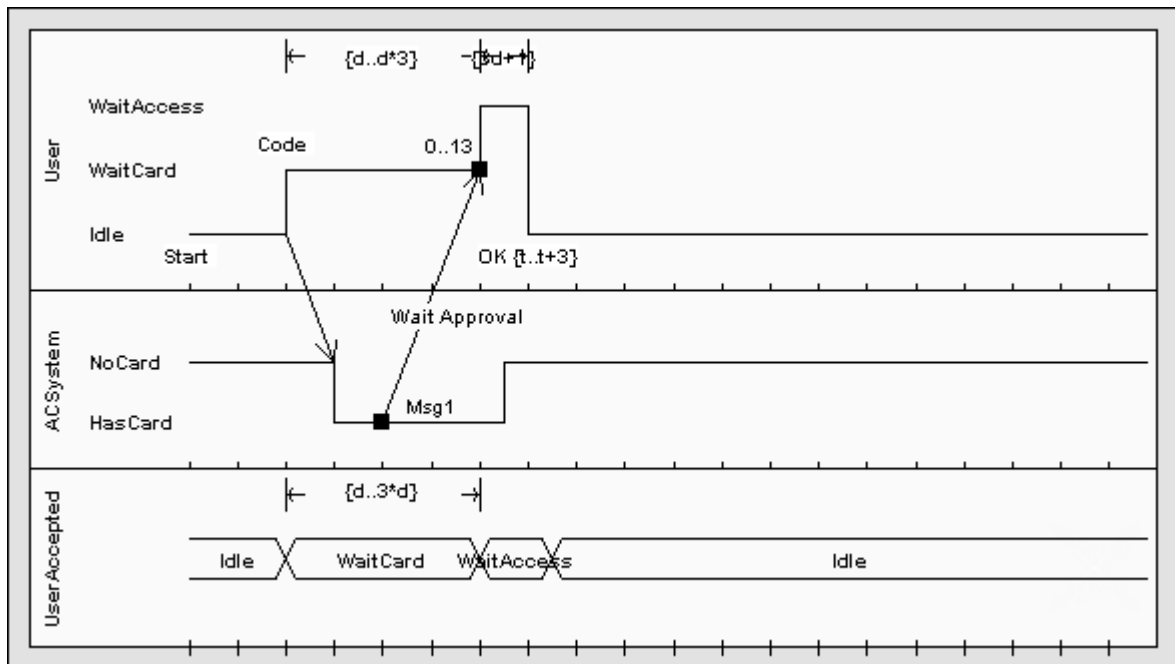
- [Self-Message](#)
- [Call](#)

5.4.17.3 Message (Timing Diagram)

[Common Usage](#)..|.. [Further Information](#)



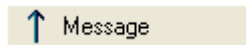
Messages are the communication links between lifelines. In the case of a timeline, a message is a connection between two timeline objects.



Refer to figure 352 (UML 2.0 Superstructure, p. 454) and figure 351 (UML 2.0 Superstructure, p. 453).

Common Usage

- [Timing Diagram](#)



Further Information

- [Timing Diagram](#)

5.4.17.3.1 Create a Timing Message

Further Information

To create a message in a Timing diagram, at least two lifeline objects (state or value), each with existing transition points, must first be created. To create a message between lifelines, follow the steps below:

1. Select a message from the *Timing* section of the UML toolbox.
2. Click on one of the lifelines in the Timeline diagram.
3. Drag the mouse onto the lifeline where the Message is to be connected.

The following dialog box will become visible:

Timing Message

Scope

Start: ACSystem End: User

Static

Start Time: 40

End Time: 60

Transition To: WaitAccess

Name: Msg1

Event: Wait Approval

Time Constraint: 0..13

Duration Constraint: 3d+1

Time Observation:

Duration Observation:

OK Cancel

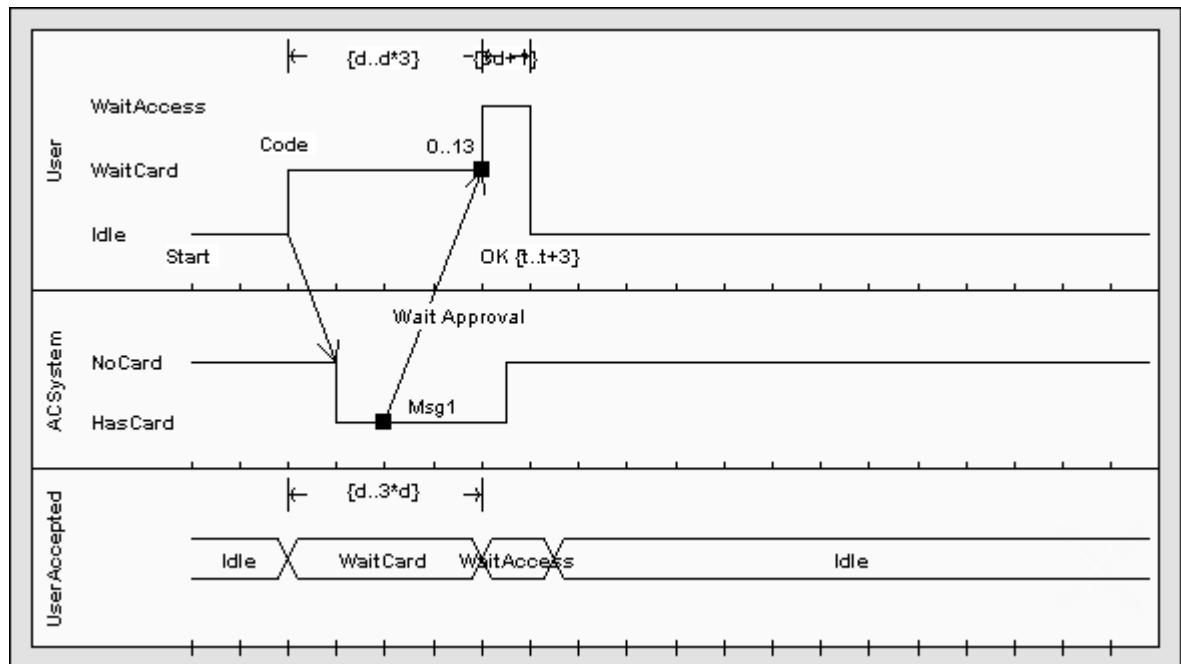
The dialog will consist of a set of transition points. Each transition point can be defined with the following properties:

Start	Defines the lifeline where the message originates.
End	Defines the lifeline where the message terminates.

These are set by default when a message is created by dragging the cursor between two lifelines.

Start Time	Specifies the start time for a message.
End Time	Specifies the end time for a message.
Event	Describes the occurring event.
Time Constraint	Refers to the time taken to transmit a message.
Time Observation	Provides information on the time of a sent message.
Duration Constraint	Pertains to a lifeline's period at a particular state. The constraint could be instigated by that lifeline's receipt of a message.
Duration Observation	Indicates the interval of a lifeline at a particular state, begun from a message receipt.

The following diagram shows the message configured by the above dialog snapshot.



Refer to figure 352 (UML 2.0 Superstructure, p. 454) and figure 351 (UML 2.0 Superstructure, p. 453).

Further Information

- [Message \(Timing Diagram\)](#)

5.4.17.4 Message Endpoint

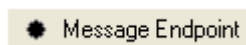
[Common Usage](#) | [Further Information](#) | [OMG UML Specification](#)



A *message endpoint* element defines the termination of a state or value lifeline in a Timing diagram.

Common Usage

- [Timing Diagram](#)



Further Information

- [Timing Diagram](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 434) states:

"A *Stop* is an *EventOccurrence* that defines the termination of the instance specified by the *Lifeline* on which the *Stop* occurs."

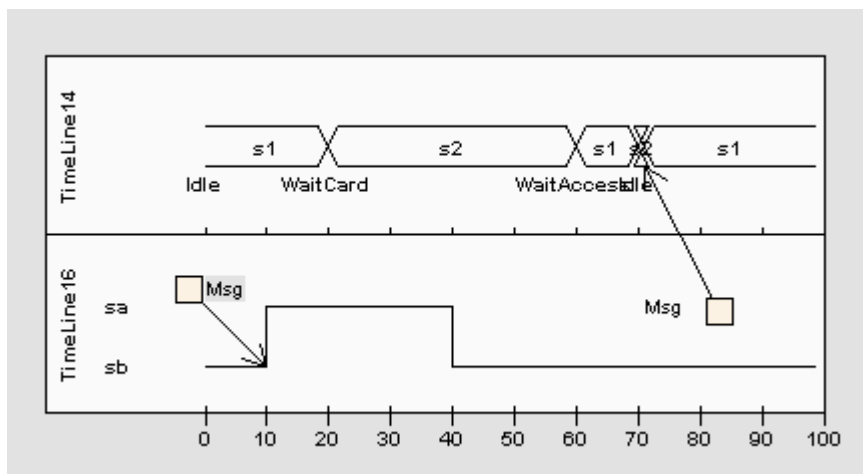
5.4.17.5 Message Label

[Common Usage](#)..|..[Further Information](#)..|..[OMG UML Specification](#)



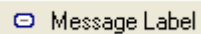
A *message label* is an alternate way to denote messages between lifelines, which is useful for 'uncluttering' timing diagrams strewn with messages. To indicate a message between lifelines, draw a connector from the source lifeline into a message label. Next, draw a connector from another message label to the target lifeline. Note that the label names must match to reflect that the message occurs between the two message labels.

The below diagram illustrates how message labels are used to construct a message between lifelines.



Common Usage

- [Timing Diagram](#)

 Message Label

Further Information

- [Timing Diagram](#)

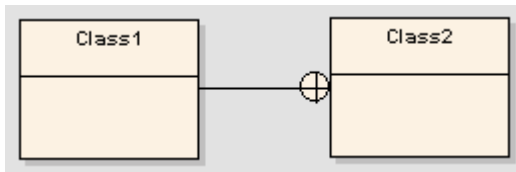
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 451) states:

"Labels are only notational shorthands used to prevent cluttering of the diagrams with a number of messages crisscrossing the diagram between Lifelines that are far apart. The labels denote that a Message may be disrupted by introducing labels with the same name."

5.4.18 Nesting

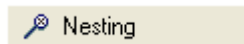
[Common Usage](#)



The *nesting* connector is an alternative graphical notation for expressing containment or nesting of elements within other elements. It is most appropriately used for displaying package nesting.

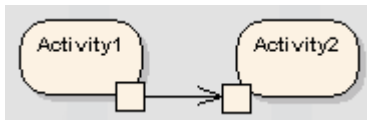
Common Usage

- [Package Diagram](#)



5.4.19 Object Flow

[Common Usage](#)...[Further Information](#)...[OMG UML Specification](#)



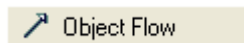
Object flows are used in Activity diagrams and State Machines. When used in an Activity diagram, an object flow connects two elements, with specific data passing through it. Please refer to the [Further Information](#) section below to view [sample Activity diagrams](#) using object flows. With respect to State Machines, an object flow is a specification of a state flow or transition. It implies the passing of an object instance between elements at run-time.

You can insert an object flow from the State or Activity sections of the toolbox, or from the drop-down list of all relationships, located in the header toolbar. You can also modify a transition connection to an ObjectFlow by checking the ObjectFlow check box on the connection properties dialog.

See [ControlFlow](#) for information on setting up Guards and Weights on ObjectFlows.

Common Usage

- [Activity Diagram](#)
- [Interaction Overview Diagram](#)



Further Information

- [Using Object Flows in Activity Diagrams](#)
- [Action Pin](#)
- [Object](#)

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 12) states:

"A state in an activity diagram that represents the passing of an object from the output of actions in one state to the input of actions in another state."

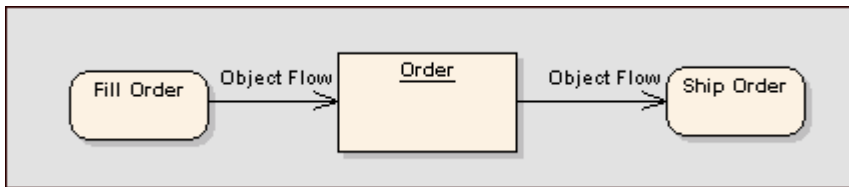
The OMG UML specification (*UML 2.0 Superstructure*, p. 344) states:
 "An object flow is an activity edge that can have objects or data passing along it."

5.4.19.1 Using Object Flows in Activity Diagrams

Further Information

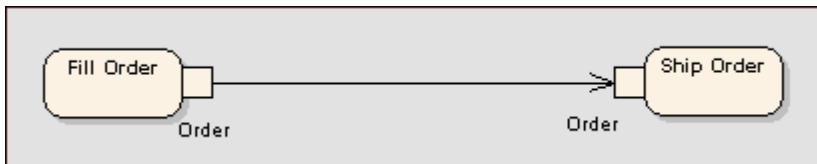
In Activity diagrams, there are several ways to define the flow of data between objects.

The below diagram depicts a simple object flow between two actions, Fill Order and Ship Order, both accessing order information.



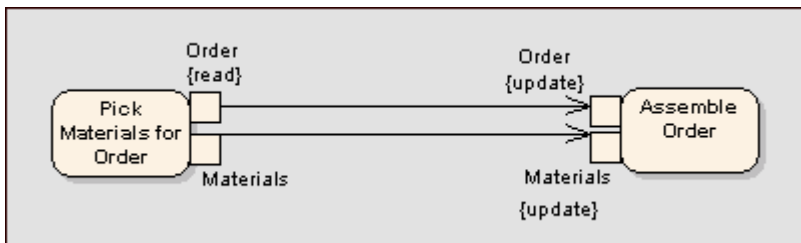
Refer to figure 271 (UML 2.0 Superstructure, p. 347).

This explicit portrayal of the data object, Order, connected to the activities by two object flows, can be refined by using the following format. Here, action pins are utilized to reflect the order.



Refer to figure 271 (UML 2.0 Superstructure, p. 347).

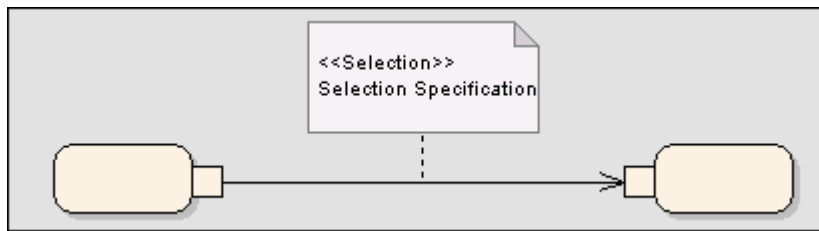
Below is an example of multiple object flows exchanging data between two actions.



Refer to figure 272 (UML 2.0 Superstructure, p. 347).

Selection and transformation behavior, together composing a sort of query, can specify the nature of the object flow's data access. Selection behavior determines which objects are affected by the connection. Transformation behavior might then further specify the value of an attribute pertaining to a selected object.

Selection and transformation behaviors can be defined by attaching a note to the object flow. To do this, right-click on the object-flow and select *Attach Note or Constraint*. A dialog will indicate other flows in the diagram, to which you can select to attach the note, if the behavior applies to multiple flows. To comply with UML 2, preface behavior with the notation <<selection>> or <<transformation>>.



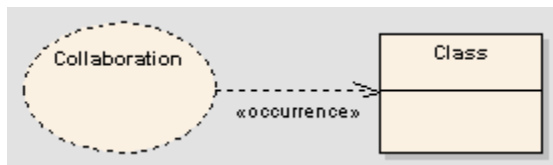
Refer to figure 268 (UML 2.0 Superstructure, p. 346).

Further Information

- [Object Flow](#)

5.4.20 Occurrence

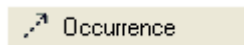
[Common Usage](#) | [OMG UML Specification](#)



An *occurrence* relationship indicates that a collaboration represents a classifier. An occurrence connector is drawn from the collaboration to the classifier.

Common Usage

- [Composite Structure Diagram](#)



Further Information

- [Collaboration](#)

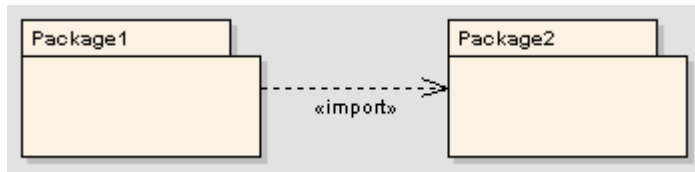
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 161) states:

"A dashed arrow with a stick arrowhead may also be used to show that a collaboration represents a classifier, optionally labelled with the keyword «occurrence»."

5.4.21 Package Import

[Common Usage](#)..|..[OMG UML Specification](#)



A *package import* relationship is drawn from a source package to a package whose contents will be imported. Private members of a target package cannot be imported.

Common Usage

- [Package Diagram](#)



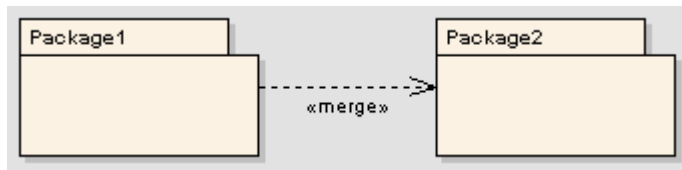
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 38) states:

"A *package import* is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace. Conceptually, a *package import* is equivalent to having an *element import* to each individual member of the imported namespace, unless there is already a separately-defined *element import*."

5.4.22 Package Merge

[Common Usage](#)..|..[OMG UML Specification](#)



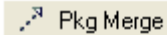
A *package merge* indicates a relationship between two packages whereby the contents of the target package are merged with those of the source package. Private contents of a target package are not merged. The applicability of a package merge addresses any situation of multiple packages containing identically-named elements, representing the same thing. A merging package will merge all matching elements across its merged packages, along with their relationships and behaviors. Note that a package merge essentially performs generalizations and redefinitions of all matching elements, but the merged packages and their independent element representations still exist and are not affected.

The package merge serves a graphical purpose in Enterprise Architect, but creates an ordered "Package" relationship applied to related packages (which can be seen under the *Link tab* in the package's *Properties* dialog). Such relationships can be reflected in XMI exports or EA Automation Interface scripts for code generation or other MDA (Model Driven Architecture) interests.

Package merge relationships are useful to reflect situations where existing architectures contain functionalities involving like elements, which will be merged in a developing architecture. Merging doesn't affect the merged objects, and supports the common situation of product progression.

Common Usage

- [Package Diagram](#)

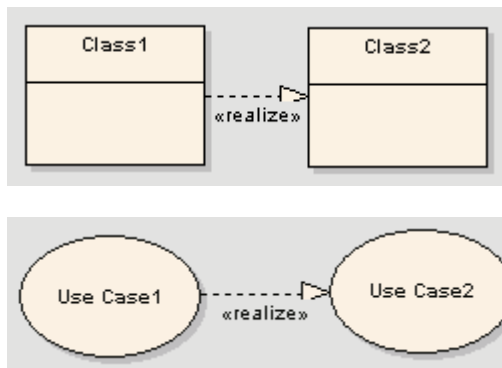

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 101) states:

"A package merge is a relationship between two packages, where the contents of the target package (the one pointed at) is merged with the contents of the source package through specialization and redefinition, where applicable. This is a mechanism that should be used when elements of the same name are intended to represent the same concept, regardless of the package in which they are defined. A merging package will take elements of the same kind with the same name from one or more packages and merge them together into a single element using generalization and redefinitions. It should be noted that a package merge can be viewed as a short-hand way of explicitly defining those generalizations and redefinitions. The merged packages are still available, and the elements in those packages can be separately qualified. From an XML point of view, it is either possible to exchange a model with all PackageMerges retained or a model where all PackageMerges have been transformed away (in which case package imports, generalizations, and redefinitions are used instead)."

5.4.23 Realize

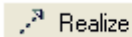
[Common Usage](#) .. [OMG UML Specification](#)



The source object *implements* or *realizes* the destination. *Realize* is used to express traceability and completeness in the model - a business process or requirement is realized by one or more use cases which are in turn realized by some classes, which in turn are realized by a component, etc. Mapping requirements, classes, etc. across the design of your system, up through the levels of modeling abstraction, ensures the big picture of your system remembers and reflects all the little pictures and details that constrain and define it.

Common Usage

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

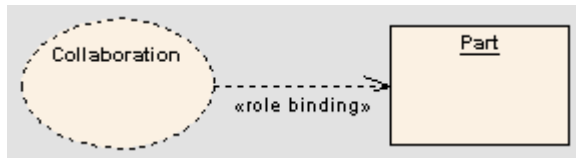

OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 110) states:

"A Realization signifies that the client set of elements are an implementation of the supplier set, which serves as the specification. The meaning of 'implementation' is not strictly defined, but rather implies a more refined or elaborate form in respect to a certain modeling context. It is possible to specify a mapping between the specification and implementation elements, although it is not necessarily computable."

5.4.24 Role Binding

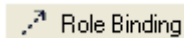
[Common Usage..](#) | [OMG UML Specification](#)



Role binding is the mapping between a collaboration occurrence's internal roles and the respective parts needed to implement a specific situation. The associated parts can have properties defined to enable the binding to occur, and the collaboration to take place.

A role binding connector is drawn between a collaboration and the classifier's fulfilling roles, with the collaboration's internal binding roles labeled on the classifier end of the connector.

Common Usage



- [Composite Structure Diagram](#)

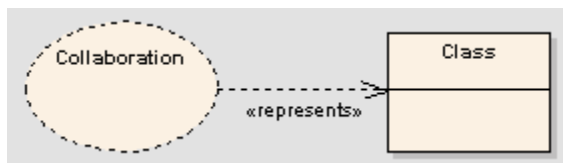
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 160) states:

"A mapping between features of the collaboration type and features of the classifier or operation. This mapping indicates which connectable element of the classifier or operation plays which role(s) in the collaboration. A connectable element may be bound to multiple roles in the same collaboration occurrence (that is, it may play multiple roles)."

5.4.25 Represents

[Common Usage..](#) | [Further Information..](#) | [OMG UML Specification](#)



The *represents* connector indicates a collaboration is used in a classifier. The connector is drawn from the collaboration to its owning classifier.

Common Usage



- [Composite Structure Diagrams](#)

Further Information

- [Collaboration](#)

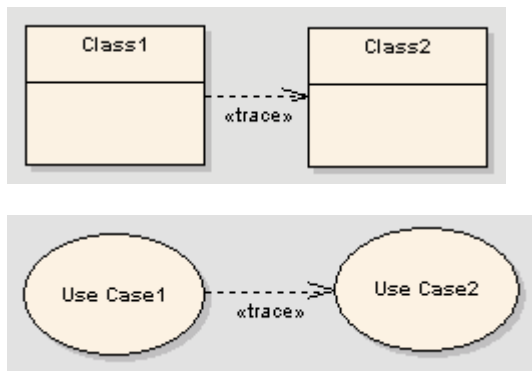
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 161) states:

"A dashed arrow with a stick arrowhead may be used to show that a collaboration is used in a classifier, optionally labelled with the keyword «represents»."

5.4.26 Trace

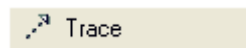
[Common Usage](#)..|.. [Further Information](#)..|.. [OMG UML Specification](#)



The *trace* relationship is a specialization of a dependency, linking model elements or sets of elements that represent the same idea across models. Traces are often used to track requirements and model changes. As changes can occur in both directions, the order of this dependency is usually ignored. The relationship's properties can specify the trace mapping, but the trace is usually bi-directional, informal and rarely computable.

Common Usage

- [Class Diagram](#)
- [Use Case Diagram](#)
- [Object Diagram](#)
- [Composite Structure Diagram](#)



Further Information

- [Dependency](#)

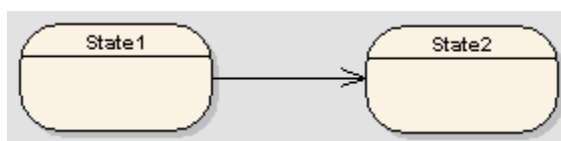
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

"A dependency that indicates a historical or process relationship between two elements that represent the same concept without specific rules for deriving one from the other."

5.4.27 Transition

[Common Usage](#)..|.. [OMG UML Specification](#)



A *transition* defines the logical movement from one state to another. The transition can be controlled through the following properties:

Trigger	Specifies the event instigating the transition.
Guard	An expression that is evaluated after an event is dispatched, but before the corresponding transition is triggered. If the guard is true at that time, the transition is enabled; otherwise, it is disabled.
Effect	Specifies an optional activity to be performed during the transition.

Note: Fork and join segments can have neither triggers nor guards.

Common Usage

- [State Machine Diagram](#)



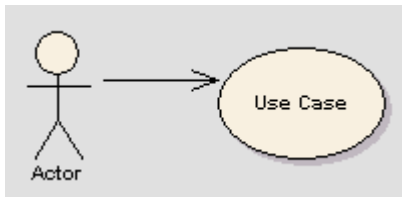
OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 17) states:

"A relationship between two states indicating that an object in the first state will perform certain specified actions and enter the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to fire."

5.4.28 Use

[Common Usage](#) | [OMG UML Specification](#)



A use link indicates that one element requires another to perform some interaction. The Usage relationship does not specify how the target supplier is used, other than that the source client uses it in definition or implementation. A usage relationship is a sub-typed dependency relationship.

You are likely to use the use relationship mostly in Use Case diagrams to model how actors use system functionality (Use Cases), or to illustrate usage dependencies between classes or components.

Note: To depict a usage dependency on a Class or Component diagram, draw a dependency connector. Right-click on the dependency, and select **Dependency Stereotypes | use**.

Common Usage

- [Use Case Diagram](#)
- [Class Diagram](#)
- [Component Diagram](#)



OMG UML Specification

The OMG UML specification (*UML 2.0 Superstructure*, p. 111) states:

"A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. In the metamodel, a Usage is a Dependency in which the client requires the presence of the supplier."

Part



6 Modeling with UML

What is UML?

UML is the *Unified Modeling Language*, a standard that defines the rules and notation for specifying software systems. The notation supplies a rich set of graphic elements for modeling object-oriented elements, and the rules say how those elements may be connected and used.

UML is not a prescriptive process for creating software systems - it does not supply a method or process, simply the language. You can therefore use UML in a variety of ways to specify and develop your software engineering project. Enterprise Architect supports many different kinds of UML elements (as well as some custom extensions). Together with the links and connections between elements, these form the basis of the model.

In addition to the base UML elements, the modeling environment can be extended using UML Profiles. A Profile is a set of stereotyped and tagged elements that together solve some modeling problem or scenario. Examples are UML Profiles for modeling XML Schema or Business Process Modeling.

This section contains a list of the basic UML building blocks. Familiarity with these elements and their purpose will greatly assist your modeling tasks. The full meaning of each of the elements is beyond the scope of this user guide.

For more information, see the UML section at <http://www.omg.org> or any good **book**

Modeling can be defined as the act of representing something, usually on a smaller scale or with reduced detail. Using EA, modeling can be more specifically described as the act of graphically representing a business process or software system. A model thus created can be used to emphasize a certain aspect of the system being represented and record, document and communicate its detail. A study of such a model can allow insight or understanding of the system.

Modeling with UML

UML is the *Unified Modeling Language*, a standard that defines the rules and notation for specifying software systems. The notation supplies a rich set of graphic elements for modeling object-oriented elements, and the rules say how those elements may be connected and used.

UML is not a prescriptive process for creating software systems - it does not supply a method or process, simply the language. You can therefore use UML in a variety of ways to specify and develop your software engineering project. Enterprise Architect supports many different kinds of UML elements (as well as some custom extensions). Together with the links and connections between elements, these form the basis of the model.

In addition to the base UML elements, the modeling environment can be extended using UML Profiles. A Profile is a set of stereotyped and tagged elements that together solve some modeling problem or scenario. Examples are UML Profiles for modeling XML Schema or Business Process Modeling.

This section contains a list of the basic UML building blocks. Familiarity with these elements and their purpose will greatly assist your modeling tasks. The full meaning of each of the elements is beyond the scope of this user guide. For more information, see the UML section at <http://www.omg.org> or any good book on UML.

Recommended Reading:

In addition to the UML Specification available from the OMG, two books which provide excellent introductions to UML are:

- *Schaum's Outlines: UML* by Bennet, Skelton and Lunn. Published by McGraw Hill. ISBN 0-07-709673-8
- *Developing Software with UML* by Bern Oestereich. Published by Addison Wesley. ISBN 0-201-36826-5

More Information

Modeling with UML:

- [Working with UML Diagrams](#)
- [Working with UML Elements](#)
- [Working with UML Connections](#)
- [UML Stereotypes](#)
- [MDG Technologies](#)

- [UML Profiles](#)
- [UML Patterns](#)
- [Requirements Management](#)
- [Element Relationship Matrix](#)

See also:

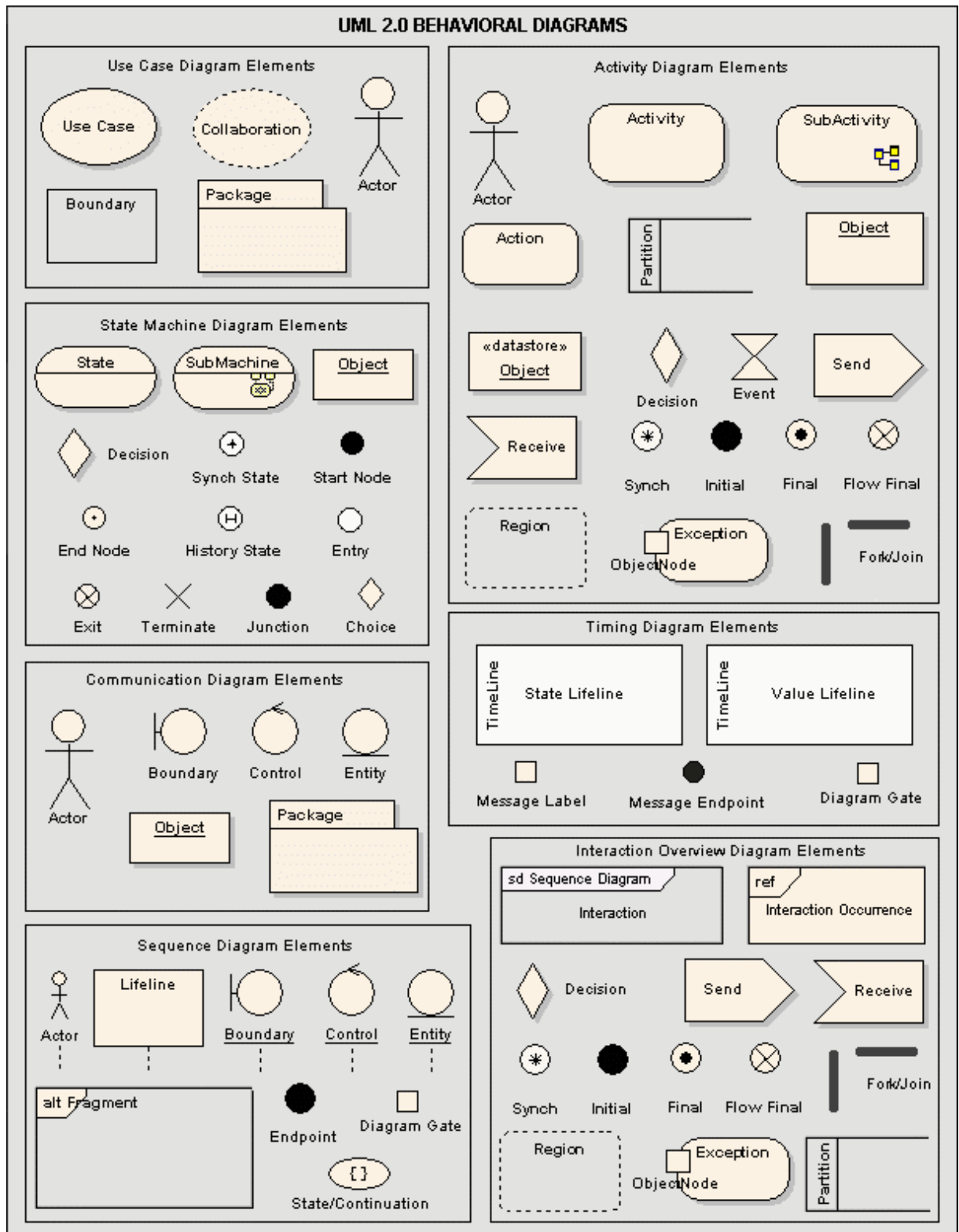
- [UML Diagrams](#)
- [UML Elements](#)
- [UML Connections](#)

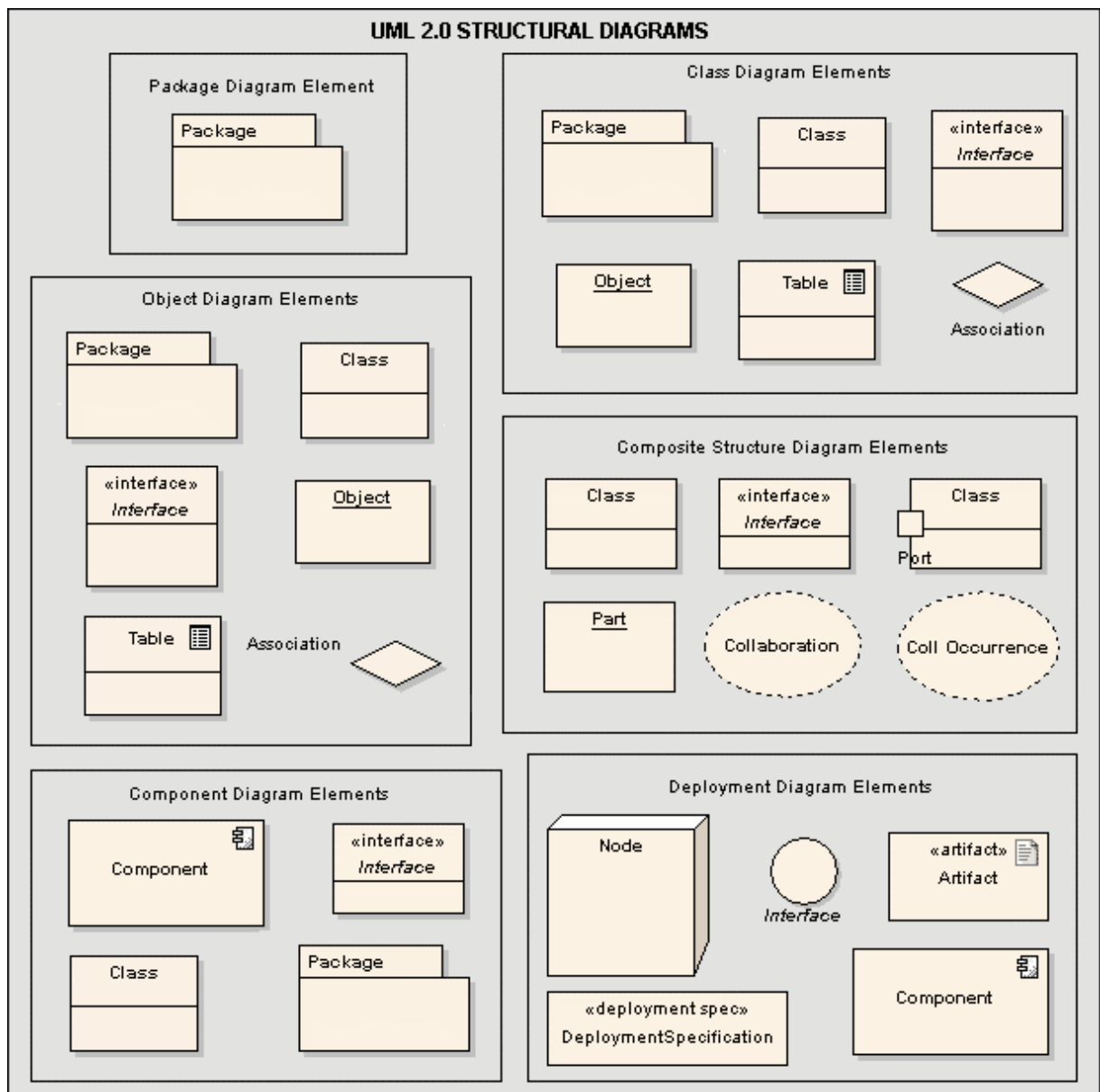
6.1 Working with UML Diagrams

UML Diagrams are collections of project elements laid out and inter-connected as required using Enterprise Architect supports several kinds of UML diagrams (as well as some custom extensions).

Together with the Enterprise Architect elements and connections, these form the basis of the model. Diagrams are stored in packages and may have a parent object (optional). Diagrams may be moved from package to package.

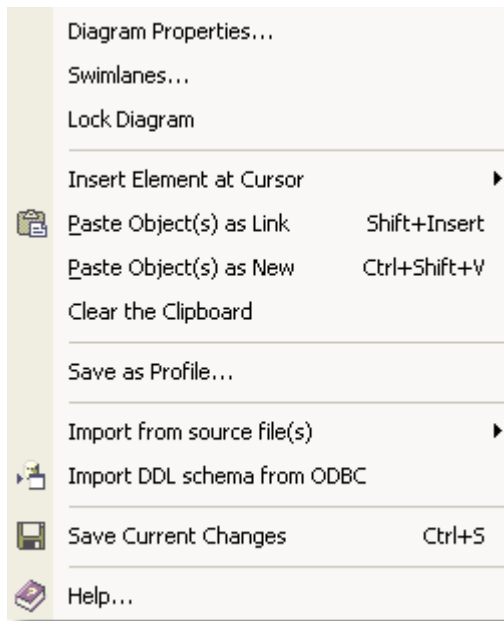
Basic UML diagram elements:





6.1.1 Diagram Context Menu

Right click on the diagram window when a valid diagram is selected to open the diagram context menu. Not all menu options shown below will appear on all diagram context menus.



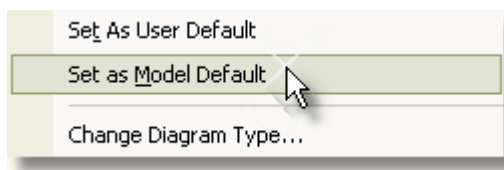
The diagram context menu allows you to:

- View the diagram properties dialog
- Protect a diagram from inadvertent changes (lock diagram)
- Insert various elements into a diagram (select from the following - Boundary, Note, Text, Diagram Notes, Hyperlink)
- Paste element(s) as a link or as new elements
- Import source code (reverse engineer) - *not available in the Desktop edition*
- Import database tables from an ODBC data source - *not available in the Desktop edition*
- Save the current diagram
- View the EA Help files

6.1.2 Set the Default Diagram

A project may have a *default diagram*. If set, this diagram will load when EA first opens the model. It is often convenient to place hyperlinks to other diagrams and resources on the default diagram, thus creating a kind of Home Page for your model.

To set the currently active diagram as the model default, select *Set as Model Default* from the *Diagram* menu.



Tip: Once you have specified a default diagram, the 'Home' icon on the diagram toolbar will take you to that diagram.



The Enterprise Architect Example Model Home Page:

Guide to the Example Model





Welcome to this Guide to the Example Model

This section entitled "Guide to the Example Model" has been constructed to explain the example. It has been kept separate from the example so as not to "contaminate" the example with explanatory notes that would not typically occur in a model for a 'real' system. You may however consider constructing such a guide to your own model for the benefit of your clients or for newcomers to the team. The example is deliberately incomplete to show you a snapshot of a real project "in construction".

How to Use this Guide

The guide can be used to explore the example model. The model is deliberately quite large and while you are free to explore the model under the project root node "Messenger" this guide will help you locate parts of the model that are complete or of particular interest.

Double Click on the following Links:

-  [Explore Example Diagrams](#)
-  [Explore Example Models](#)
-  [Explore Example Views](#)
-  [Explore Example Documentation](#)

About the Example

The example model contains a project for a system that is "in construction". The example is of a project called "Messenger". The background to this project is the need for customers (External and Internal to the organization) to send and receive messengers in any format from their desktop. The worker should not need to get up from her desk but should be able to send emails, sms, faxes and conventional mail messages from a single interface. The system should take care of how the message is delivered and the customer should be able to set their contact preferences. The project is being built for an international organization whose head office is in Zurich and which has regional offices in Seattle and Singapore. The intention is that it will be built and used internally to give the organization customer service, efficiency and productivity gains.

6.1.3 Common Diagram Tasks

This section details some of the common tasks associated with managing diagrams:

- [New Diagrams](#)
- [Deleting Diagrams](#)
- [Diagram Properties](#)
- [Renaming Diagrams](#)
- [Copy Diagram Element](#)
- [Convert Linked Element to Local Copy](#)
- [Z Order Elements](#)
- [Copy Image to Disk](#)
- [Copy Diagram Image to Clipboard](#)
- [Change Diagram Type](#)
- [Open a Package](#)

- [Duplicate a Diagram](#)
- [Set Feature Visibility](#)
- [Insert Diagram Properties Note](#)
- [Autosize Elements](#)
- [Drop Elements from the Project Browser](#)
- [Pasting from the Project Browser](#)
- [Place Related Elements on Current Diagram](#)
- [Swimlanes](#)
- [Using the Image Manager](#)
- [Show Realized Interfaces for a Class](#)
- [Label Menu Section](#)

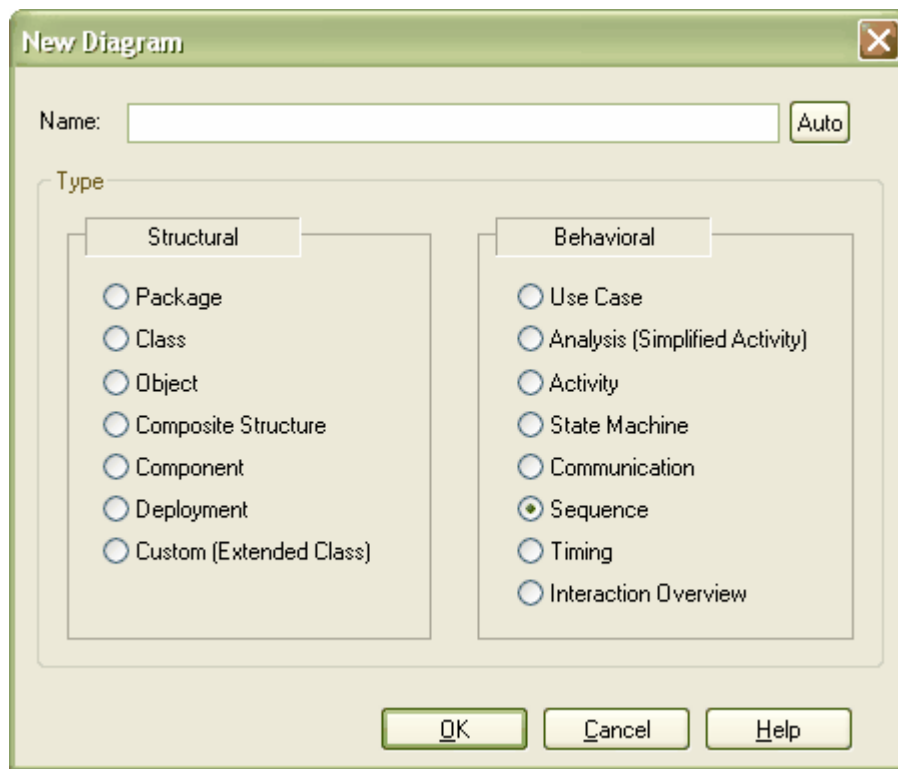
6.1.3.1 New Diagrams

To add a new diagram, follow these steps:

1. In the *Project Browser* window, select the appropriate package or element under which you wish to place the diagram.
2. Right click to open the context menu and select *New Diagram*.
3. Enter the name of the new diagram in the dialog provided and select the type of diagram you require.
4. Press *OK* to create your new diagram.



Alternatively, select *New Diagram* from the Diagram menu.



Note: the type of diagram type determines the default toolbar associated with the diagram and whether it can be set as a child of another element in the Project Browser (eg. a sequence diagram under a use case).

See also: [UML Diagrams](#).

6.1.3.2 Deleting Diagrams

Warning: There is no Undo feature in Enterprise Architect for deleting diagrams, so be very sure that you wish to delete a diagram before you do.

Note: When you delete a diagram, you do not delete the elements on the diagram from the model.

Delete a Diagram

To delete a diagram from your model, follow the steps below:

1. In the Project Browser, right click on the diagram you wish to delete.
2. Select the *Delete <diagram name>* option.
3. Confirm you wish to delete the diagram by pressing **OK**.

6.1.3.3 Diagram Properties

You may set several properties of a diagram using the *Diagram Properties* dialog (shown below). Some influence the display and some are logical attributes that appear in the documentation.

There are several options for opening the *Diagram Properties* dialog for a given diagram:

- Select *Properties* from the *Diagram* menu to open the properties dialog for the currently active diagram.
- Right click on the required diagram in the Project Browser and select *Properties* from the context menu.
- Right click on the background of the open diagram and select *Diagram Properties* from the context menu.
- Double click in the background of the open diagram.

In the *Diagram Properties* dialog you may set various properties including name, author and version information, zoom factor, paper size and layout, diagram notes, and various appearance attributes. Once you have made any necessary changes, press *OK* to save and exit.

See the following topics for further information:

- [Set AppearanceOptions](#)
- [Document Options](#)
- [Scale Image to Page Size](#)
- [Set the Page Size](#)
- [Set Visible Class Members](#)

The screenshot shows the 'Interaction Diagram: Interactions' dialog box. It is organized into several sections:

- General Properties:**
 - Name: Interactions
 - Author: John Redfern
 - Version: 1.2
 - Zoom: 100
 - Stereotype: (empty)
 - Created: 10/08/2004
 - Modified: 10/08/2004 11:22:54 AM
- Page Size:**
 - A4 Sheet, 210- by 297-millimeters
 - Landscape
 - Advanced...
- Appearance Options:**
 - Use Stereotype Icons
 - Show Page Border
 - Show Table Owner
 - Use Alias if Available
 - Hide Property Methods
 - Hide Collaboration Numbers
 - Hide Element Stereotype
 - Hide Qualifiers
 - Highlight Foreign Objects
 - Show Package Contents
 - Show Details on Diagram
 - Show Sequence Notes
 - Hide Additional Parents
 - Hide Relationships
 - Hide Stereotype on Features
 - Hide Attributes
 - Hide Operations
 - Show Tags
 - Show Requirements
 - Show Constraints
 - Show Testing
 - Show Maintenance
- Visible Class Members:**
 - Public
 - Protected
 - Private
 - Package
- Show Parameter Detail:**
 - Full Details
- Document Options:**
 - Exclude image from RTF Documents
 - Document each contained element in RTF
 - Set Layout Style
- Notes:**
 - (Empty text area)

Buttons at the bottom: Apply, OK, Cancel, Help.

6.1.3.4 Renaming Diagrams

To rename a diagram, follow the steps below:

1. Open the *Diagram Properties* dialog by double clicking on the diagram background.
2. Enter the new *Name* for your diagram.
3. Press *OK* to save changes.

6.1.3.5 Copy Diagram Element

To copy a diagram element, follow the steps below:

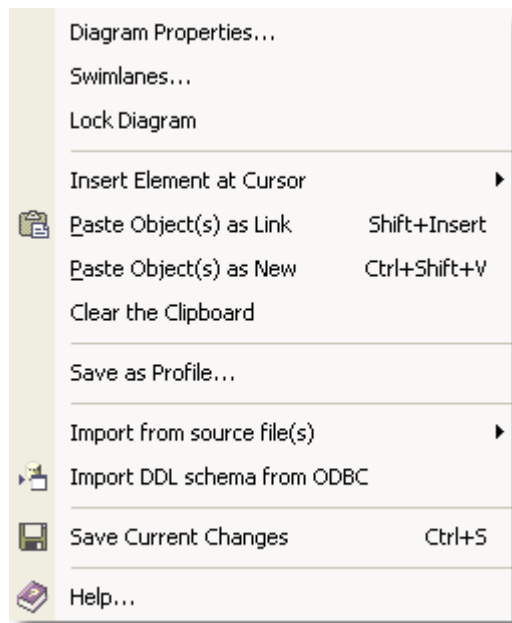
1. Select the element(s) to copy.
2. For multiple elements, right click to open the context menu and select *Copy all selected Objects to Clipboard*. Alternatively, press *Ctrl+C*.
3. For single elements, select the *Edit | Copy* command or alternatively press *Ctrl+C*.

Paste Diagram Element(s)

To paste diagram element(s), follow the steps below:

1. Open the diagram to paste into.
2. Right click on the diagram background to open the diagram context menu.
3. Select whether to paste as a new element (complete new element) or as a link to the element (reference).

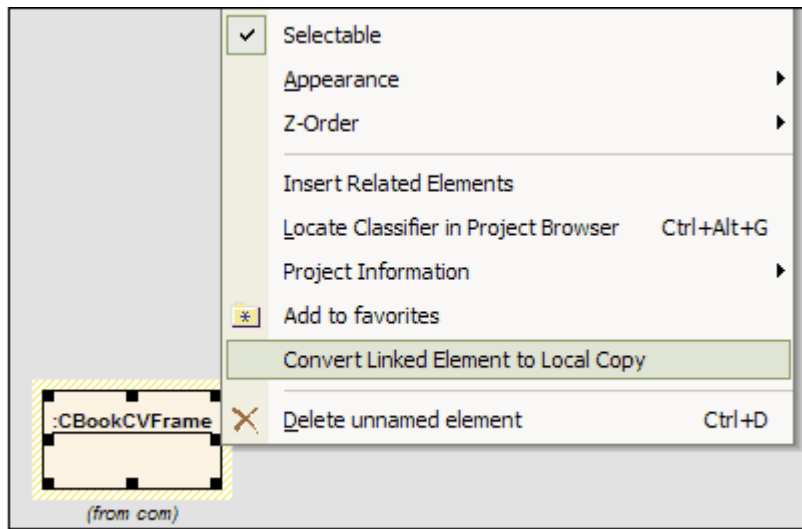
Note: *Diagram elements may be pasted as links or as new elements. Select the most appropriate action for your model.*



6.1.3.6 Convert Linked Element to 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 the element to bring up its context menu.
3. Click the Convert Linked Element to Local Copy menu option.



4. The Element will be changed to a local copy and will be placed in the appropriate package

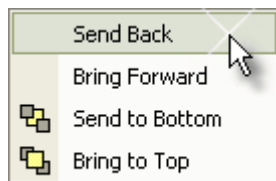
6.1.3.7 Z Order Elements

Z Order refers to an element's depth in the diagram hierarchy, and thus influences which elements appear in front of others and which appear behind.

Set the Z Order

To set the Z Order of an element, follow the steps below:

1. Select the element in the diagram view.
2. From the *Element* | *Z order* submenu, select the operation to perform (see below).
3. The element will be moved to the new position in the diagram hierarchy.



6.1.3.8 Copy Image to Disk

You can copy a diagram image to a disk file in the following formats:

- Windows bitmap (256 color bitmap)
- GIF image
- Windows Enhanced Metafile (standard metafile)
- Windows Placeable Metafile (older style metafile)
- PNG format
- JPG
- TGA

Copy a Diagram Image to File

To copy a diagram image to file, follow the steps below:

1. Open the diagram you wish to save.
2. From the *Diagram* menu, select *Save Image to File*. (Alternatively, press *Ctrl+T*).
3. When prompted, enter a name for the file and select an image format.
4. Press *OK*.

Note: EA will clip the image size to the smallest bounding rectangle that encompasses all diagram elements.

6.1.3.9 Copy Diagram Image to Clipboard

Diagram images may be copied onto the clipboard and pasted directly into MS Word or other applications.

Copy an Image to the Clipboard

To copy an image to the clipboard, follow the steps below:

1. Open the diagram you wish to copy.
2. From the *Diagram* menu, select *Save Image to Clipboard*. (Alternatively, press *Ctrl+B*).
3. Press *OK*.

The diagram has been copied to the clipboard and can now be pasted into compatible applications. You can set the clipboard format in the *Local Options* dialog (*Tools | Options*). Bitmap or Metafile format are supported.

6.1.3.10 Change Diagram Type

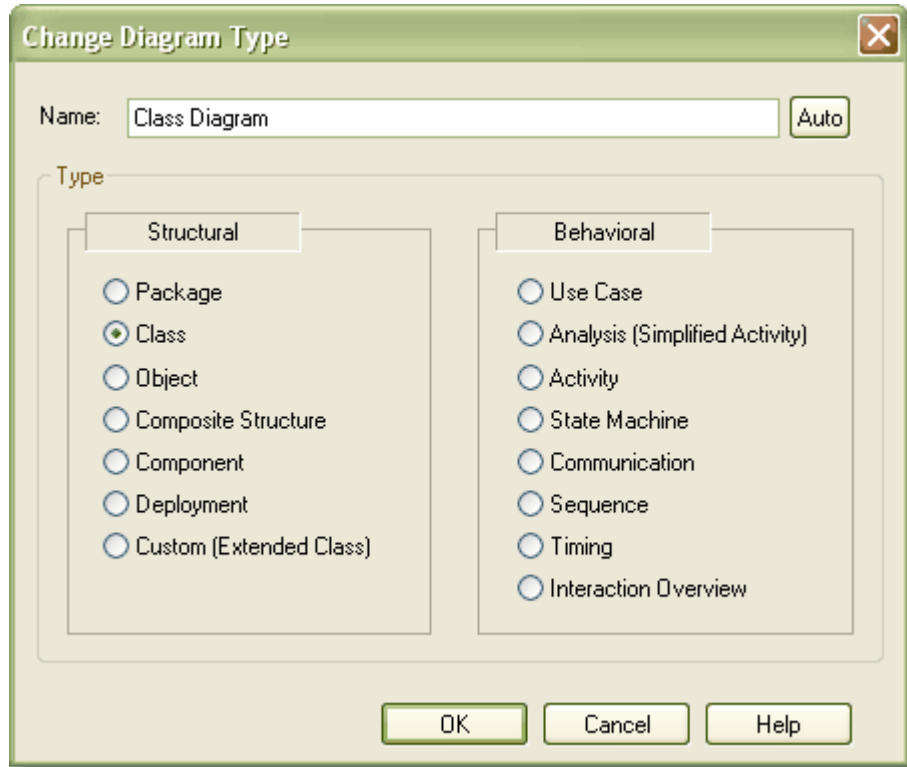
You may change a diagram to another type if required. This is useful if you have either made a mistake in selecting the diagram type to begin with, or if the purpose and nature of a diagram changes during analysis.

Change a Diagram Type

To change a diagram type, follow the steps below:

1. Open the diagram you wish to change.

2. From the *Diagram* menu, select *Change Diagram Type...*
3. In the *Change Diagram Type* dialog, select the required diagram type.
4. Press *OK* to save changes.



Note: Some diagram types will not transfer to others - for example you cannot change a Class Diagram into a Sequence diagram.

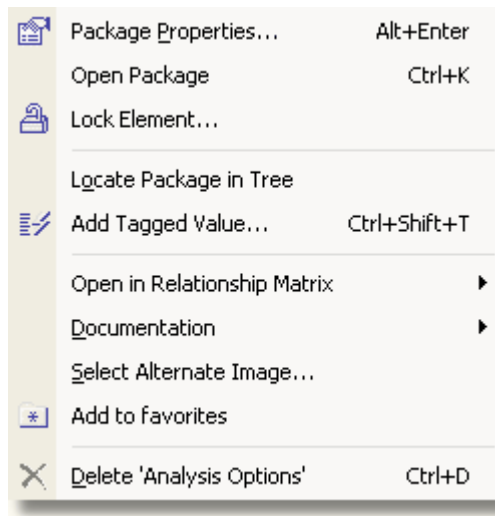
6.1.3.11 Open a Package

You may open a package up for viewing by using the right click context menu on a package in a diagram. EA will select the first available diagram in the package (selected in alphabetical order) - and open it.

Open a Package

To open a package, follow the steps below:

1. Open a diagram which shows the package you wish to open.
2. Right click on the package element to open the context menu.
3. Select *Open Package*. Alternatively, use the keyboard shortcut *Ctrl+K*.



EA will find the default diagram and open it for you.

6.1.3.12 Duplicate a Diagram

Enterprise Architect makes it easy to duplicate a complete diagram - either with links back to the original diagram elements, or with complete copies of all elements in the diagram.

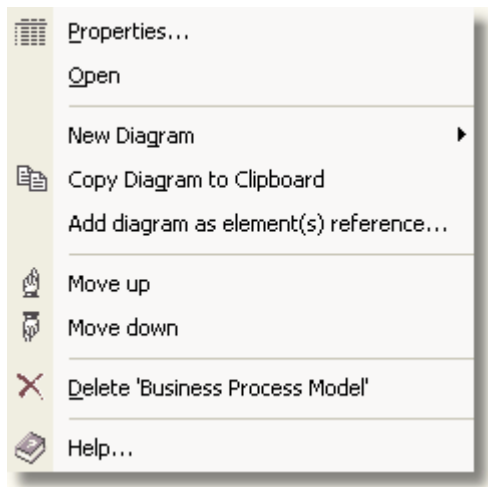
When you copy a diagram in 'shallow' mode, the elements in the new diagram are only *linked* to the originals - so if you change the properties of one, the other will reflect those changes. If you copy the diagram in 'deep' mode, then all elements are duplicated completely - so that changing an element on one does not affect the other.

Element position and size should be independent in both copy modes.

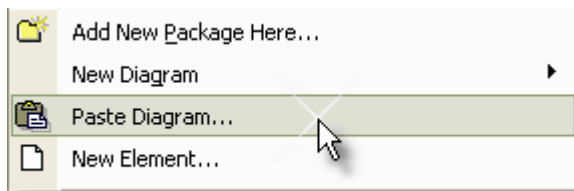
Duplicate a Diagram

To duplicate a diagram, follow the steps below:

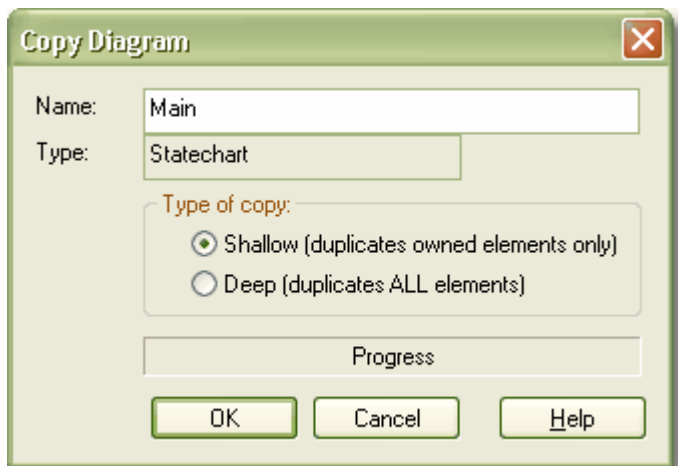
1. In the Project Browser, select the diagram you wish to copy.
2. From the *Diagram* menu, select *Save Diagram to Clipboard* -OR- select *Copy Diagram to Clipboard* from the right-click context menu. Alternatively, press *Ctrl+B*.



3. Navigate to the package you wish to host the new diagram, and right click to open the context menu.
4. Select *Paste Diagram...*



5. In the *Copy diagram* dialog, enter a *Name* for the new diagram.
6. Check the *Type of copy* you require - either linked elements (shallow copy) or complete copies of the originals (deep copy).



7. Press *OK*.

EA will automatically create the new diagram, link or create new elements and arrange as in the original diagram. All links are also copied between diagram elements where appropriate.

6.1.3.13 Set Feature Visibility

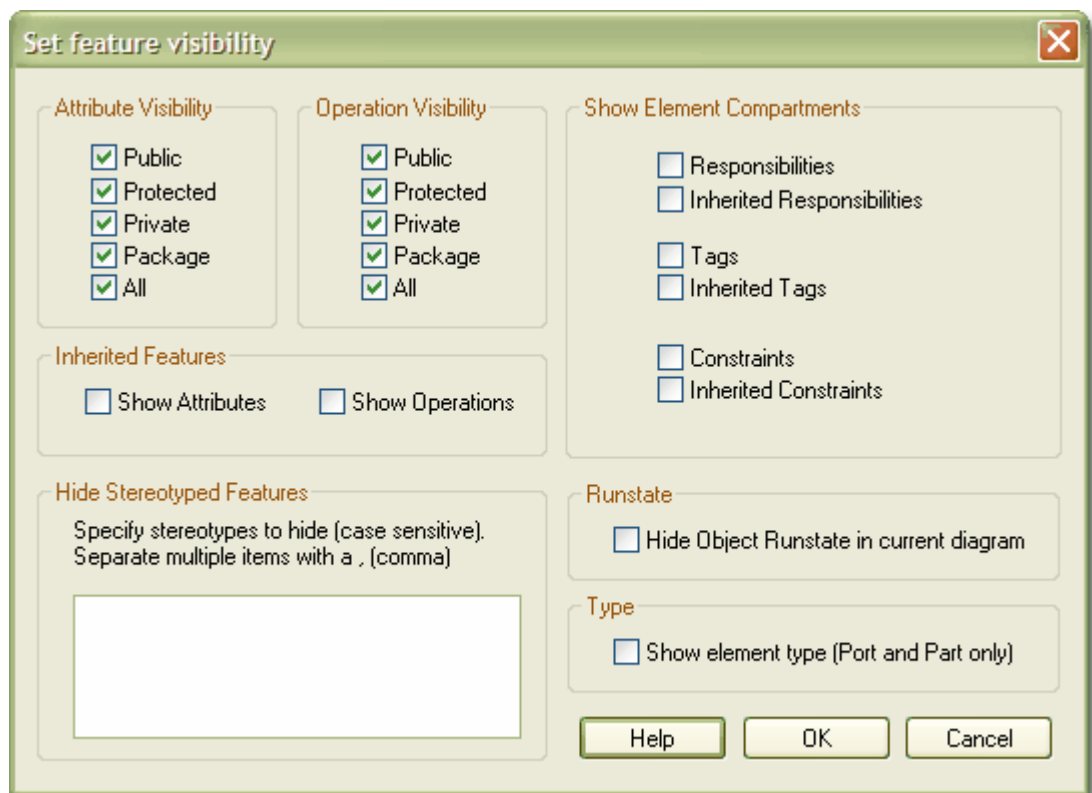
Enterprise Architect allows you to set the visibility of attributes and operations on a per class basis / per diagram basis or the visibility of attributes and operations on a package diagram. For example you can hide all protected attributes, or all private operations or any other combination of attributes and operations. The visibility you set will apply only to the current diagram - so a class may appear in one diagram with all elements displayed - and in another its elements may be hidden.

It is possible to show inherited attributes, operations, requirements, constraints and tagged values for elements that support those features. When EA displays inherited features, it creates a merged list from all generalized parents and from all realized interfaces. If a child class redefines something found in a parent, the parent feature is omitted from the Merge List.

Customize Feature Visibility

To customize feature visibility, follow the steps below:

1. From the *Element* menu, select *Set feature visibility*. Ctrl+Shift +Y is a convenient keyboard shortcut -OR- use right click on an element in a diagram to bring up its context menu. Then mouse over the *Element Feature* menu item and select *Specify Feature Visibility*.
2. In the *Set feature visibility* dialog, check the features you want visible and clear those you do not. Also set whether EA should display inherited features as well as directly owned ones.



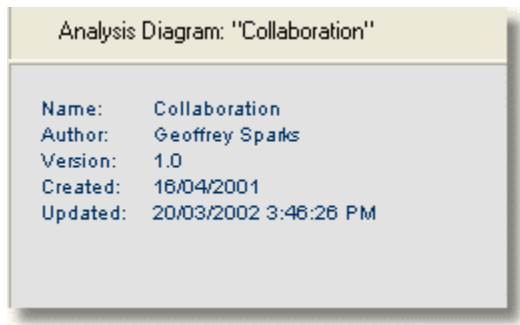
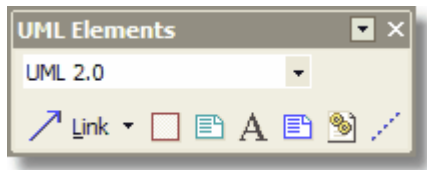
3. Press *OK* to save changes.

EA will redraw the diagram with the appropriate level of feature visibility.

6.1.3.14 Insert Diagram Properties Note

Properties of a diagram may be displayed on screen within a custom text box. You can move this text box around and change its appearance. You cannot change what the text box says.

The Diagram Properties note is the third last button on the *UML Elements* toolbar.



6.1.3.15 Autosize Elements

You can autosize a group of elements in a diagram to a best fit.

Autosize a Group of Elements

To autosize a group of elements, follow the steps below:

1. Select all the elements you wish to resize (*Ctrl+A* selects all).
2. Use the right click context menu option *Autosize all selected*. Alternatively, press *Alt+Z*.

EA will resize elements where necessary.

Note: *Not all elements will resize - some like Events and Use Cases remain the same*

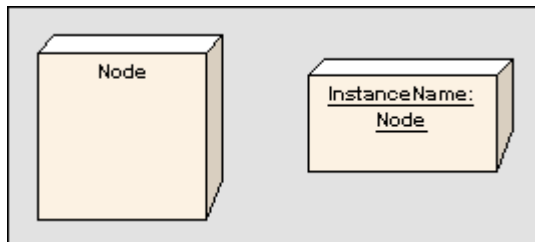
Note: *The minimum size EA will set is the default size for the element - usually around 100x100 - but different for some elements.*

6.1.3.16 Drop Elements from the Project Browser

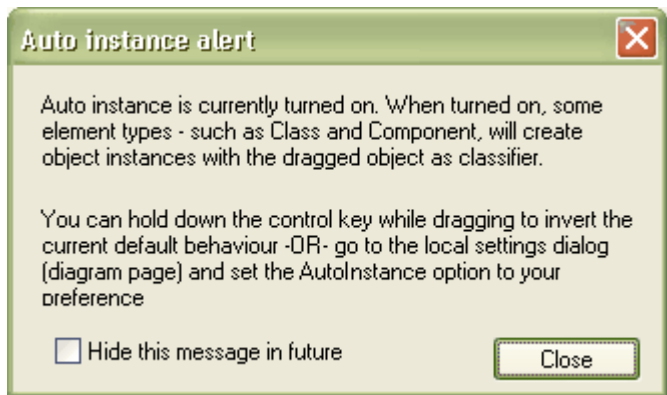
When you drag an element from the Project Browser onto a diagram, for some elements there are two possible paste options. Elements that are classifiers and support instances of themselves at runtime may be dropped as either a link to the classifier itself, or as a new instance of the classifier.

The example below shows a linked element on the left (a Node) and an instance of the Node on the right. Note that the Node instance is drawn like a simple element with the `:<ElementName>` displayed. If you name your instance it will display `<InstanceName>:<ElementName>`

If you are working on instance diagrams - such as a collaboration diagram, you will often want to quickly drag and drop classes and elements from the browser onto a diagram as an instance of the classifier. If you are working in class diagrams, you may prefer to drop links to the classifier itself. There are a couple of settings available to simplify this process.

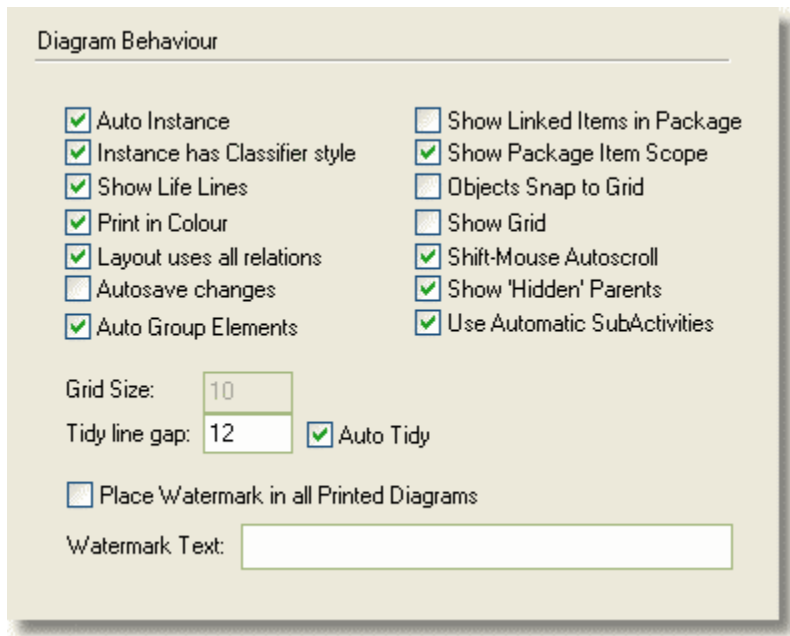


Note that EA will display a warning about this behavior (see below), indicating what is happening. Check the *Hide this message in future* option to prevent seeing this dialog.



There are two important things to remember here:

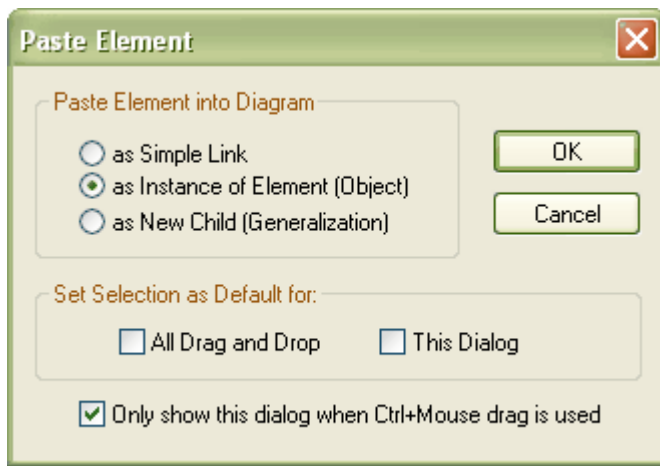
- Hold down the **Ctrl** key while dragging and dropping an element into a diagram (see [Pasting from the Project Browser](#)).
- Hold down either the **Ctrl** key or the **Shift** key to select multiple elements into a diagram (See [Pasting Multiple Elements from the Project Browser](#)).
- To change the default behavior, go to the **Local Options** dialog (**Tools | Options**), and select **Diagram | Behavior**. To enable or disable Auto Instance, check or clear the **Auto Instance** option (shown below). Remember - the **Ctrl** key will invert the current default.



6.1.3.17 Pasting from the Project Browser

You can paste an element from the Project Browser into the current diagram.

When you drag an element from the Project Browser onto the current diagram with the **Ctrl** key held down, EA will prompt you to select the type of paste action to carry out.



Three options are available:

1. Paste the element as a link. In this case the element will appear in the current diagram as a simple reference back to the original source element. Changes to the element in the diagram will affect all other links to this element.
2. Paste as an instance of the element. If the element can have a classifier (eg. an Object, Sequence

instance, Node instance etc.) you can drop the element as an instance of the source element, with the classifier pre-set to the original source. This is useful when creating multiple instances of a class in a sequence diagram, or in a Collaboration diagram.

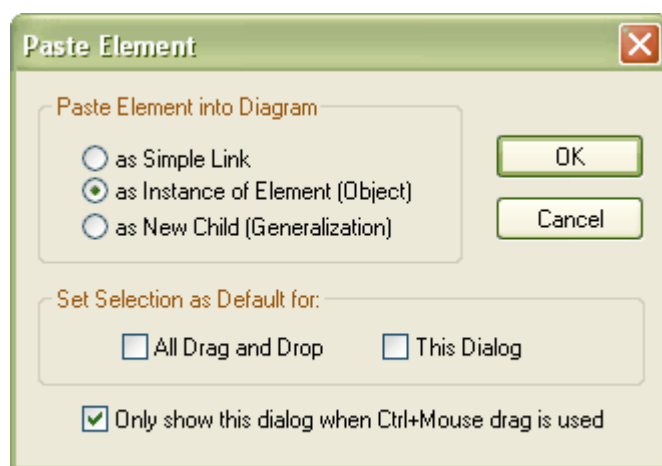
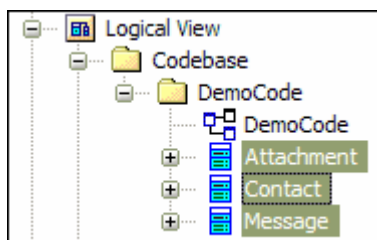
3. Create as a child of the source element. This will automatically create a new class - which you will be prompted to name - and create a Generalization link back to the source. This is very useful when you have a class library or framework from which you inherit new forms (eg. a Hashtable may be pasted as "MyHashtable" and automatically become a child of the original Hashtable). Used with the Override parents operations and features, this is a quick way to create new structures based on frameworks like the Java SDK and the .NET SDK.

Note: In order to make use of the *Only show this dialog when Ctrl + Mouse drag is used* the *Auto Instance* check box must be selected in the *Local Options | Diagram Behavior* dialog window. To access this items go to the *Tools* menu and select *Options*.

6.1.3.17.1 Pasting Multiple Items from the Project Browser

You can paste multiple elements from the Project Browser into the current diagram.

To select multiple elements, use the mouse to select items from the Project Browser and either hold down the *Ctrl* key to add a single item to the selection of multiple elements one at a time, or hold down the *Shift* key to perform a selection of all of the elements between the first selected item in the Project Browser tree and the last item selected. You can then drag the selected elements from the Project Browser onto the current diagram with the *Ctrl* key held down, EA will prompt you to select the type of paste action to carry out.



Three options are available:

1. Paste the element as a link. In this case the element will appear in the current diagram as a simple

reference back to the original source element. Changes to the element in the diagram will affect all other links to this element.

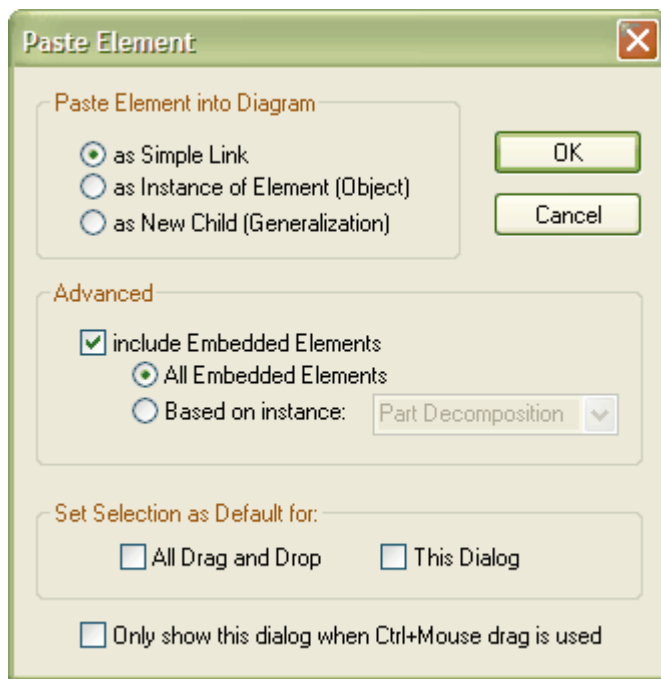
2. Paste as an instance of the element. If the element can have a classifier (eg. an Object, Sequence instance, Node instance etc.) you can drop the element as an instance of the source element, with the classifier pre-set to the original source. This is useful when creating multiple instances of a class in a sequence diagram, or in a Collaboration diagram.
3. Create as a child of the source element. This will automatically create a new class - which you will be prompted to name - and create a Generalization link back to the source. This is very useful when you have a class library or framework from which you inherit new forms (eg. a Hashtable may be pasted as "MyHashtable" and automatically become a child of the original Hashtable). Used with the Override parents operations and features, this is a quick way to create new structures based on frameworks like the Java SDK and the .NET SDK.

Note: In order to make use of the *Only show this dialog when Ctrl + Mouse drag is used* the *Auto Instance* check box must be selected in the *Local Options | Diagram Behavior* dialog window. To access this items go to the *Tools* menu and select *Options*.

6.1.3.17.2 Pasting Composite Elements

Several additional options are available to the user when pasting composite elements from one diagram to another.

When you drag a composite element from the Project Browser onto the current diagram with the *Ctrl* key held down, EA will prompt you to select the type of paste action to carry out with the composite element.



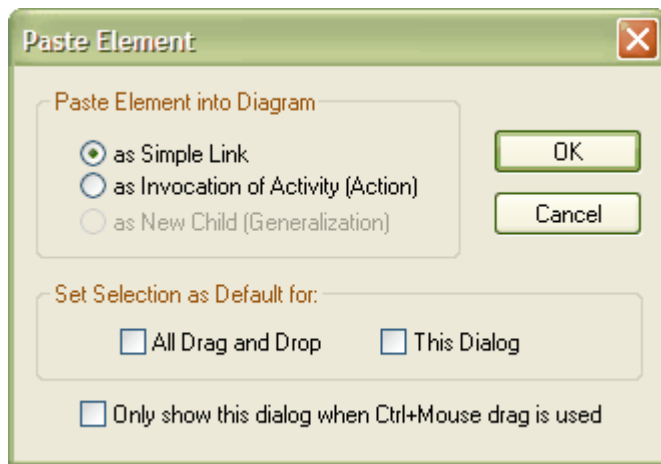
Two Advanced options are available for pasting composite elements, this requires that the *include Embedded Elements* checkbox be selected:

1. The *All Embedded* elements option, this will paste all of the composite elements embedded elements.
2. The *Based on instance* option, this will paste only the elements contained in a specific in an instance of the composite element. Select the appropriate instance from the drop down menu.

6.1.3.17.3 Pasting Activities

You can paste an activity from the Project Browser into the current diagram.

When you drag an activity from the Project Browser onto the current diagram with the *Ctrl* key held down, EA will prompt you to select the type of paste action to carry out.



Two options are available:

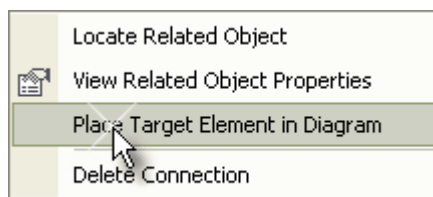
1. Paste the activity as a link. In this case the activity will appear in the current diagram as a simple reference back to the original source activity. Changes to the activity in the diagram will affect all other links to this activity.
2. Paste as an invocation of the activity.

Note: In order to make use of the *Only show this dialog when Ctrl + Mouse drag is used* the *Auto Instance* check box must be selected in the *Local Options | Diagram Behavior* dialog window. To access this items go to the *Tools* menu and select *Options*.

6.1.3.18 Place Related Elements on Current Diagram

If you wish to find and place related elements on the current diagram, use the Relationships tab on the Properties window.

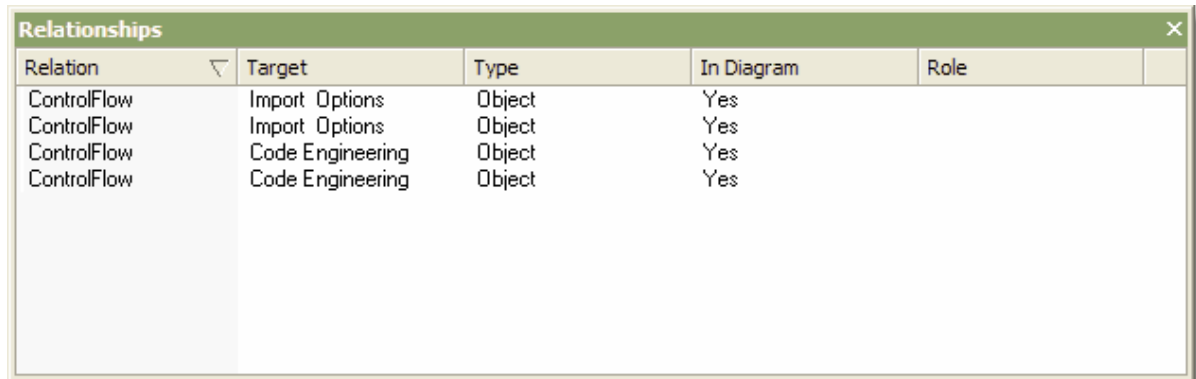
Right click on any link in the list to open the context menu



If an element is not present in the current diagram, the context menu will contain the *Place Target Element in Diagram* option. This is useful when you are building up a picture of what an element interacts with - especially when reverse engineering an existing code base.

Select the *Place Target Element in Diagram* option. Move the mouse cursor to the desired position in the

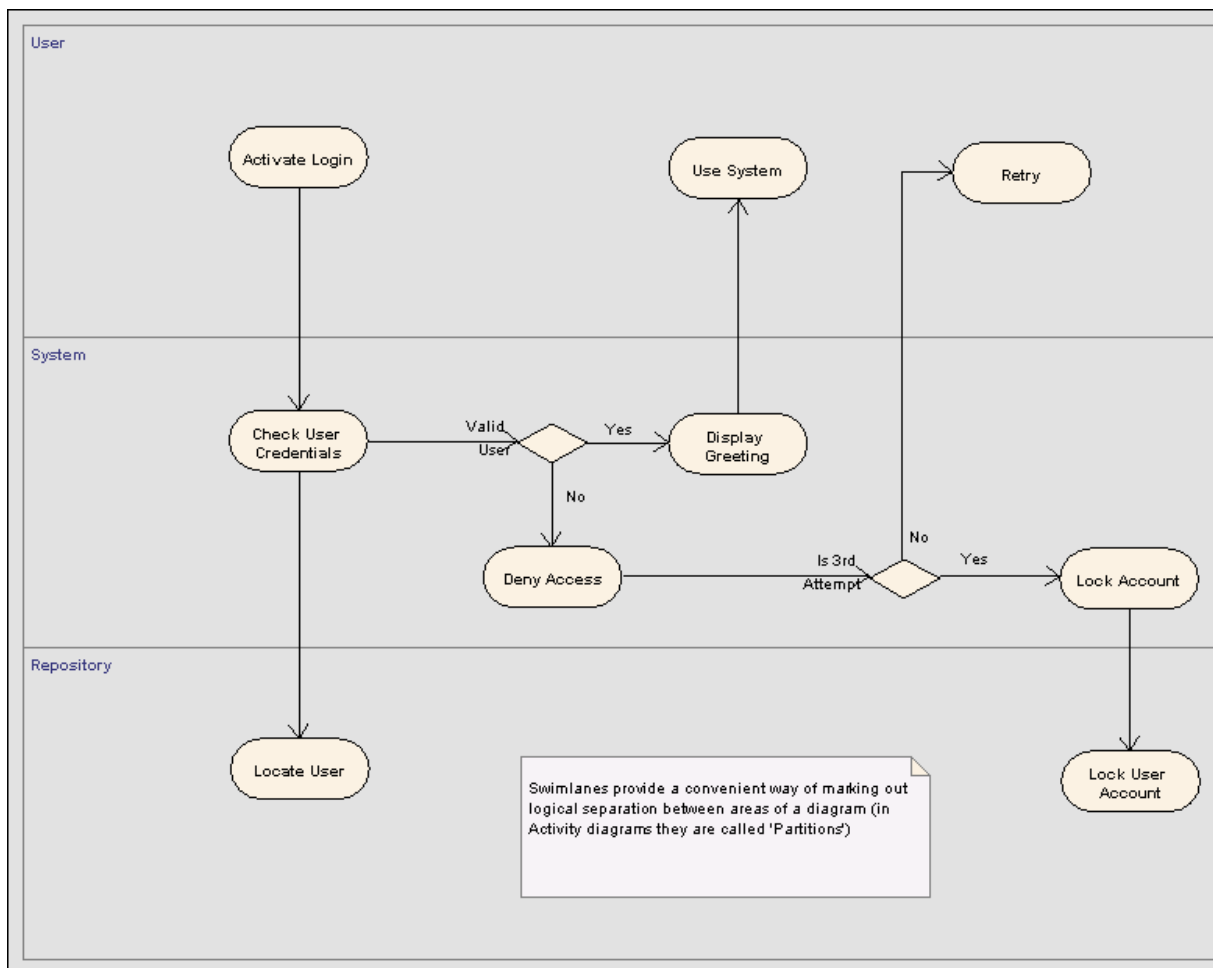
diagram and left click to place the element. Alternatively, press the *Esc* key to cancel the action.



Relation	Target	Type	In Diagram	Role
ControlFlow	Import Options	Object	Yes	
ControlFlow	Import Options	Object	Yes	
ControlFlow	Code Engineering	Object	Yes	
ControlFlow	Code Engineering	Object	Yes	

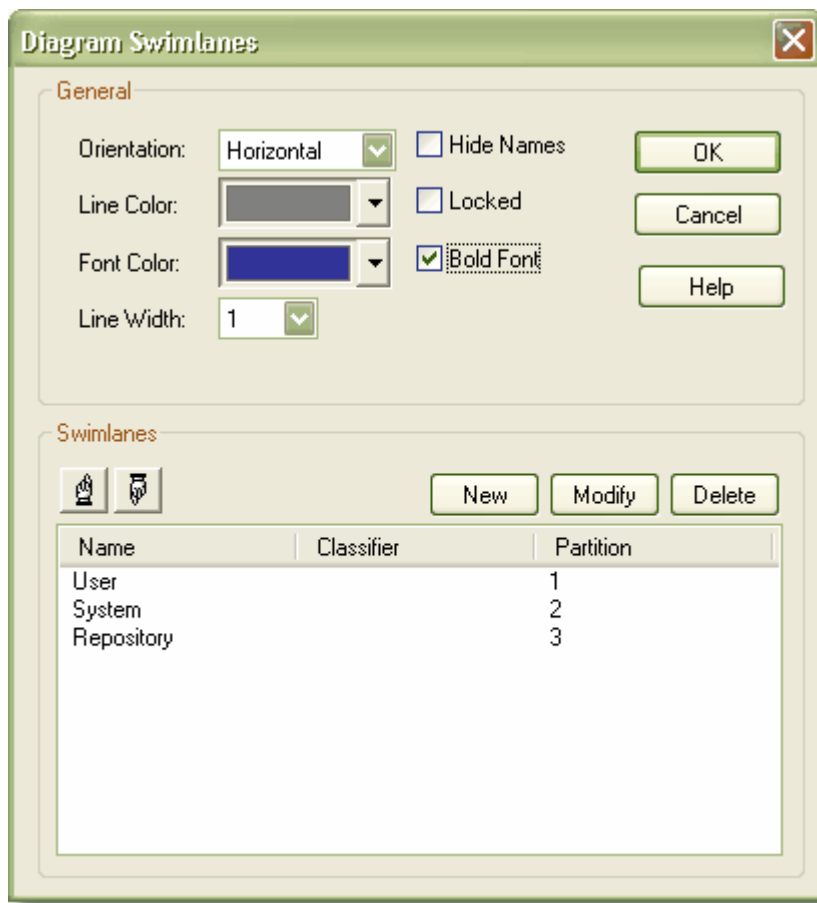
6.1.3.19 Swimlanes

EA diagrams support Swimlanes for all diagram types. Swimlanes are vertical or horizontal bands in a diagram that divide the diagram into logical areas or *partitions*. In the example below the activities relating to particular entities within the model (eg. the User, or the back end Repository) are placed within a containing swim lane to indicate their association.



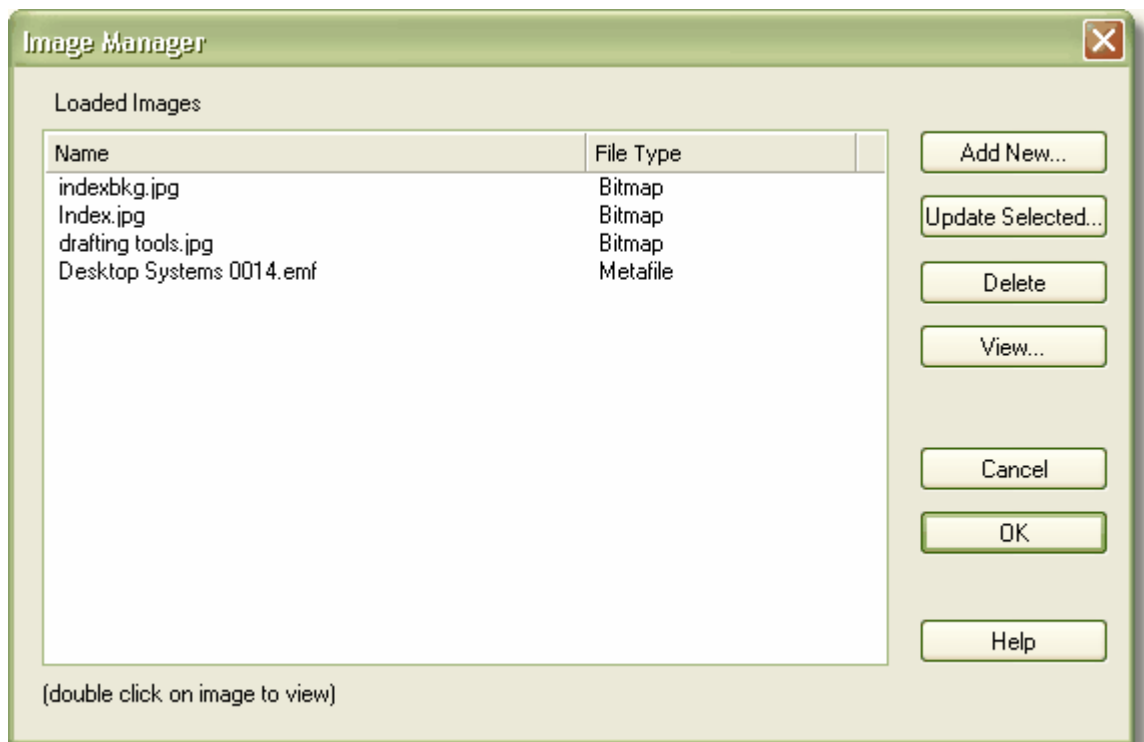
To manage swimlanes, use the *Diagram Swimlanes* dialog, accessed by selecting *Configure Swimlanes...* from the *Diagram* menu.

This dialog allows you to set the orientation (vertical or horizontal), line color and width of the swimlanes, and also specify font color, bold font or hidden names. You may also lock the swimlanes to prevent further movement. Use the *New*, *Modify* and *Delete* buttons to change aspects of the selected swim lane. Use the up and down buttons to switch the order of swimlanes within the diagram.



6.1.3.20 Using the Image Manager

The Image Manager allows users of EA to place alternate images in diagrams rather than standard UML elements. Often you may wish to place a custom background image on a diagram - or have a UML element display a custom image (eg. a Router or PC).



The following options are available when using the Image Manager dialog:

Dialog Button	Description
Add New	If you don't have images loaded, use the <i>Add New</i> button to search for and import another image. Images may be in BMP, PNG, EMF, WMF, TGA, PCX or JPG format. Internally they are always stored in PNG format to conserve space.
Update Selected	You may highlight an image and update the associated image using the <i>Update Selected</i> button.
Delete	If you delete an image you are first warned about how many elements use the image. If you continue, those elements will have the information about the associated image deleted and they will revert back to their previous appearance.
View	To preview an image, select the image name in the <i>Loaded Images</i> section of the dialog then press the <i>View</i> button or double click on the image name.
Cancel	The <i>Cancel</i> button closes the image manager
OK	Confirms the selection of the alternate image for the element selected in the diagram.
Help	Opens the help file to this topic.

See also:

- [Select Alternate Image](#)
- [Create Custom Diagram Background](#)
- [Import Image Library](#)

6.1.3.20.1 Select Alternate Image

By using the Image Manager a custom image may be used for elements on diagrams. To perform this operation use the following procedure:

1. Right click on the element within the diagram to bring up its context menu, mouse over the *Appearance* item and from the submenu choose the *Select Alternate Image* option, or select the element in the diagram and use the *Ctrl + Shift + W* hotkey combination.
2. Use the *Image Manager* dialog to select an appropriate image as the alternate for the element. For more information regarding the use of the Image Manager see the [Using the Image Manager](#) topic.
3. Press the *OK* button when have selected the desired image.

Note: *If you are creating many elements of the same type that will have this particular image, then you should use a [custom stereotype](#) with an associated metafile.*

6.1.3.20.2 Create Custom Diagram Background

By using the Image Manager a custom, non-tiled background may be created for diagrams. To perform this operation use the following procedure:

1. Create a Boundary object from the *Use Case* section of the [UML Toolbox](#), do not use the Boundary object from the other sections of the UML toolbox.
2. Stretch the Boundary to a size which will be able to contain all of the elements that you intend to place on the diagram and drag it to the edges of the diagram workspace.
3. Right click on the Boundary element and from the context menu mouse over the *Z-Order* menu item, select *Send to Bottom* from the *Z-Order* submenu, this will ensure that the boundary object is not displayed in front of any other element in the diagram.
4. Press the *Ctrl + Shift + W* hotkey combination or right click on the boundary element to bring up the elements context menu, mouse over the *Appearance* menu item and from the sub menu choose the *Select Alternate Image* option.
5. Use the *Image Manager* dialog to select an appropriate image as the diagram background, ensure that the image size is large enough to span the desired size of the diagram background. For more information regarding the use of the Image Manager see the [Using the Image Manager](#) topic.
6. Press the *OK* button when have selected the desired image.

6.1.3.20.3 Import Image Library

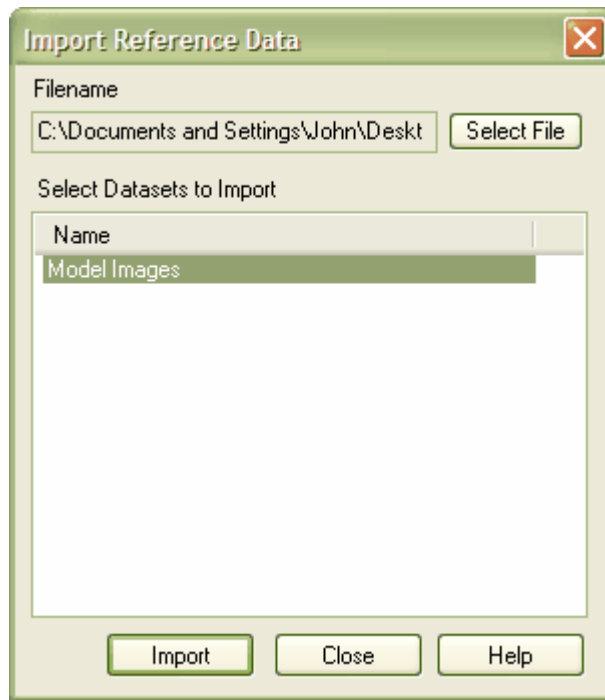
Using the Image Library allows users of EA to create attractive diagrams with custom images. Sparx Systems now offers the a bundled clip art collection of UML based images as an Imported Image Library available from http://www.sparxsystems.com.au/Image_Library.htm. Image libraries allow the user to import a collection of images into the Image Manager in one process.

Note: *Images contained within the Image Library are copyright of Sparx Systems and are only available for use when used in conjunction with EA and supplied on the basis that they will not be used under any other circumstance.*

Importing an Image Library

To import an Image Library users will need a suitable Image Library file. To import the Image Library, follow the steps below:

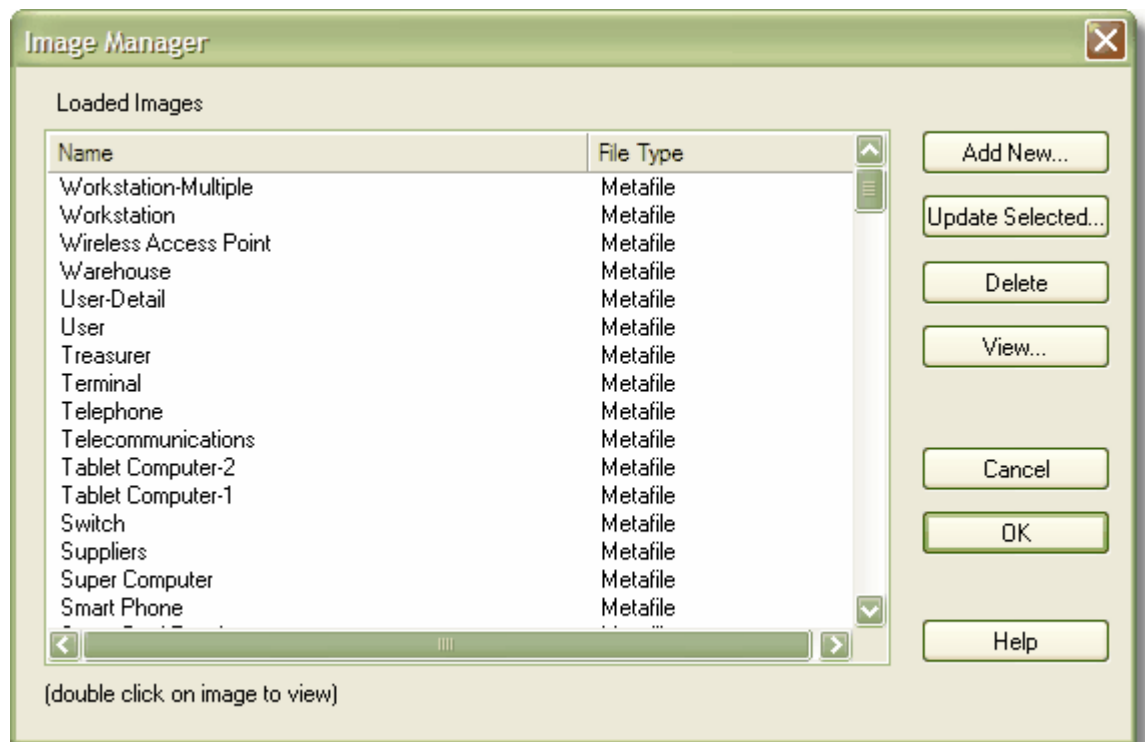
1. Download the Image Library from http://www.sparxsystems.com.au/Image_Library.htm.
2. From the *Tools* menu select the Import reference data option. The Import Reference Data dialog will open.
3. Locate the XML Image Library file to import using the Select File button, The file name will be *ImageLibrary.xml* in the directory which you saved the file to.
4. Select the data set containing the Image Library. Then press the *Import* button.



Using the Image Library

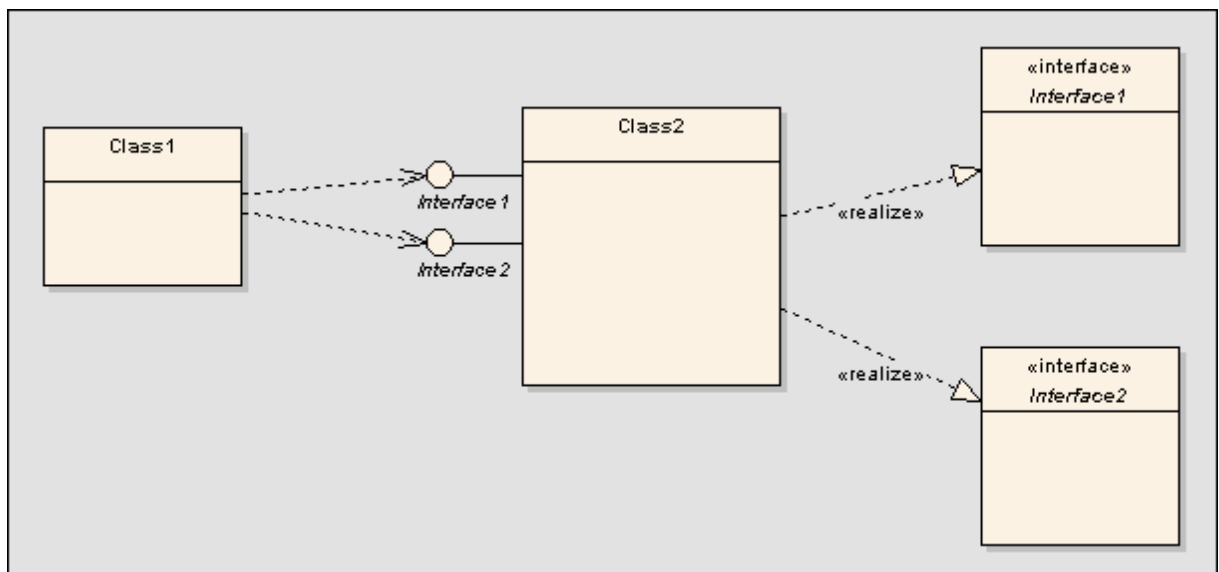
To use the images contained within the Image Library use the following Instructions:

1. Create a diagram that you wish to associate with the images contained in the Image Library.
2. Select the element that you wish to change from the default appearance to one of the images contained from within the library.
3. Press the *Ctrl + Shift + W* hotkey combination or right click the selected element to bring up its context menu and then from the *Appearance* submenu choose the *Select Alternate Appearance* option.
4. From the *Image Manager* dialog select the appropriate image by highlighting the image from the *Name* field and then press the *OK* button.



6.1.3.21 Show Realized Interfaces for a Class

You can display each interface directly realized by a class as a "lollipop" style interface node which protrudes from the left-hand side of the class. Connectors can be directly attached to the node, indicating usage of the interface part of the class or component. See the example below:



In this example, Class2 realizes Interface1 and Interface2. This is represented by the interface nodes protruding from the left hand side of the class. Class1 is dependent on these two interfaces, which is shown by the dependency arrows linking to the nodes.

To show nodes for the interfaces a class realizes as in the above diagram, right click on the class and select *Show Realized Interfaces*. This setting will only apply to the selected class, and can be changed at any time.

6.1.3.22 Label Menu Section

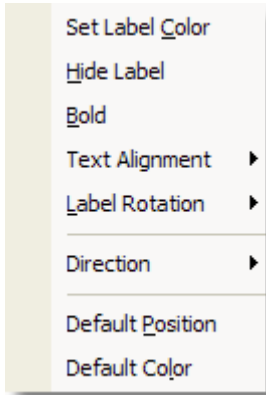
The Labels menu associated with connectors contains the following options:

Menu Option	Description
Set Label Color	Enables the user to specify a color for the label.
Hide Label	Hide the label, to unhide the label use the Set Label Visibility option.
Bold	Sets the label font to bold.
Text Alignment	Aligns the text within the label text area, the options available from the submenu allows the user to specify left, center and right alignment.
Label Rotation	The submenu allows the label to be orientated in the horizontal or vertical planes, with the vertical plane offering the option of clockwise are anticlockwise positioning.
Direction	Sets a small arrow at the end of the label pointing wither to the label source or destination dependent upon selection from the available options.
Default Position	Moves the label to the default location.
Default Color	Sets the label color to the initial default color.

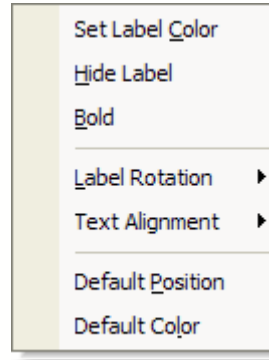
The Labels menu associated with embedded elements contains the following options

Menu Option	Description
Set Label Color	Enables the user to specify a color for the label.
Hide Label	Hide the label, to unhide the label use the show label option in the Embedded Elements context menu.
Bold	Sets the label font to bold.
Text Alignment	Aligns the text within the label text area, the options available from the submenu allows the user to specify left, center and right alignment.
Label Rotation	Allows the label to be orientated in the horizontal or vertical planes, with the vertical plane offering the option of clockwise are anticlockwise positioning.
Default Position	Moves the label to the initial default location.
Default Color	Sets the label color to the default color.

Example context Label Menu for a Connector:



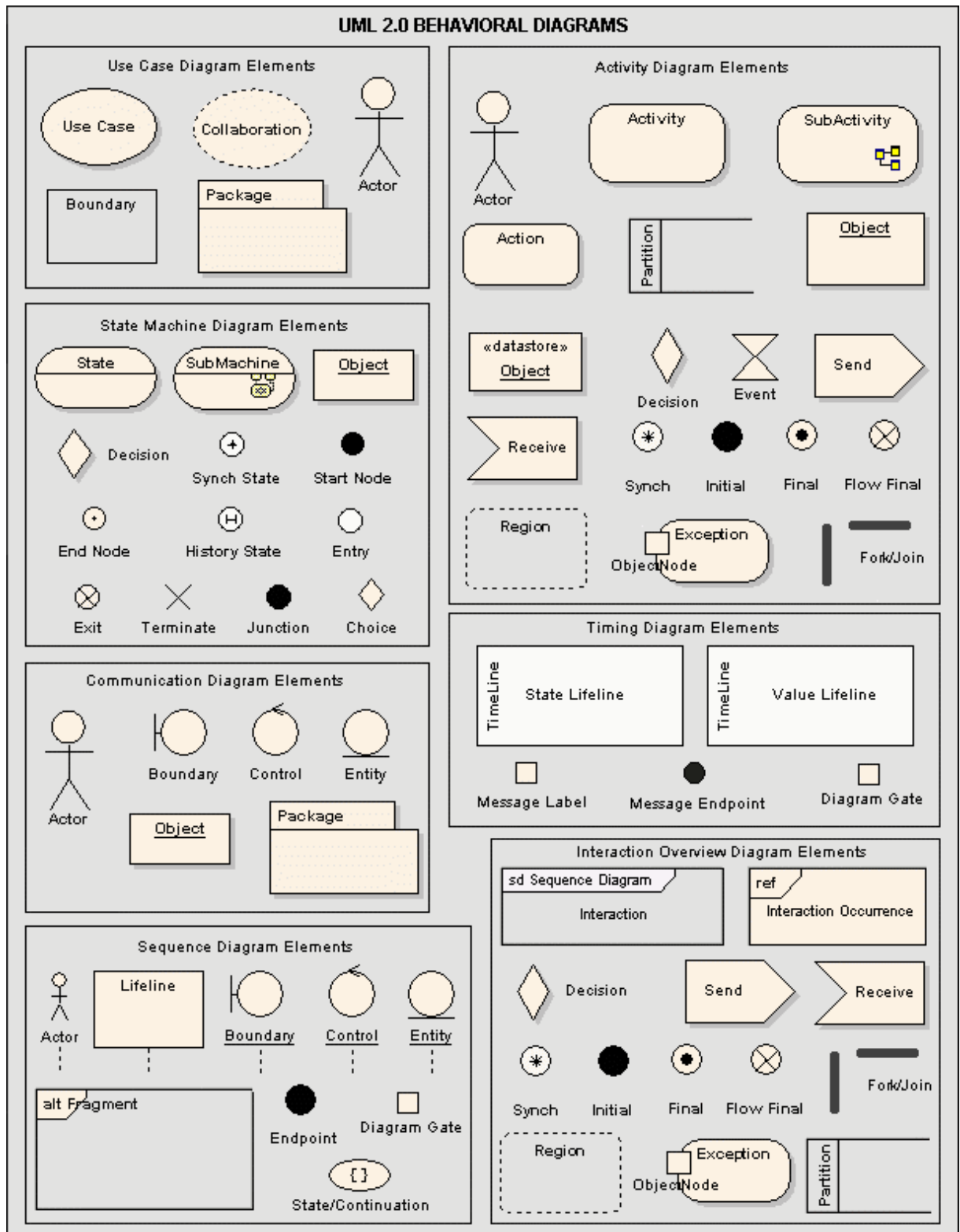
Example Context Label Menu for an Embedded Element:

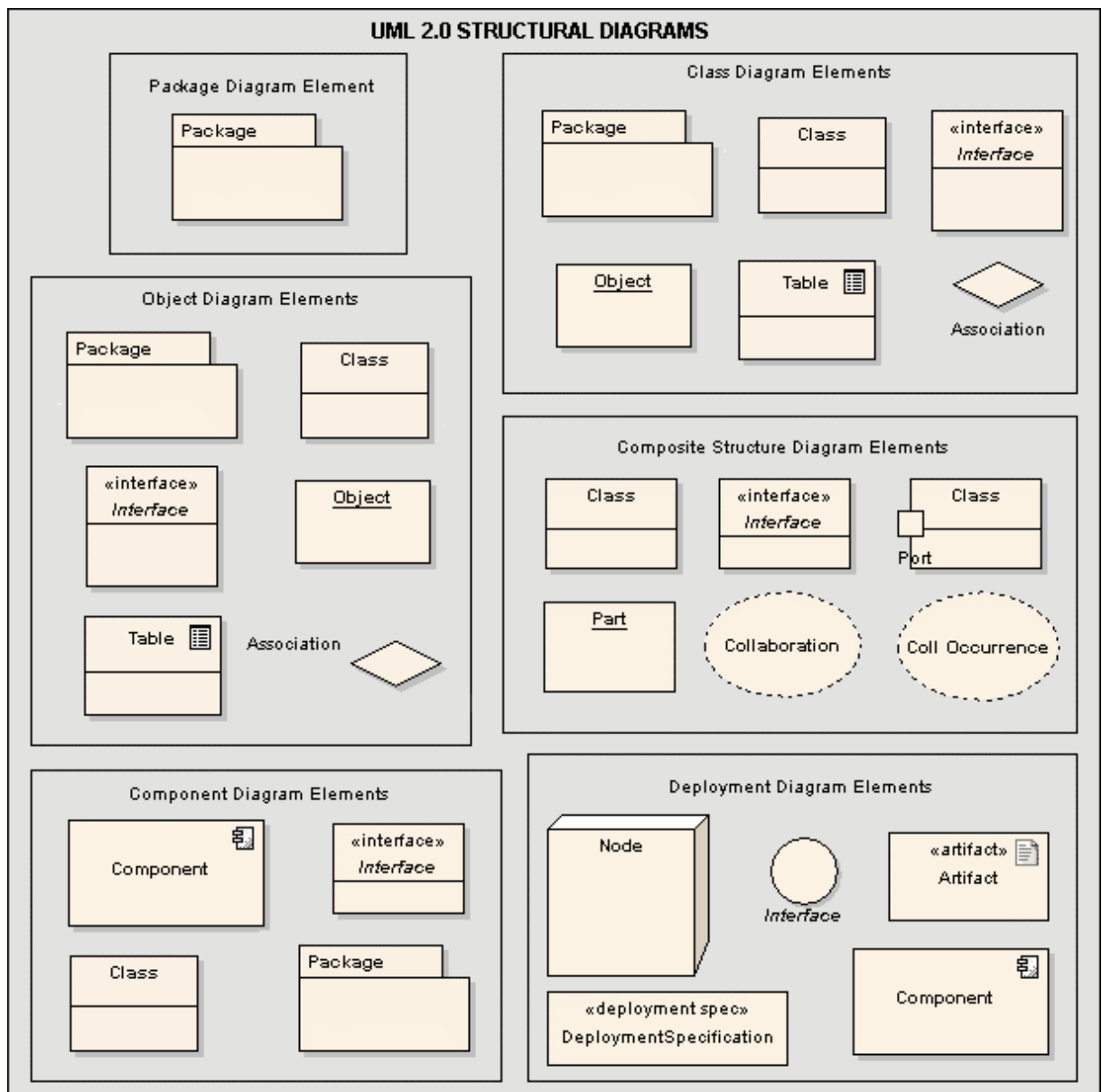


6.2 Working with UML Elements

UML Models are constructed from elements, each of which has its own meaning, rules and notation. Elements can be used at different stages of the design process for different purposes.

The basic elements for UML 2.0 are:





6.2.1 Element Context Menu

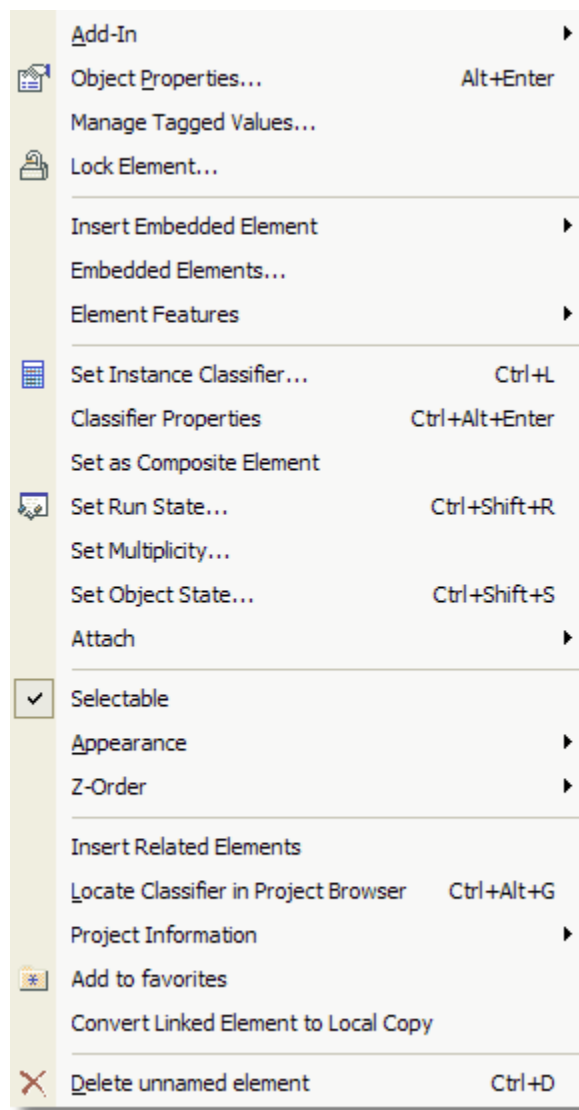
Right click on a single element in a diagram to open the element context menu. If two or more elements are selected, a different, [multiple selection context menu](#) will be displayed.

The element context menu is split into eight distinct sections:

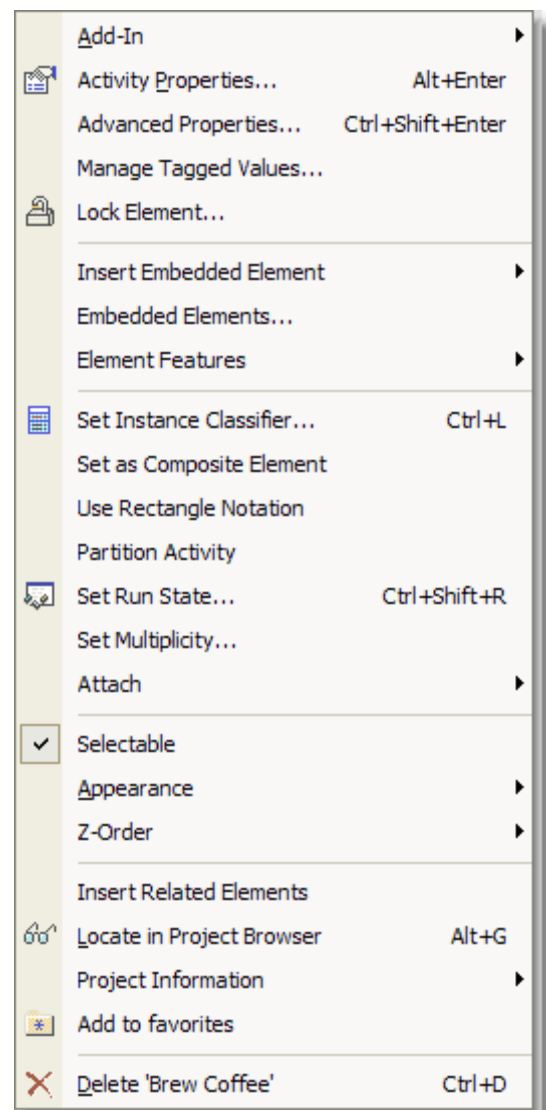
- [Zicom Mentor](#) - this will only show in the first section if you have Zicom Mentor installed and registered.
- [Properties](#)
- [Embedding and features](#)
- [Type](#)
- [Code Engineering](#)
- [Appearance](#)
- [Common Actions](#)
- Delete - you can delete the element from this menu option.

Note: Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

Example Context Menu for a Class:



Example Context menu for an Activity:



6.2.1.1 Properties Menu Section

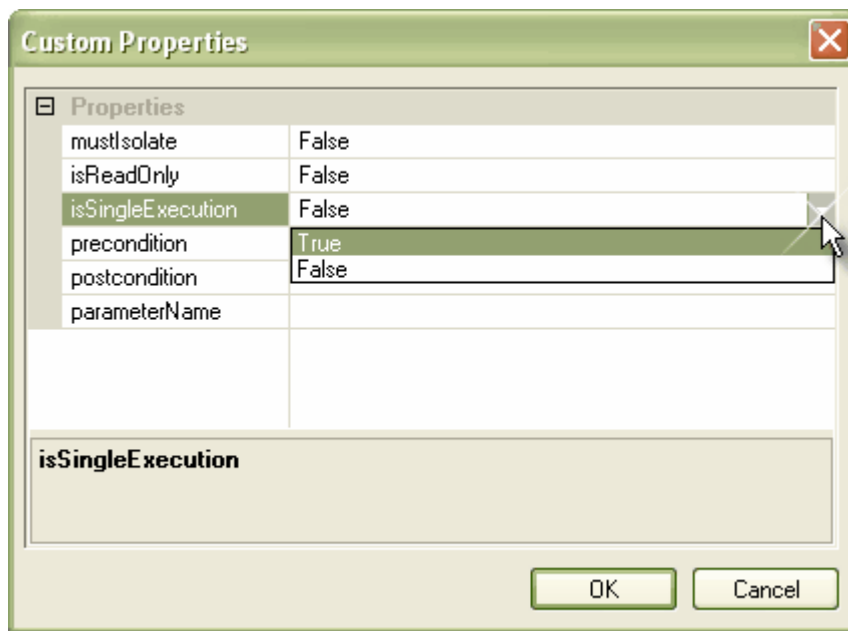
The Properties menu section on the element context menu may contain the following options:

Menu Option	Description
<Element type> Properties	Opens the Properties dialog for the selected element.
Open Default Diagram	If the element is (or has been) a Composite Element , this opens the child diagram the element points to.
Advanced Properties	Opens the Advanced Properties dialog.
Lock Element	Locks the element so it can't be edited. It can be unlocked by selecting Lock Element again.

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.1.1 Advanced Properties Dialog

Some element and connectors feature the *Advanced Properties* option in their context menu. The example below is the Advanced Properties dialog belonging to an Activity element. Properties will vary depending on the kind of element or connector.



As shown above, properties can be altered either with a drop down list or by typing in the field to the right of the property.

6.2.1.2 Embedding and Features Menu Section

The Embedding and Features menu section on the element context menu may contain the following options:

Menu Option	Description
Insert Embedded Elements	The sub-menu will show a list of any elements which can be embedded into this element type. In the case of a class, the list consists of: <ul style="list-style-type: none">• Port• Show Realized Interfaces• Show Dependent Interfaces
Embedded Elements	Opens the Embedded Elements window.
Element Features	See table below for sub-menu options.
Show Labels	This item is only present when the embedded element has a hidden label, choose this item to unhide the label.

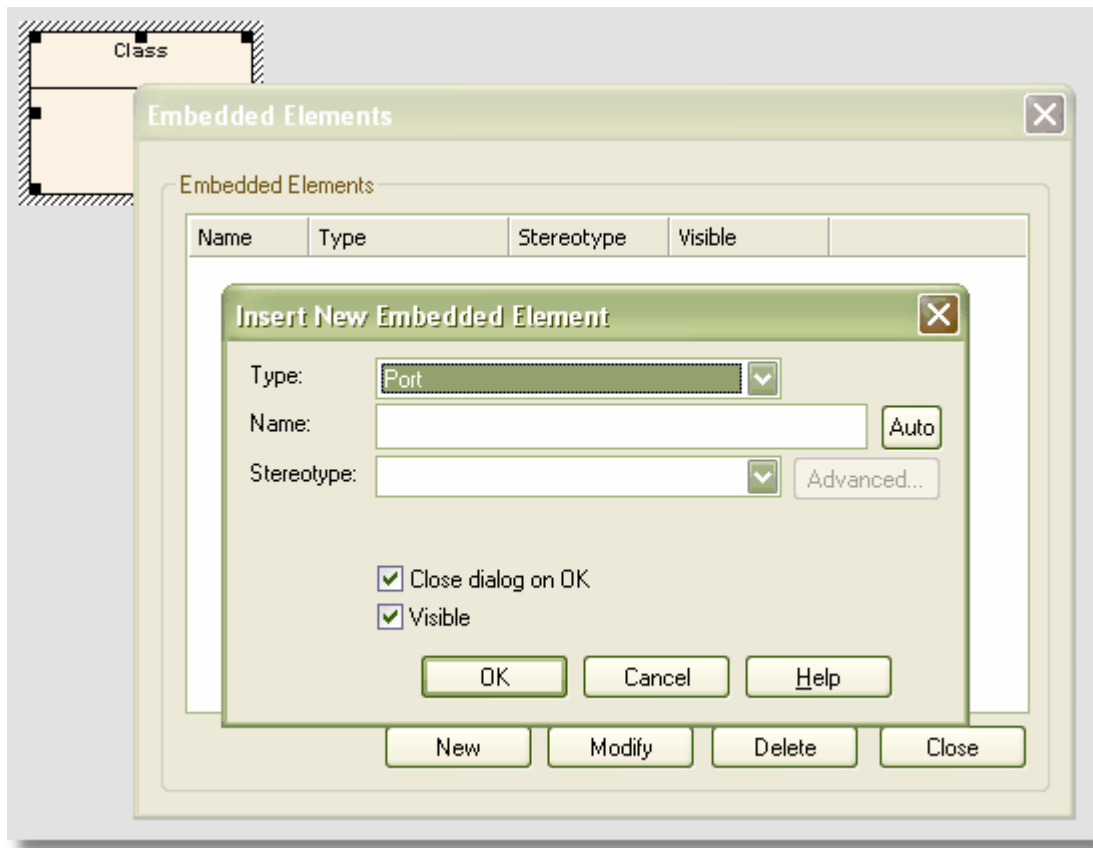
Element Features Sub-Menu

Menu Option	Description
Attributes	Opens the Attributes dialog.
Operations	Opens the Operations dialog.
Add Tagged Value	Allows you to add a tagged value .
Specify Feature Visibility	Opens the Set Feature Visibility dialog.

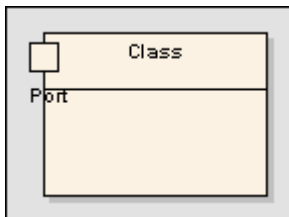
Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.2.1 Embedded Elements

The *Embedded Elements* dialog allows you to embed particular elements into other elements. For example, a Port can be embedded into a Class. The *Embedded Elements* option is available on the context menu of some elements.



In the *Embedded Elements* dialog, press *New* to create a new embedded element. Enter details such as type, name and stereotype, and press *OK*. The embedded element will now show on the primary element as shown below.



You can add as many embedded elements as necessary. Modify or delete embedded elements using the *Embedded Elements* dialog.

6.2.1.3 Type Menu Section

The Type menu section on the element context menu may contain the following options:

Menu Option	Description
Set Element Parent	Allows you to set the element parent .
Set as Composite Element	Set the element as a composite element .
Link Class to Association	Available if the element is a class. Allows you to link the class to a new Association .
Set Instance Classifier	Set the instance classifier for the element.
Use Rectangle Notation	Use rectangle notation for the element.
Partition Activity	Define an Activity Partition .
Set Run State	Add a new instance variable to the element using the Define Run State dialog.
Attach	Attach a note or attach a constraint to the element using the sub-menu provided.

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.4 Code Engineering Menu Section

The Code Engineering menu section on the element context menu may contain the following options:

Menu Option	Description
Generate Code (forward engineer)	Generate source code for the selected element.
Synchronize Model (reverse engineer)	Reverse engineer source code for the selected element.
View/Edit Source Code	Opens source code if it has been generated for the selected element.

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.5 Appearance Menu Section

The Appearance menu section on the element context menu may contain the following options:

Menu Option	Description
Selectable	Toggle whether the element is selectable. Selectable means shows the context menu - if it is unselectable, only the Selectable menu item shows on right click.
Appearance	See table below for sub-menu.
Z-Order	Set the Z-Order of the element.

Appearance Sub-Menu

Menu Option	Description
Adjust Appearance	Change the element's default appearance .
Select Alternate Image	Select an alternate image using the image manager .
Set Font	Change the font used for an element, to select the font for multiple elements within a diagram select the elements by holding down the CTRL key and then use the set font command .
Copy Appearance to Painter	Copy the element appearance to the painter
Copy Image of Selected Object(s) to Clipboard	Copy the element to the clipboard, so you can paste it into an external application.

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.6 Common Actions Menu Section

The Common Actions menu section on the element context menu may contain the following options:

Menu Option	Description
Insert Related Elements	Opens the Insert Related Elements dialog.
Locate in Project Browser	Locate the currently selected element in the Project Browser.
Project Information	See table below for sub-menu options.
Add to Favorites	Add the element to the Favorites folder .

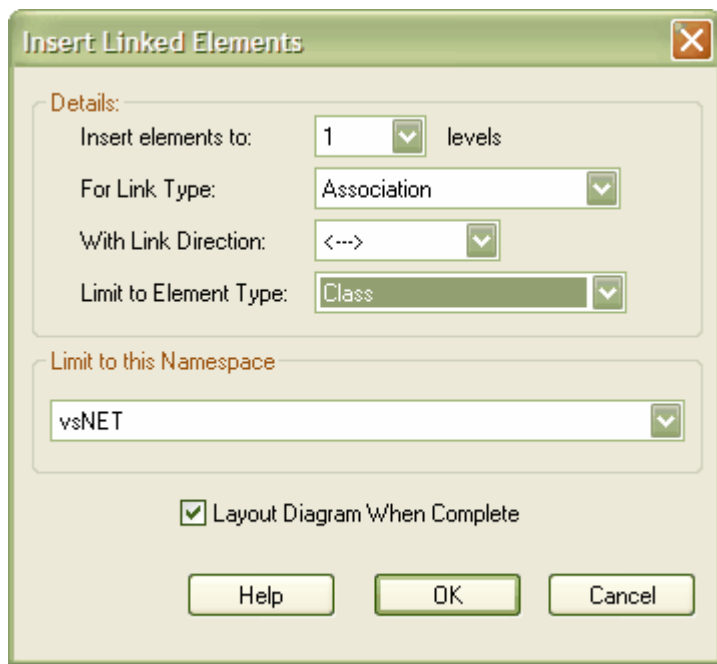
Project Information Sub-Menu

Menu Option	Description
Project, Metrics and Resource Information	Opens the Project Details dialog.
Usage	Opens the Element Usage dialog.
See Also	Allows you to set up Cross References .

Note: Not all menu options shown here will be present on all element context menus. Context menus vary slightly between element types. The Code Engineering section won't appear in a Use Case element, for example.

6.2.1.6.1 Insert Related Elements

The *Insert Related Elements* dialog can be accessed from most element context menus. This dialog allows you to insert linked elements from other diagrams into the current diagram.

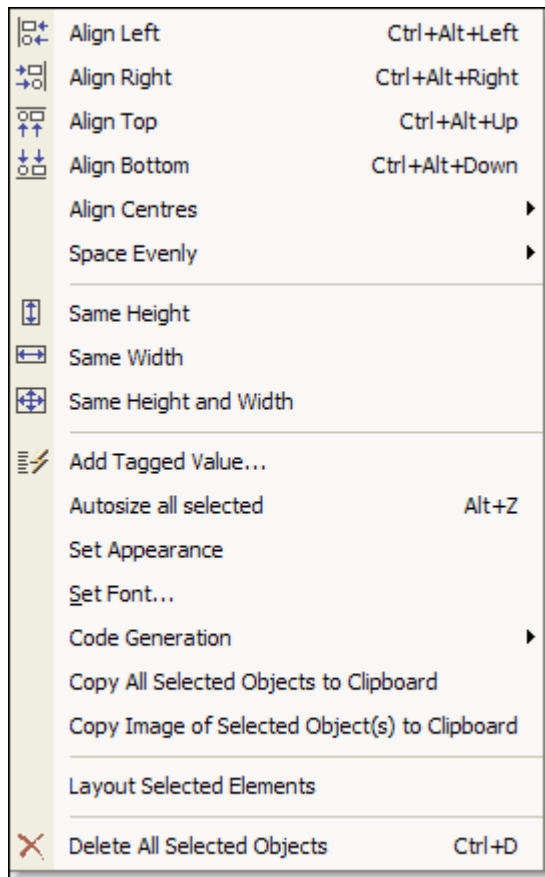


You can specify the following details:

Element	Description
Insert linked classes to <<x>> levels	Select the level you want to insert linked elements - levels 1-5 are available.
Link Type	Select the type of link you want the inserted elements to be connected by.
Link Direction	Select whether you want the links to be a single direction or bidirectional.
Element Type	Select the element type you want to insert.
Limit to this namespace	Limit the namespace you want the inserted elements to come from.
Layout Diagram When Complete	Select whether you want EA to layout the diagram after the elements have been inserted.

6.2.2 Element Context Menu - Multiple Selection

Right clicking on a selection of two or more elements in a diagram will cause the following context menu to be displayed:



This allows you to:

- Align elements (left, right, top, bottom or center)
- Space elements evenly (across or down)
- Set the appearance for multiple elements at once
- Specify the visibility of features for all selected elements
- Generate code for all selected elements at once
- Copy all selected elements to the clipboard
- Delete all selected elements

Tip: It is much faster to assign an appearance or characteristic to a group of elements than doing it one at a time.

6.2.3 Common Element Tasks

This topic covers various common UML tasks you can perform in Enterprise Architect.

6.2.3.1 Creating Elements

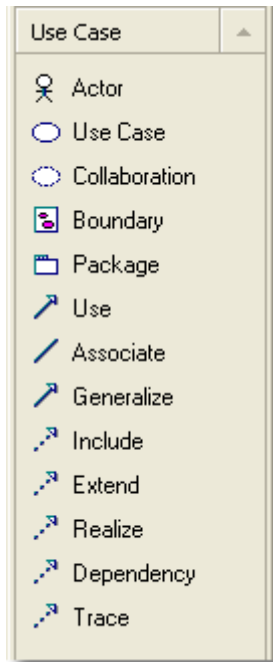
Use the UML Toolbox to select elements and connections to insert into the current diagram

The toolbar contains a number of different element groups, press the named button for the group you need to select from. Each diagram has a default toolbar group which will open automatically as the diagram is opened, however elements or connections from any group may be placed into any diagram.

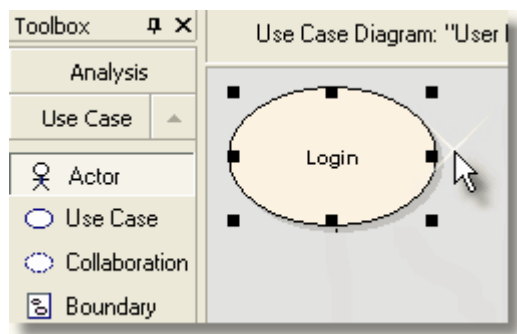
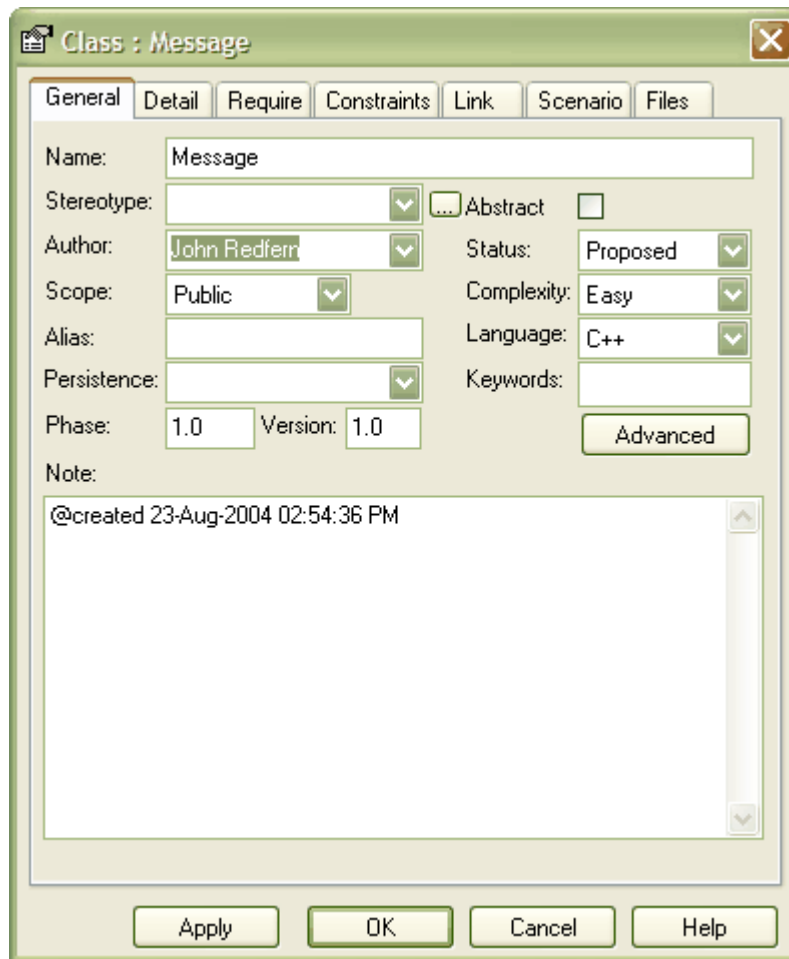
Create a New Element in a Diagram

To create a new element in a diagram, follow the steps below:

1. Select the required category in the UML Toolbox.
2. Left click an *element icon* to create a new element of that type in the current diagram.



3. Enter a *Name* when prompted.
4. Enter any other settings you require, such as *Notes* and *Version* or *Phase number*.
5. Press *OK*. The element will appear in the upper left corner of the current view.
6. Drag the element to the required location and resize if necessary.



Note: If there is no current diagram, no element will be inserted. You must first have created and opened a suitable diagram for your new model element.

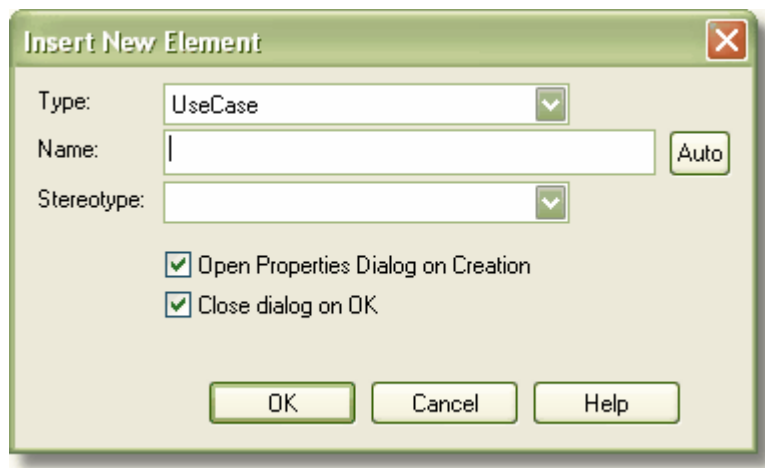
6.2.3.2 New Elements

New Elements may be quickly added to a package without the necessity of adding a diagram element at the same time. This is particularly useful if you wish to define a group of requirements, changes, issues, base classes or other element type that may not require diagrammatic representation in the model.

Add a New Element to a Package

To add a new element to a package, follow the steps below:

1. In the Project Browser, right click an appropriate package to open the context menu.
2. Select *New Element*.
3. Select the element *Type* from the drop down list.
4. Enter a *Name* for the new element and an optional *Stereotype*.
5. Check the *Open Property Dialog* option if you want the property dialog to open immediately after the element is created.
6. Clear the *Close Dialog on OK* option if you want to add multiple elements in one session.



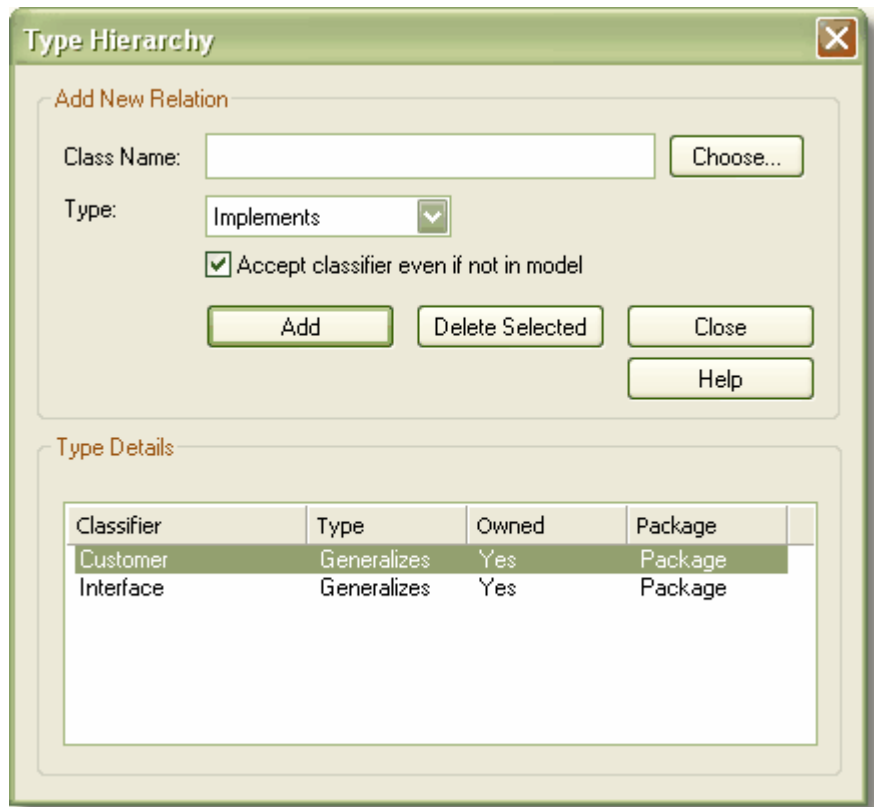
6.2.3.3 Set Element Parent

You can manually set an element's parent or an interface it realizes, using the *Type Hierarchy* dialog.

Set the Element Parent

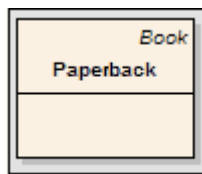
To set the element parent, follow the steps below:

1. Select a generalizable element in a diagram.
2. From the *Element* menu, select *Set Parents and Interfaces*. Alternatively, press *Ctrl+I*.
3. The *Type Hierarchy* dialog will open.



4. You can elect to enter a parent or interface name by either manually by typing it in, or by using the *Choose...* function to locate an element within the current model.
5. Set the *Type* of relationship (implements or inherits) from the drop down list.
6. Press *Add* to add the relationship.
7. Press *Delete Selected* to remove the current selected relationship.

Note: Parents that do not have their corresponding related element in the same diagram will display their parentage in the top right corner of the child element, as shown below:



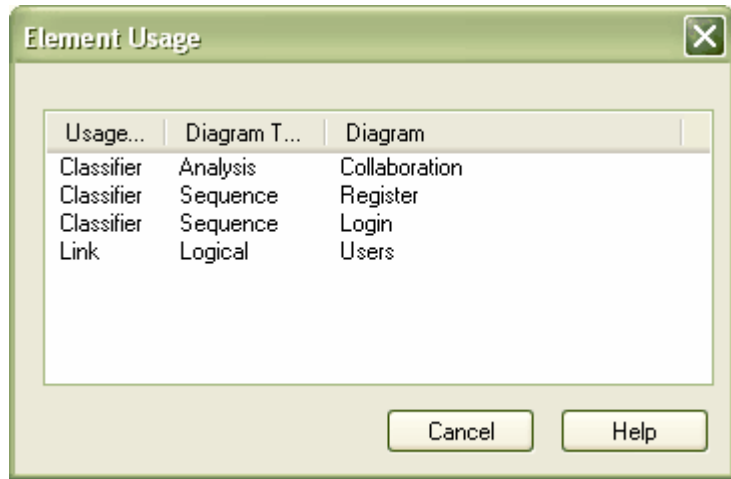
6.2.3.4 Show Element Usage

You may display the usage of an element using the *Show Usage* dialog. This lists all occurrences of the element throughout the model, and allows you to easily navigate to any occurrence.

Show Element Usage

To show element usage, follow the steps below:

1. Select an element in a diagram.
2. From the *Element* menu, select *Show Usage*. Alternatively, press *Ctrl+U*.
3. The *Element Usage* dialog will open, displaying all occurrences of the current element in the model.



4. Double click a line item to open the relevant diagram and display the selected element.

Note: You may also access this feature from the Project Browser - select an element in the tree and select *Show Usage* from the *Element* menu.

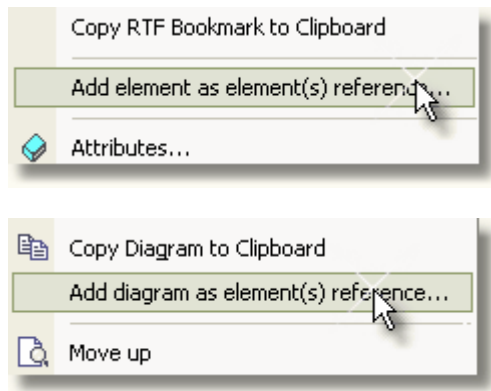
6.2.3.5 Set Up References (Cross References)

It is possible to set up a cross reference from one element in EA to another.

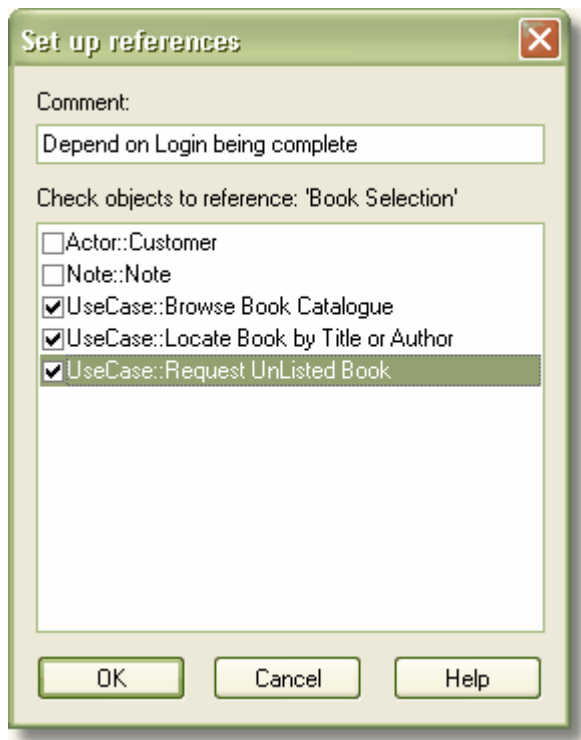
Set Up a Cross Reference

To set up a cross reference, follow the steps below:

1. In the Project Browser, locate the target element or diagram (this will be the subject of the cross reference).
2. Open a diagram that contains the element(s) that will have the currently selected element as a reference.
3. Right click on the element to open the element context menu.
4. Select *Add element as element(s) reference....* (In the case of a diagram select *Add diagram as element(s) reference....*)



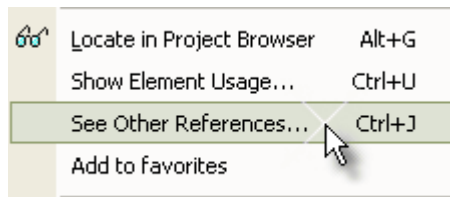
5. In the *Set Reference* dialog, elements you wish to have include the currently selected item in the explorer as a reference.
6. Enter an optional *Comment* to describe the purpose of the reference.



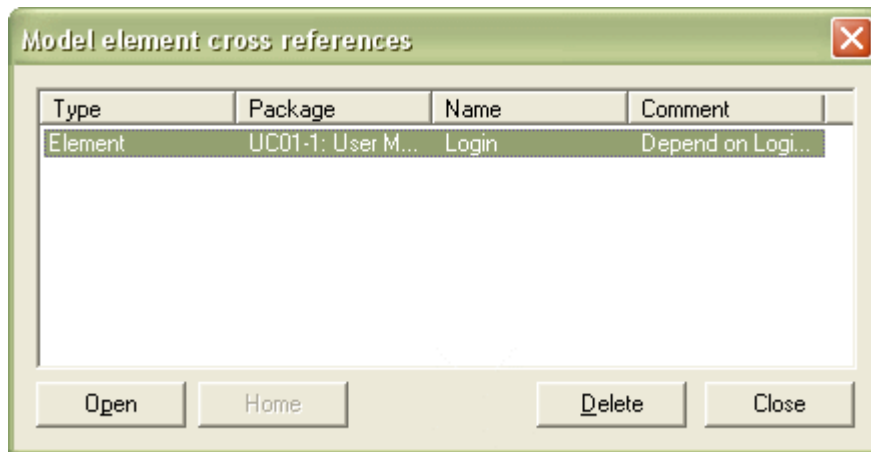
Use the Cross Reference

To use the cross reference, follow the steps below:

1. Select an element in a diagram.
2. From the *Element* menu, select *See Other References*. Alternatively, press *Ctrl+J*.



3. The *Cross Reference* dialog will appear and display a list of elements that have been set as cross references.
4. You can open a selected element by highlighting it and pressing *Open*.
5. If you have a diagram cross reference, you can *Open* that diagram.
6. If you have a string of diagram links, press *Home* to return to the original diagram.



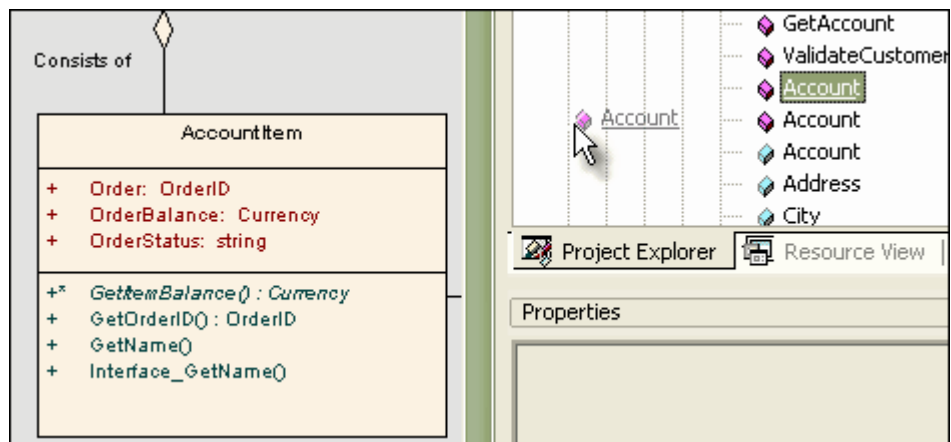
6.2.3.6 Copying Attributes and Operations Between Elements

Using drag and drop, you can copy an attribute and/or an operation from an element in the Project Browser on to another element in a diagram. or to another element within the Project Browser.

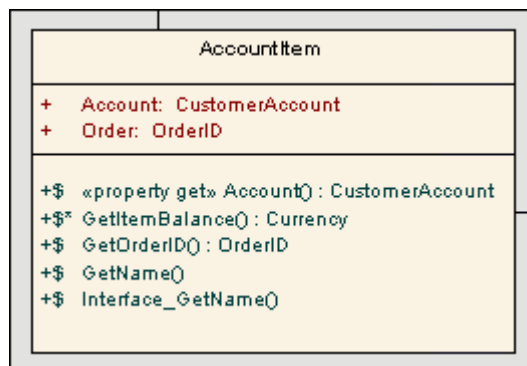
Copy an Element Feature

To copy an element feature, follow the steps below:

1. Open a diagram which contains the target element (in the example below, the AccountItem class is the target and Customer element is the donor).
2. Left click on the Attribute or Operation and drag to the target element.
3. Release the mouse button.



The image below shows AccountItem after the attribute 'Account' has been dropped from the browser on to it.



Move Multiple Element Features in the Project

To copy an element feature, follow the steps below:

1. Open a diagram which contains the target element (in the example below, the AccountItem class is the target and Customer element is the donor).
2. Left click on the Attribute or Operation and drag to the target element.
3. Release the mouse button.

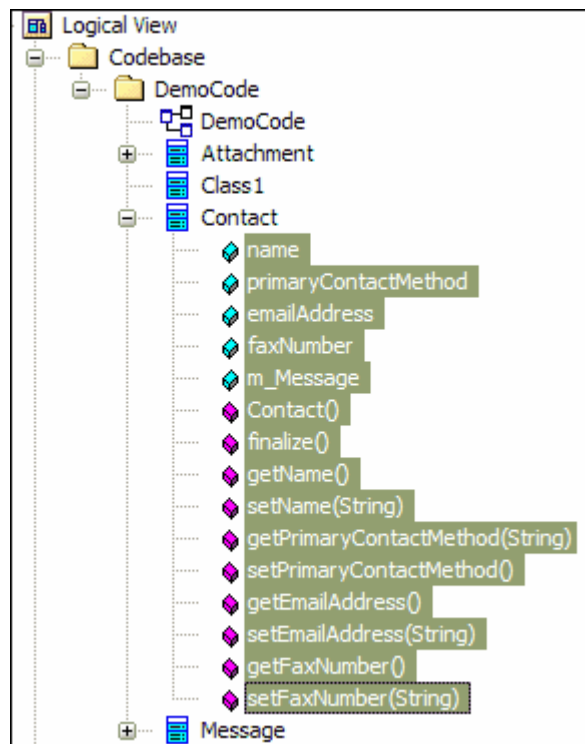
6.2.3.7 Moving Attributes and Operations between Elements

By using drag and drop, you can move attributes and/or operations from an element in the Project Browser on to another element within the Project Browser.

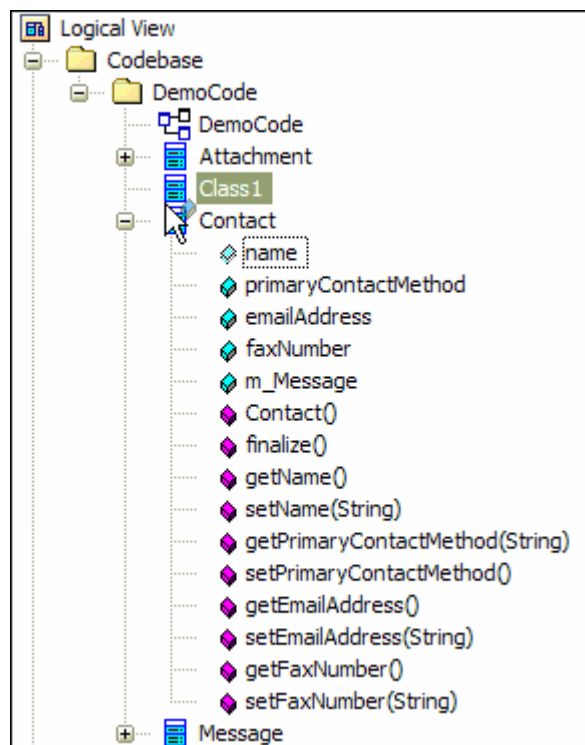
Move Multiple Element Features in the Project Browser

To move element features, follow the steps below:

1. Locate the attributes and/or operation in the Project Browser that you wish to move from the target element and select them by holding down the **Ctrl** (single item select) or the **Shift** key (multiple item select) .

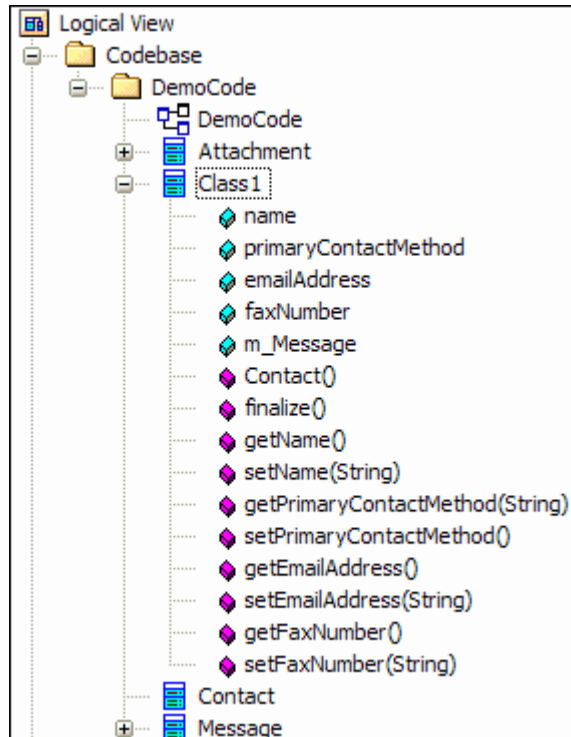


2. Holding down the *right mouse button* and the *Shift* or *Ctrl* key move the attributes and/or operations to the target element. This move will display only the last selected element feature during the move, however all of the selected features will be moved.



3. Release the mouse button. The image below shows the final stage of the attribute and operations

move between the Contact class and Class1.



6.2.3.8 Moving Elements between Packages

To move an element between packages, follow the steps below:

1. In the *Project Browser*, locate the element you wish to move.
2. Left click on the element and drag the mouse cursor to the package you wish to drop the element into.
3. Release the mouse button.

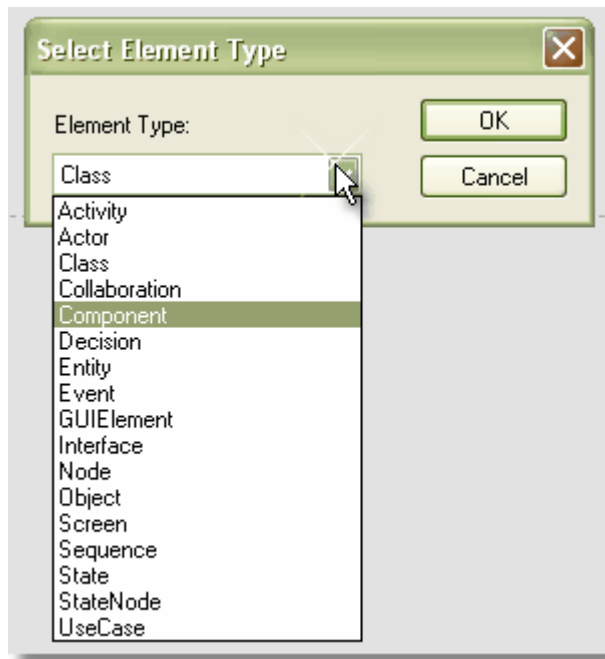
EA will move the element into the new package.

Tip: You can also drag the element under a host element in the new package - for example drag an element under a class

6.2.3.9 Changing Element Type

To change an element type, follow the steps below:

1. In the diagram view, element you wish to change.
2. From the *Element* menu, select *Change Type*.
3. Select the required *Element Type* from the drop down list.
4. Press *OK*.

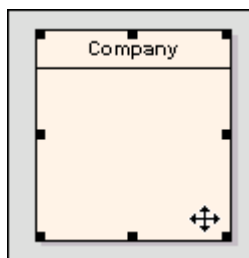


The target will be transformed into the required type.

6.2.3.10 Moving Elements

Any one of the following options will allow you to move an element within a diagram. Select an element or group of elements in the diagram view, then:

- Use the mouse to drag to the desired position (mouse cursor switches to the four-arrow icon as shown below), or



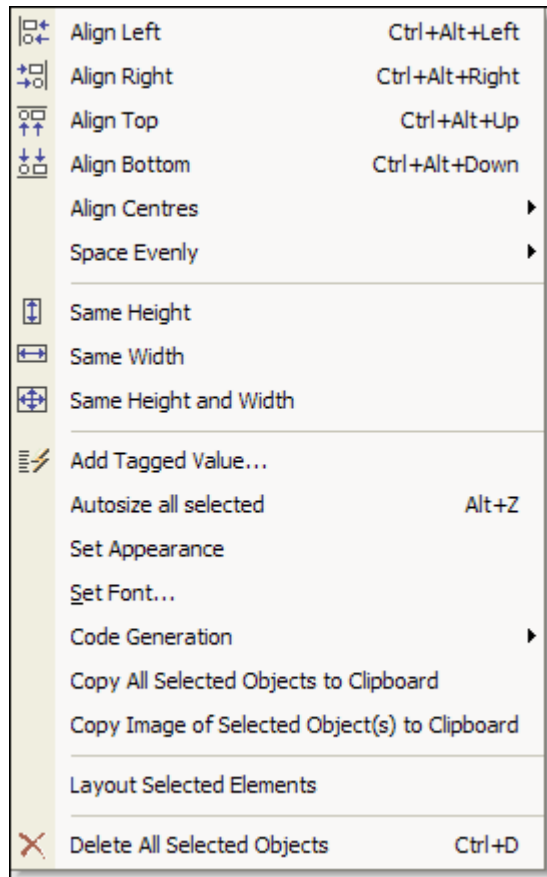
- Hold down the *Shift* key, and use the *Arrow* keys to move to the desired position, or
- Use the *Left*, *Right*, *Up* and *Down* options in the *Element | Move* submenu, or
- Multiple elements may be aligned to one another using the *Element | Alignment* submenu, the Alignment options in the right-click context menu, or the Alignment buttons on the *Diagram* toolbar



6.2.3.11 Aligning Elements

To align multiple elements, follow the steps below:

1. Select a group of elements by drawing a selection box around them all. (Or select one by one by holding down the *Ctrl* button and left clicking each element)
2. To open the context menu, right click on the element(s) in the group you wish to align others to.



3. Select the alignment function you require.

All selected elements will be aligned to the one beneath the cursor.

Tip: You can also use the diagram toolbar. The first four buttons are used to align elements, and are made available when more than one element is selected in a diagram.

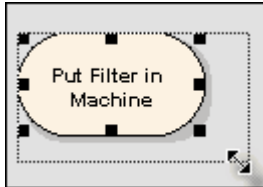


Tip: You can also select *Alignment* from the *Element* menu.

6.2.3.12 Resizing Elements

Any one of the following options will allow you to resize an element. Select an element or group of elements in the diagram view, then:

- Use the resize handles which appear at each corner and side to resize the element(s) by dragging with the mouse (mouse cursor switches to the double-ended arrow as shown below), or

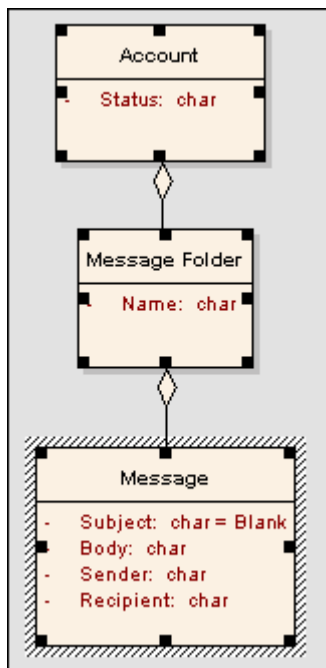


- Hold down the **Ctrl** key, and use the **Arrow** keys to resize as desired, or
- Use the **Wider**, **Narrower**, **Taller** and **Shorter** options in the **Element | Move** submenu, or
- Autosize selected element(s) using the option in the **Element | Appearance** submenu, or by pressing **Alt+Z**. (With multiple elements selected, **Autosize all selected** appears in the right-click context menu), or
- Set multiple elements to the same height, width or both, using these options in the **Element | Make Same** submenu, or the options in the right-click context menu

Resizing a Set of Objects to a Specific Size

Right clicking a selected set of object allows you to resize them to the same height, width or both. When you select multiple elements using **Control+Click**, then resize the height and/or width, the height and/or width of the selected (hatched) object on which the right-click is performed will be used to set the height and/or width of the other selected objects.

For example, in the diagram below, the Message class's height and width will be used to set the height and width of the Account and Message Folder classes. The aim is to make the Account and Message Folder elements the same height and width as the Message element.



To do this follow the steps below:

1. Set one element to the size you want (eg. Message as above).

2. Select all other elements (eg. Account and Message Folder as above).
3. Right-click the pre-sized element (eg. Message).
4. Select your resizing option (same height, width etc.).

See also: [Highlight Context Element](#)

6.2.3.13 Deleting Elements

To delete an element from a diagram, follow the steps below:

1. In the active diagram, select the element you wish to delete.
2. Press the *Delete* key, or right click to open the context menu and select *Delete <element name>*.

Note: *This does not delete the element from the model, only from the current diagram.*

To delete an element from the model, follow the steps below:

1. In the Project Browser, right click on the element you wish to delete.
2. From the context menu, select *Delete <element name>*.
3. Confirm *Yes* when prompted.

-OR-

4. Press the *Ctrl + Delete* key combination on a selected element in a diagram to completely remove an element from the model.

To delete an multiple elements from a diagram and model, follow the steps below:

1. Open the diagram with the elements that you wish to remove from the model.
2. Select all of the elements in a diagram by pressing the *CTRL + A* hotkey combination to target specific elements use the *CTRL + left mouse* click combination.
3. Press the *CTRL + Delete* to completely remove the elements from the model .

To delete an multiple elements from the Project Browser and model, follow the steps below:

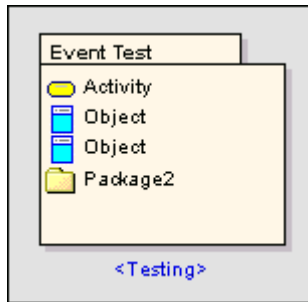
1. Select the items in the Project browser by holding down the *CTRL* or *SHIFT* key and holding down the right mouse button. The *CTRL* key will enable the user to select multiple items from the Project Browser one at a time whereas the *SHIFT* key select all of the items in the Project Browser tree between the first clicked item and the last clicked item.
2. Press the *CTRL + Delete* to completely remove the elements from the model .

Note: *If you delete an element here by using the *CTRL + Delete* hotkey combination, all its properties and connections are deleted as well.*

6.2.3.14 Customize Visible Elements

Some elements are hidden from view in packages and in RTF documents by default - this includes Events, Decisions, Sequence elements and Associations. You have the option of turning these elements back on.

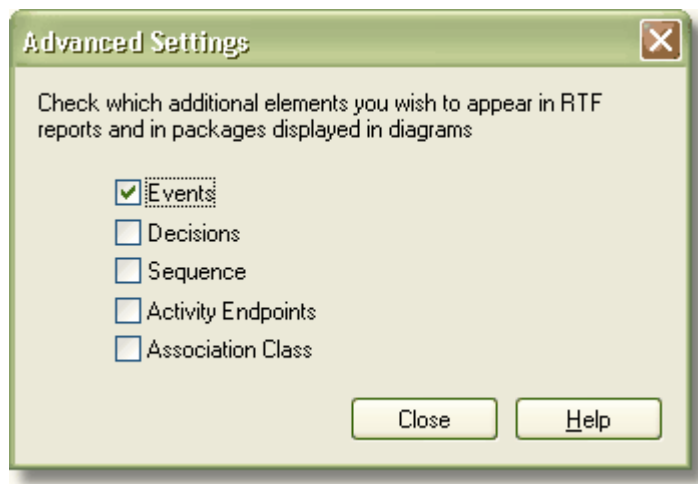
For example, a package containing some events and decisions does not show these in the package view - as in the example below.



Customize Visible Elements

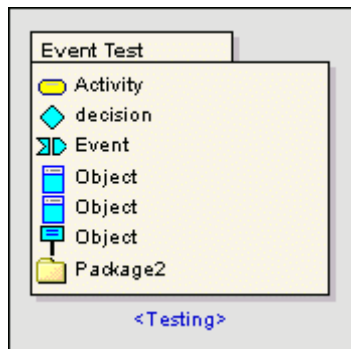
To show additional elements, follow these steps:

1. From the *Tools* menu, select *Options* to open the *Local Options* dialog.
2. Select the *Objects* tab.
3. Press *Advanced* to open the *Advanced Settings* dialog:



4. Check the elements you wish to show in packages and in RTF documents.
5. Press *Close*. Close the *Local Options* dialog.
6. Reload the current diagram if required.

The package from the example above will now show the Event, Sequence and Decision elements it contains:



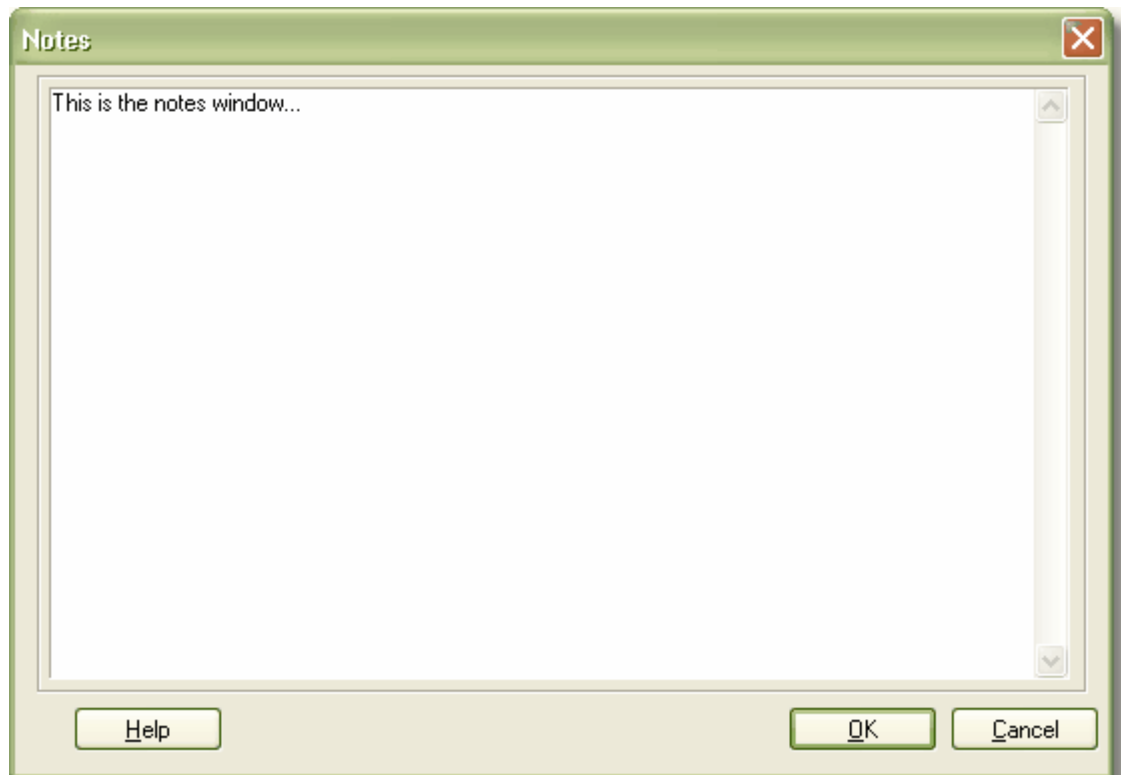
6.2.3.15 Creating Notes and Text

You can create notes and text in EA - the two are slightly different.

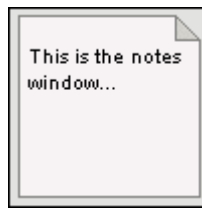
Creating a Note

One way to create a note is:

1. Right click on the background of a diagram to open the context menu.
2. From the *Insert Element at Cursor* submenu, select *Note*. The *Notes* window appears.



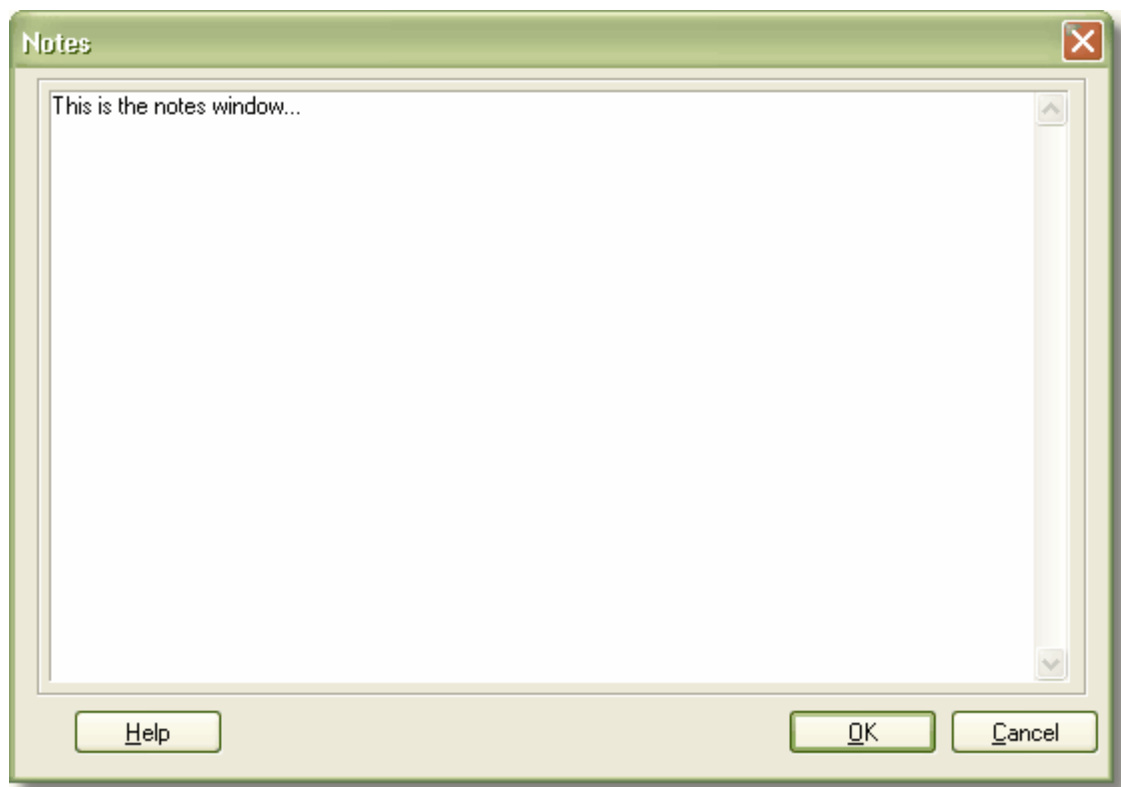
3. Type your note then press *OK* to save.
4. Your note will appear on the diagram in the following format:



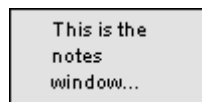
Creating Text

To create text, follow the steps below:

1. Right click on the background of a diagram to open the context menu.
2. From the *Insert Element at Cursor* submenu, select *Text*. The *Notes* window appears.



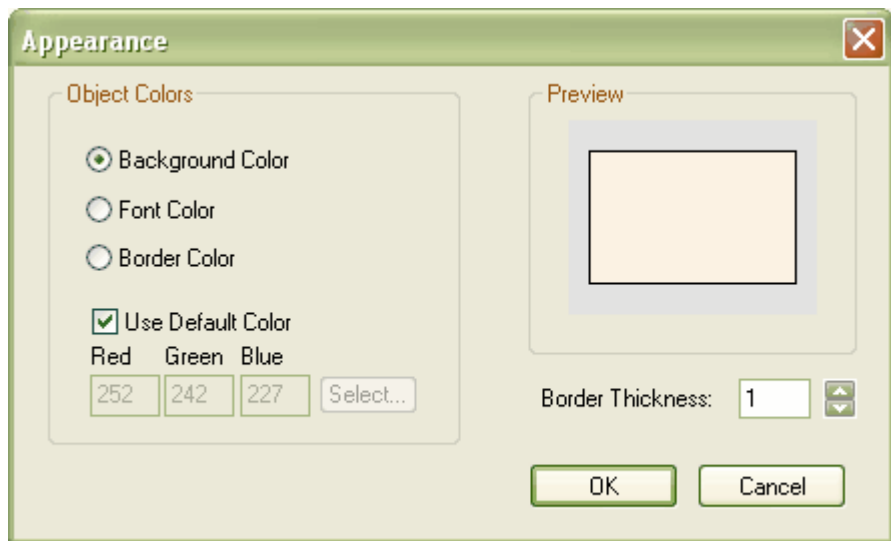
3. Type your note then press *OK* to save.
4. Your text will appear on the diagram in the following format:



6.2.3.16 Configure an Element's Default Appearance

The Appearance dialog may be used to create a global change to the appearance of a specific element in all of the diagrams where it appears within a model, to change a single of an element for a single diagram use the [Format toolbar](#). Configure the default appearance of an element using the [Appearance](#) dialog. Access this dialog by selecting an element, then selecting [Configure Default Appearance...](#) from the [Element | Appearance](#) submenu.

Note: Although changes to the appearance of a specific element made through the [Appearance](#) dialog effect all of the elements of its specific type throughout the model, if the element on a specific diagram is changed via the [Format](#) toolbar, the changes made by the [Format](#) toolbar will override any of the changes defined by the [Appearance](#) dialog.



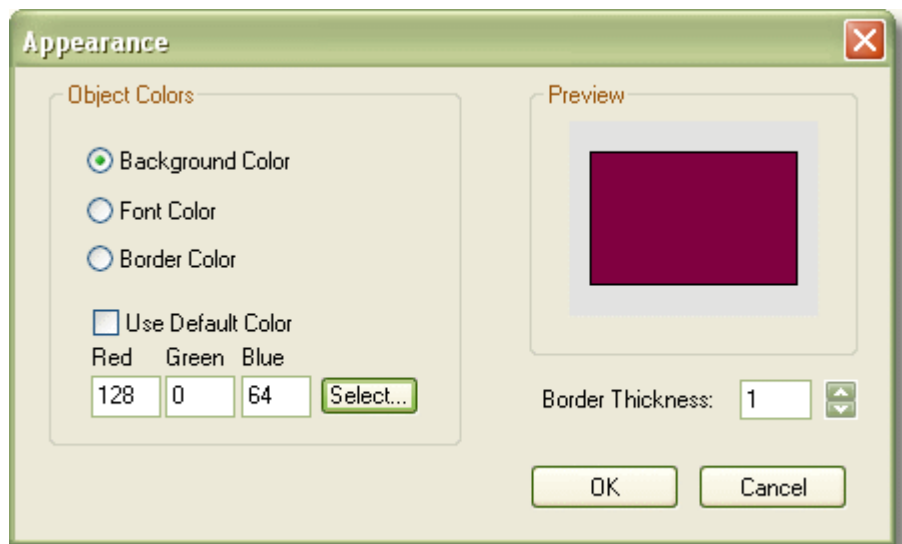
Changing a the Background, Font or Border Color

You can set the background color, the font color or the border color by following the steps below:

1. Select the [Background Color](#), [Font Color](#) or [Border Color](#) radio button as appropriate.
2. Uncheck the [Use Default Color](#) option to enable the [Select](#) button.
3. Press [Select](#) to open the [Color](#) dialog.



4. Select the color you want (use *Define Custom Colors* for a more exact color) and press *OK*.
5. You will see in the *Preview* that the element's color has changed to the selected color. This won't be applied to the actual element until you press *OK*.



Note: The color can be easily be returned to the default color by checking the *Use Default Color* checkbox.

Note: You can change it to a different color by pressing *Select* again.

Changing the Border Thickness

To change the border thickness, change the number in the *Border Thickness* field on the right. Use the scroll arrows to increase or decrease the number, or type a new number directly into the text box.



You can see the effect in the preview pane. This won't be applied to the actual element until you press **OK**.

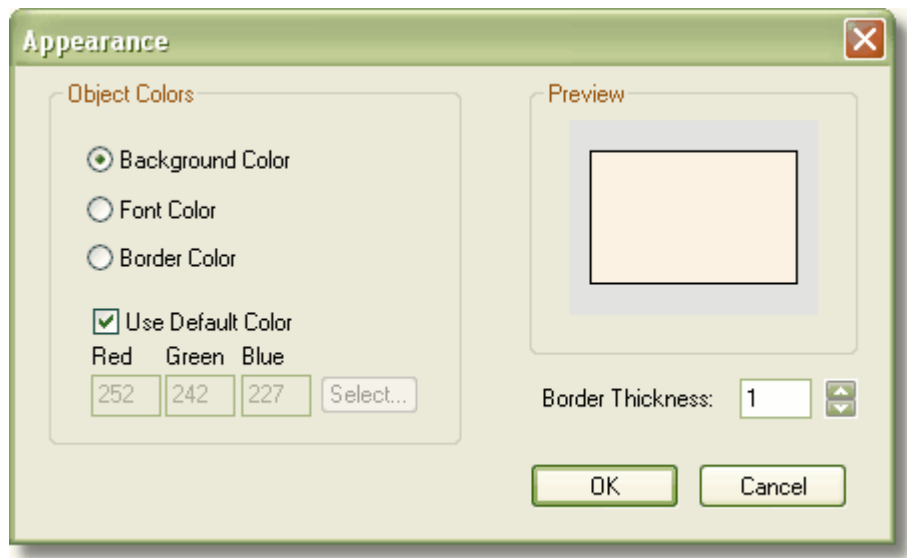
6.2.3.17 Get/Set Project Custom Colors

There may be times when more than one person is working on a project and they may wish to use specific colors for elements within the project. The **Get** and **Set Project Custom Colors** items on the **Project** menu allow you to **set** specific colors then **get** the colors in a different session, without having to remember RGB values.

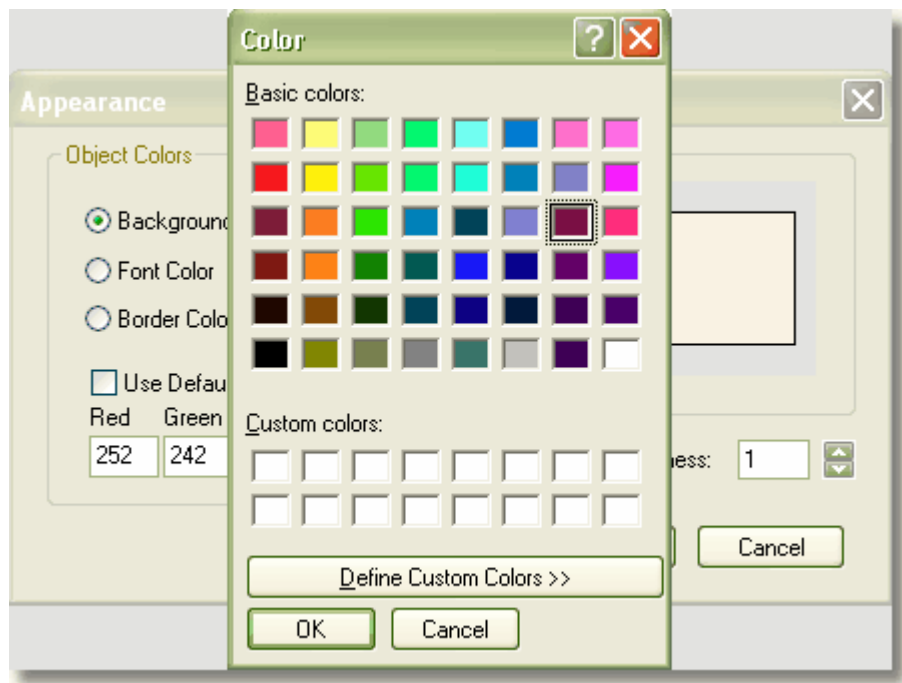
Set a Project Custom Color

Follow the steps below to set your project's custom colors:

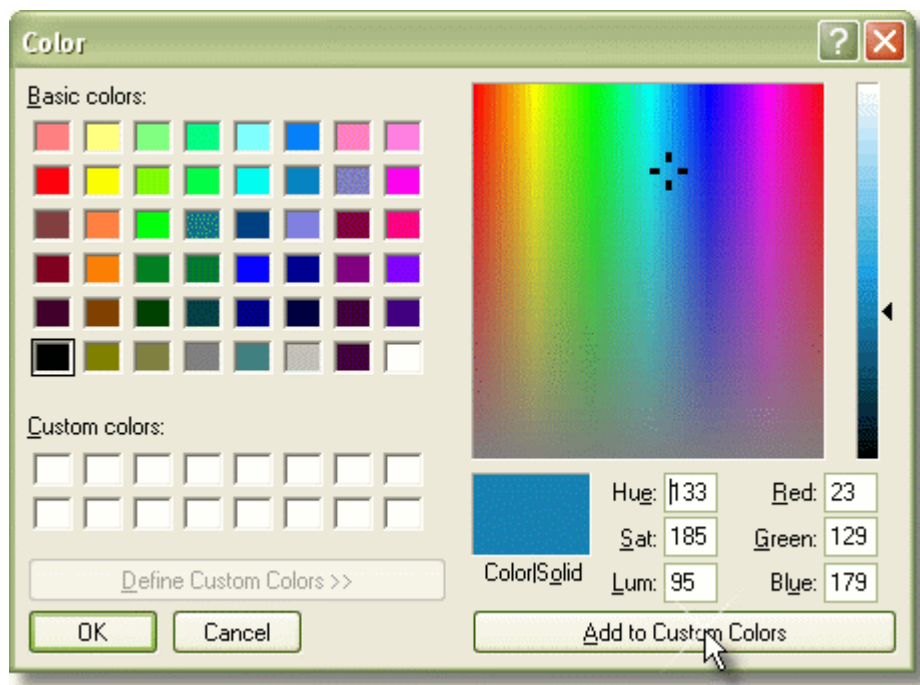
1. Select an element you would like to color.
2. From the **Element | Appearance** submenu, select **Configure Default Appearance...** to open the **Appearance** dialog.



3. Uncheck the **Use Default Color** option to enable the **Select** button.
4. Press **Select** to open the **Color** dialog.



5. Press *Define Custom Colors >>*.
6. Create the color in the color mixer on the right - see below.



7. Press *Add to Custom Colors*. This will add the color to the *Custom colors* listing on the left hand side of the dialog - see below:

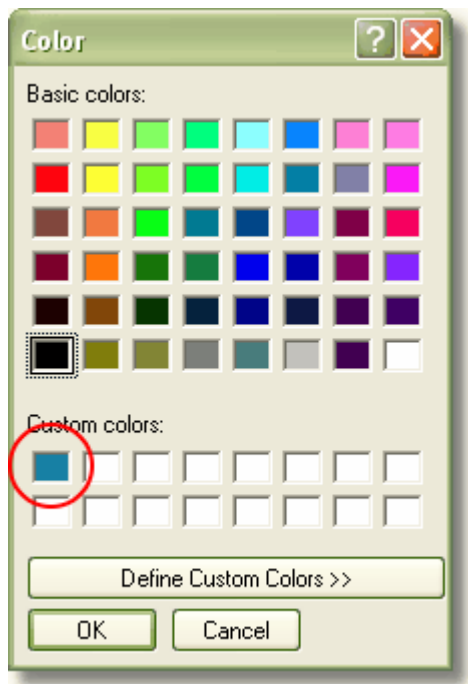


8. Press **OK** to close the **Color** dialog, then **OK** to close the **Appearance** dialog.
9. From the **Project** menu, select **Set Project Custom Colors** to save the custom color you have created.

Get a Project Custom Color

Follow the steps below to get your project's custom colors:

1. From the **Project** menu, select **Get Project Custom Colors** . This will apply any saved custom colors to this project.
2. From the **Element** | **Appearance** submenu, select **Configure Default Appearance...** to open the **Appearance** dialog.
3. Uncheck the **Use Default Color** option to enable the **Select** button.
4. Press **Select** to view the applied custom color(s) - they will appear as circled below:

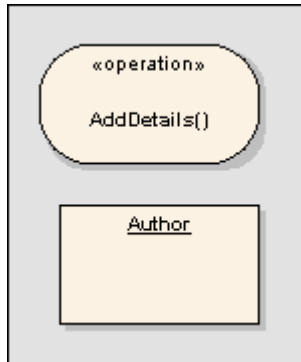


6.2.4 Attributes and Operations

Attributes are features of classes and some other elements that correspond to the data and state an element maintains. Operations are features that indicate the behavior and functionality an element has.

6.2.4.1 Adding Use Case Attributes and Operations to Activity Diagrams

You can add a Use Case's attributes and operations to an Activity diagram as Activity elements. Simply drag the attribute or operation from the Project Browser onto an Activity Diagram, and this will automatically create the element.



6.2.4.2 Attributes

Attributes are features of a class or other element that represent the properties or internal data elements of that element. For a Customer class, CustomerName and CustomerAddress may be attributes. Attributes have several important characteristics, such as type, scope (visibility), static, derived and notes.

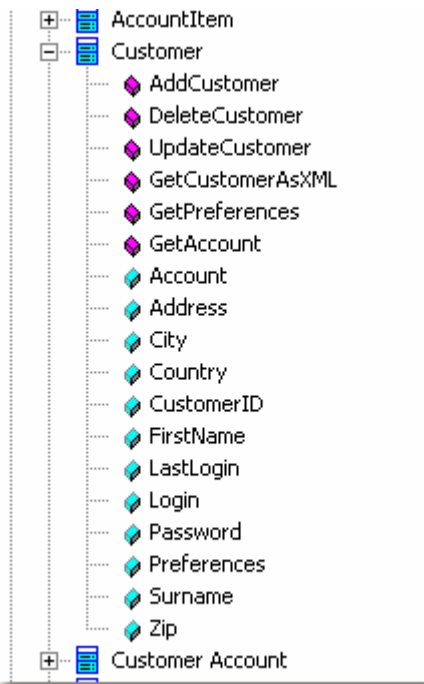
Creating and Modifying Element Attributes

1. In the diagram view, right click on the element to be edited.
2. From the context menu *Element Features* submenu, select *Attributes* to open the *Attributes* dialog. Alternatively, press *F9*.

Note: This option will only be available on the context menu if the element supports Attributes

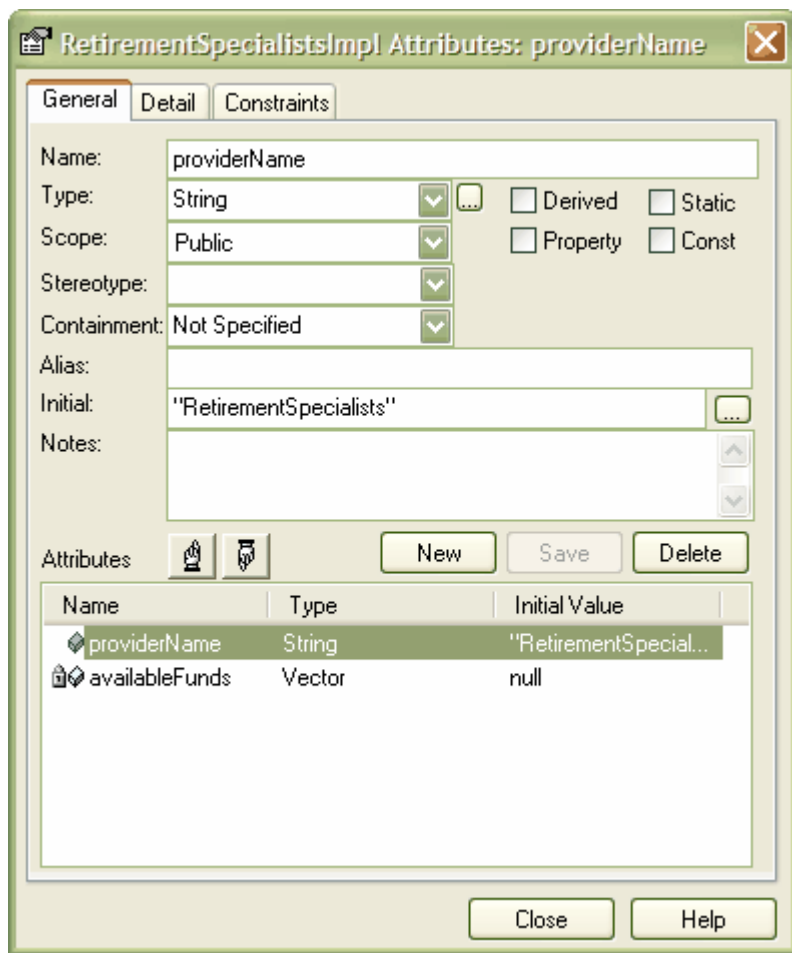


Note: Attributes are displayed in the Project Browser beneath the class:



6.2.4.2.1 Attributes Main Page

The *General* tab of the *Attributes* dialog is shown below:



Control	Description
Name	Attribute name
Type	Data type of attribute - select from the drop down list
Build button	Opens the <i>Select Attribute Type</i> dialog
Scope	Public/Protected/Private/Package
Stereotype	Optional Stereotype of the attribute
Containment	Containment type(by reference/value)
Derived	Indicates attribute is a calculated value
Static	Attribute is a static member
Property	Select automatic property creation
Const	Attribute is a constant
Alias	An optional alias for the attribute
Initial	An optional initial value
Notes	Free text notes
Attribute List	List of defined attributes. Select an attribute to make it current
Up/Down buttons	Use to change the order of attributes in the list
New	Create new attribute
Save	Save new attribute, or save modified details for existing attribute
Delete	Delete currently selected attribute

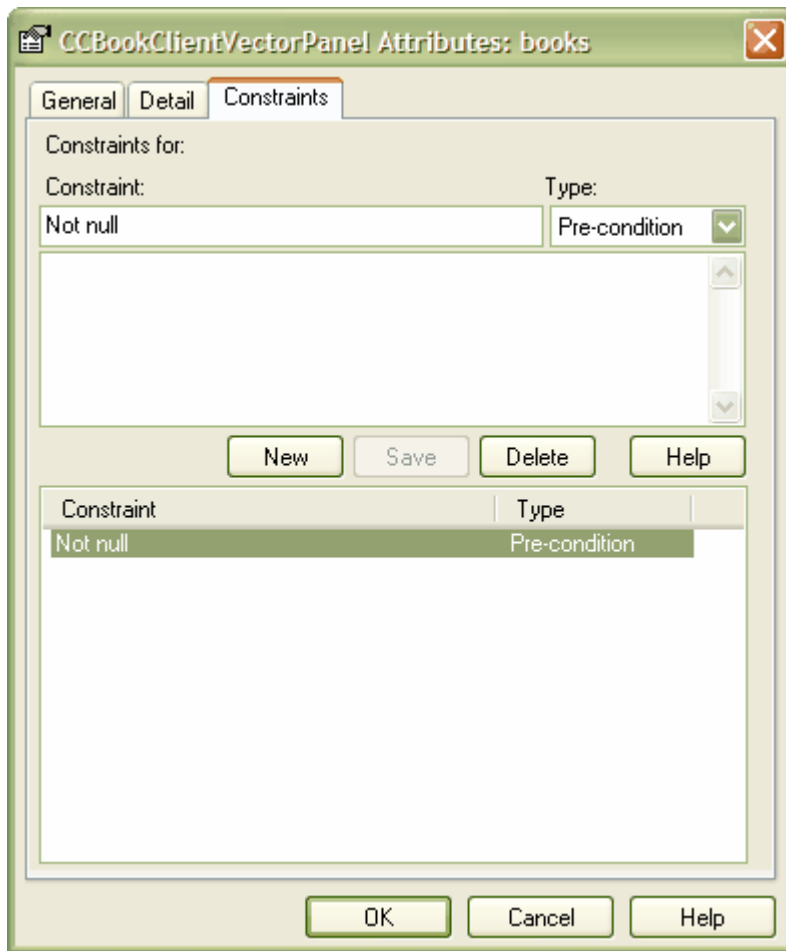
6.2.4.2.2 Attributes Detail

The *Detail* tab of the *Attributes* dialog has some additional details relating to collections.

Control	Description
Lower Bound	A lower limit
Upper Bound	An upper limit to the number of elements in the collection
Ordered Multiplicity	Set if the collection is ordered
Attribute is a Collection	Check if the attribute is a collection
Allow Duplicates	Set if duplicates are allowed
Container Type	The container type
Save	Save changes

6.2.4.2.3 Attribute Constraints

Attributes may also have *Constraints* associated with them. Typically this will indicate such things as maximum value, minimum value, length of field etc.



Control	Description
Constraint	Constraint name
Type	Constraint type
Notes	Constraint details
Constraint list	A list of constraints already defined
New	Create new attribute constraint
Save	Save new constraint details
Delete	Delete currently selected constraint
Help	Opens this help document

6.2.4.2.4 Attribute Tagged Values

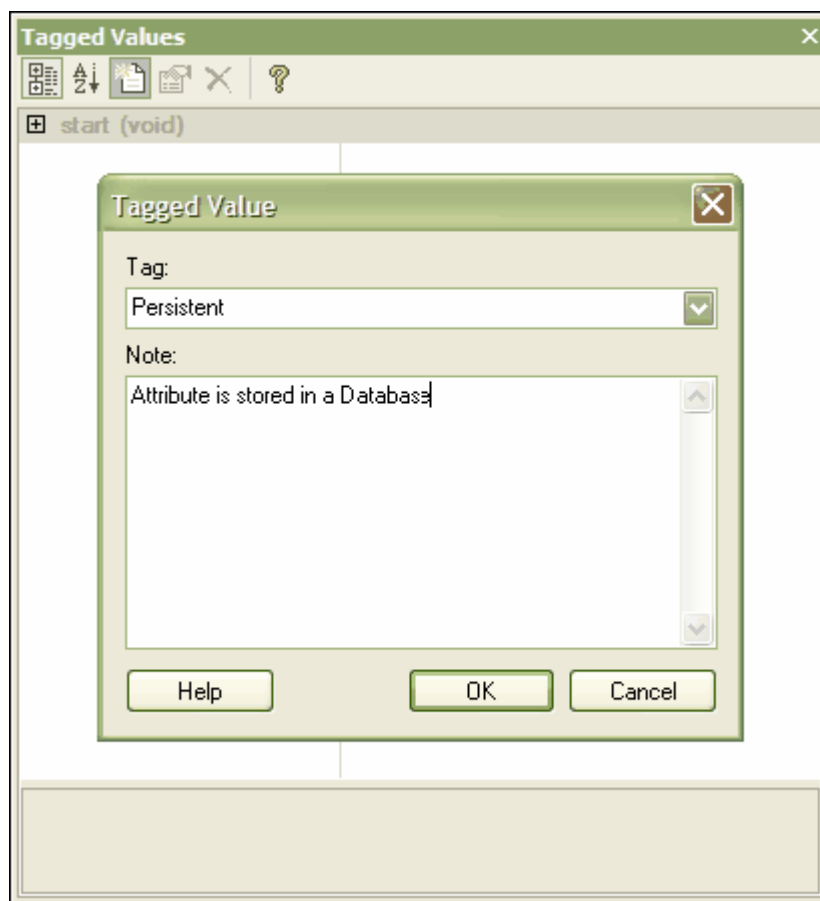
An attribute may have *Tagged Values* defined for it. Tagged values are a convenient means of extending the properties a model element supports. This in turn can be used by code generators and other utilities to transform UML models into other forms.

Tip: Tagged values are supported for Attributes, Operations, Objects and Connectors.

Add a Tagged Value

To add a tagged value for an attribute, use the following steps :

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the attribute by double clicking on the attribute in a diagram or on the attribute in the Project View.
3. The *Tagged Values* window will now have the attribute selected, press either the *New Tags* button or the *Ctrl + N* hotkey combination.
4. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
5. Then press *OK* the button to confirm the operation.



Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

6.2.4.2.5 Creating Properties

EA has some capabilities for automatically creating properties in various languages. Property creation is controlled from the *General* tab of the *Attribute* dialog. Select the *Property* option to activate this feature.

Name: TakeNeed

Type: boolean Derived Static

Scope: Private Property Const

Stereotype:

Containment: Not Specified

Initial:

Notes: Private declarations

Attributes

This opens the *Create Property Implementation* dialog (shown below). By default the class language is picked up as the default, however you may change this and generate for any language. Each language has slightly different syntax and generates slightly different results. For example, Java and C++ generate get and set functions, C# and VB.Net create property functions and Delphi creates get and set functions as well as a specialized Delphi property tagged value.

Name: City

Type: String Derived Property

Scope: Public

Stereotype:

Create Property Implementation

Language

- C++
- Java
- Visual Basic
- C#
- Delphi
- VB Net

Property Details

Name: City

Getter: GetCity()

Setter: SetCity(Value: String)

Stereotype: Published

Get Scope: Public Set Scope: Public

Enter your required details and press **OK**. EA will generate the required operations and/or properties to comply with the selected language. Note that get and set functions will be stereotypes with <<property get>> <<property set>> etc. making it easy to recognize property functions. You may also hide these specialized functions by checking the *Hide Properties* check box in the *Diagram Properties* dialog for a specific diagram. This makes it easier to view a class, uncluttered by many get and set methods.

Appearance Options

<input checked="" type="checkbox"/> Use Stereotype Icons	<input checked="" type="checkbox"/> Highlight Foreign Objects	<input type="checkbox"/> Hide Attributes
<input type="checkbox"/> Scale Printing to 1 Page	<input checked="" type="checkbox"/> Show Package Contents	<input type="checkbox"/> Hide Operations
<input checked="" type="checkbox"/> Show Page Border	<input type="checkbox"/> Show Details on Diagram	<input type="checkbox"/> Show Tags
<input type="checkbox"/> Use Alias if Available	<input type="checkbox"/> Show Sequence Notes	<input type="checkbox"/> Show Requirements
<input type="checkbox"/> Hide Property Methods	<input type="checkbox"/> Hide Additional Parents	<input type="checkbox"/> Show Constraints
<input type="checkbox"/> Hide Collaboration Numbers	<input type="checkbox"/> Hide Relationships	<input type="checkbox"/> Show Testing
<input type="checkbox"/> Hide Element Stereotype	<input type="checkbox"/> Hide Stereotype on Features	<input checked="" type="checkbox"/> Show Maintenance
<input type="checkbox"/> Hide Qualifiers	<input type="checkbox"/> Show Table Owner	

Visible Class Members

<input checked="" type="checkbox"/> Public
<input checked="" type="checkbox"/> Protected
<input checked="" type="checkbox"/> Private
<input checked="" type="checkbox"/> Package

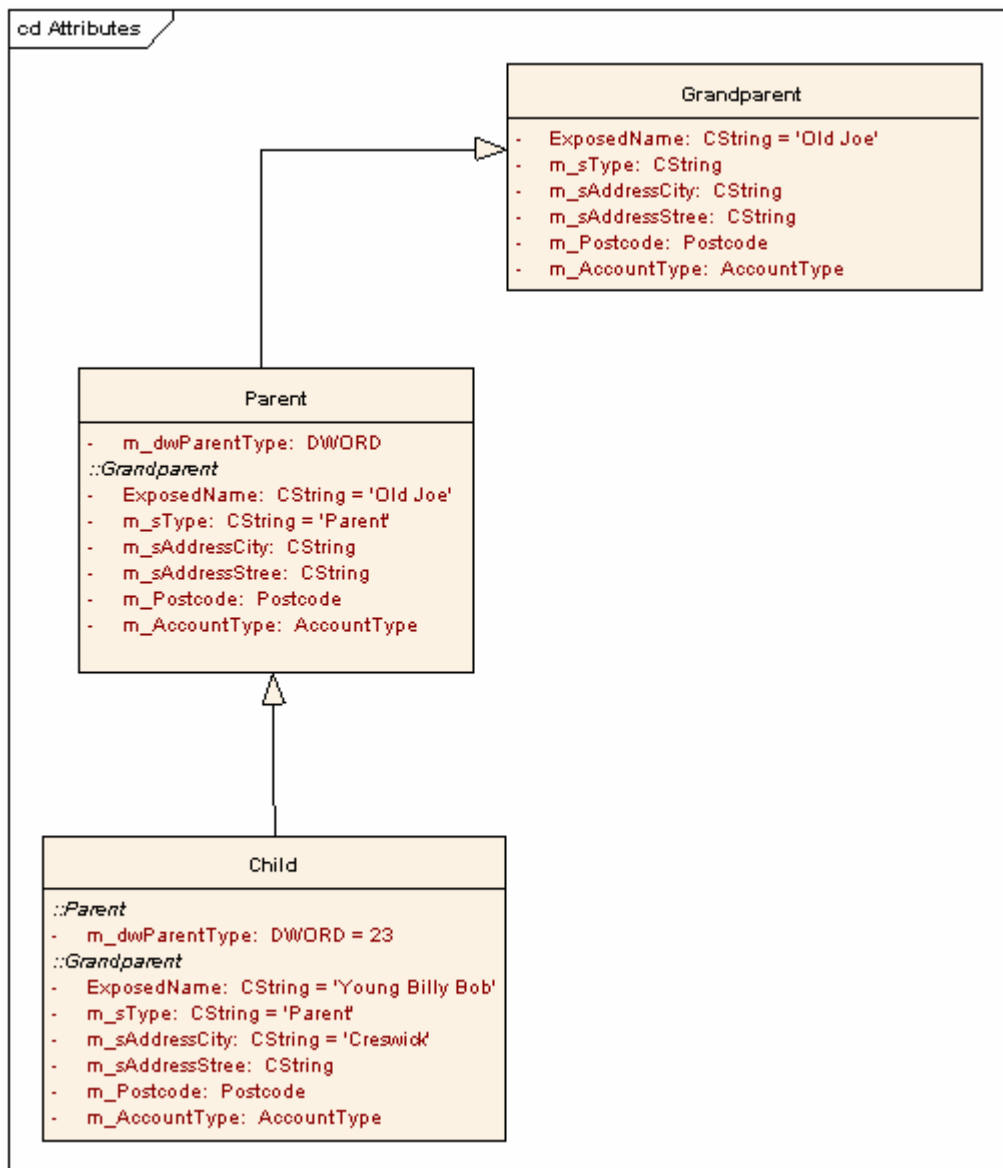
Show Parameter Detail:
Type Only

Note that for Delphi you need to enable the 'tagged values' compartment to see the generated properties. See [Compartment](#)s for how to do this.

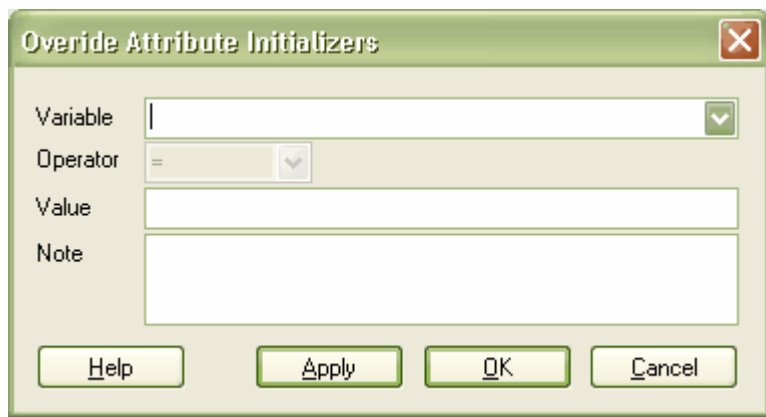
6.2.4.2.6 Displaying Inherited Attributes

When displaying a class with attributes in a diagram, it is possible to also show the inherited Attributes from all parents in the elements type hierarchy (ancestors).

To show inherited Attributes, use the [Specify Feature Visibility](#) dialog.



Note that it is also possible to override an inherited attributes "initial value". This is done using the context menu item "Set Attribute Initializers" from the element context menu. This only applies to elements which have Attributes. In the Attribute Initializer dialog, select the variable name and enter a new initial value. An optional note may also be entered. When you display inherited Attributes, EA will merge the list of Attributes from all ancestors and also merge the Attribute initializers, so that the final child class displays the correct Attribute set and initial values.



6.2.4.3 Operations

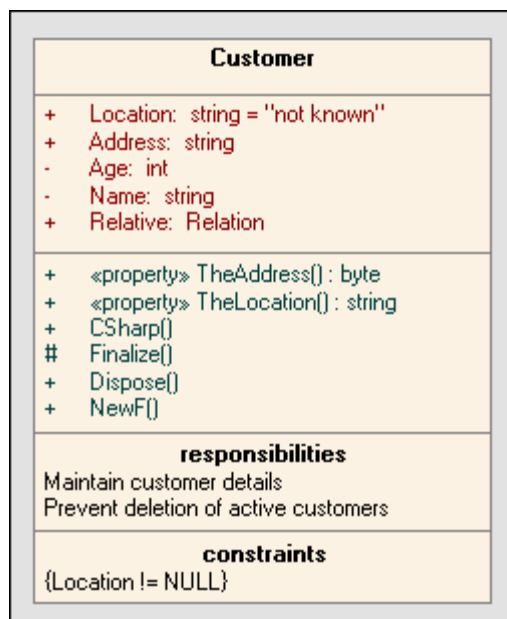
Operations are features of a class or other element that represent the behavior or services an element supports. For a Customer class, UpdateCustomerName and GetCustomerAddress may be operations. Operations have several important characteristics, such as type, scope (visibility), static, abstract and notes.

How to Access Operations

If an element supports operations (typically classes and interfaces), the *Element Features* submenu of the right click context menu will contain the *Operations* item. Select this to open the *Operations* dialog. Alternatively, press **F10**. For more information regarding this dialog, see below.

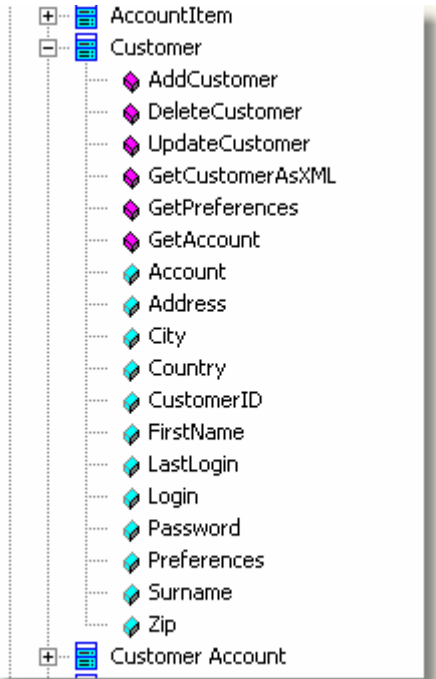
How Operations Appear in Diagrams

Elements with operations (typically classes) will display their features in diagrams in the manner shown below. As the diagram illustrates, some characteristics will display in shorthand form, for example 'static' will display as '\$', abstract as '*'



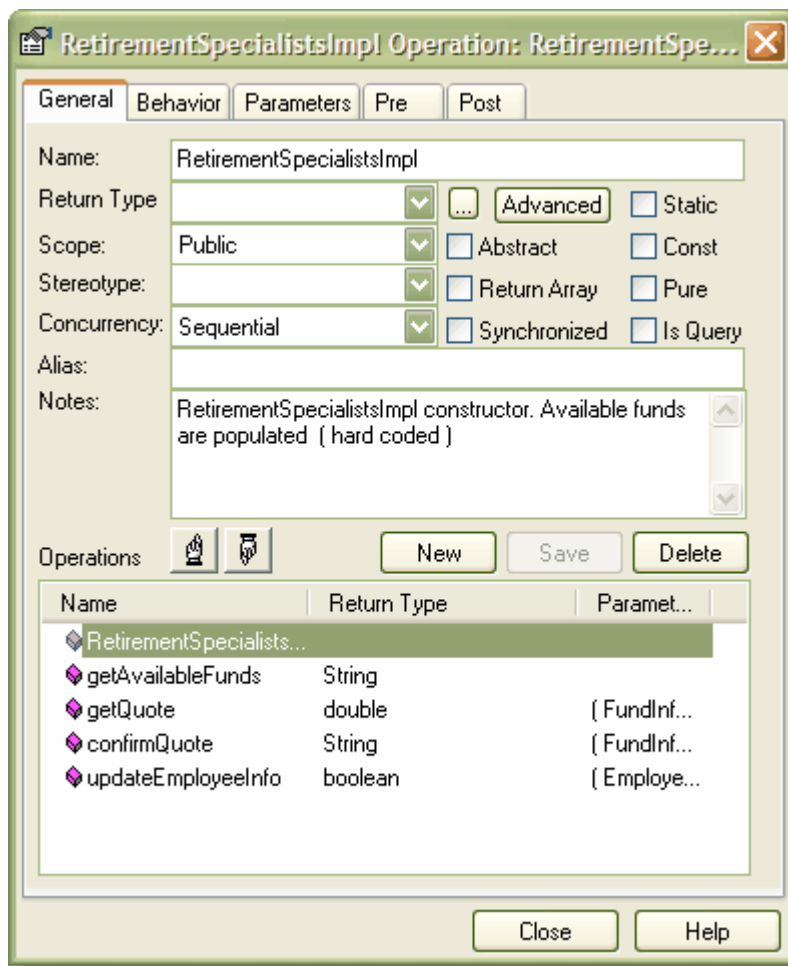
Operations in the Project Browser

Classes with operations will have their features collected beneath them in the Project Browser. Right click on an operation and select *Operation Properties...* to open the *Operations* dialog and edit details for the feature.



6.2.4.3.1 Operations Main Page

The *General* tab of the *Operations* dialog allows you to define new operations and set the most common properties, including name, access type, return etc.



Tip: To go directly to the parameters page, double click an operation in the operations list.

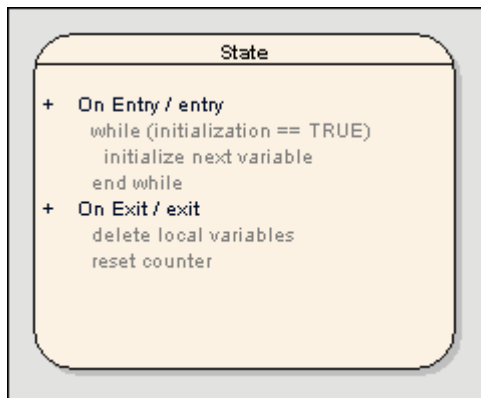
Control	Description
Name	Operation name
Type	Data type returned by operation
Build button	Opens the <i>Set Element Classifier</i> dialog
Scope	Public/Protected/Private/Package
Stereotype	An optional stereotype for this operation
Concurrency	Concurrency of operation
Virtual/Abstract	If the operation's language is set to C++, this option maps to the C++ Virtual keyword. Otherwise this option is Abstract, pertaining to an abstract function.
Return Array	The return value is an array
Synchronized	A code engineering flag which relates to multithreading in Java
Static	Operation is a static member
Const	Operation is a constant
Pure	Relates to C++ pure virtual syntax - eg. <code>virtual void myFunction() = 0;</code>
IsQuery	Operation is a database query
Alias	An optional alias for the operation
Notes	Free text notes
Operation List	List of defined operations
Up/Down Buttons	Use to change the order of operations in the list
New	Create new operation
Save	Save new operation, or save modified details for existing operation
Delete	Delete currently selected operation

6.2.4.3.2 Operations Detail

The *Behavior* tab of the *Operations* dialog allows you to enter free text to describe the functionality that an operation will have. Use pseudo code, structured English or just a brief description.

You can also use this field to formally describe a Method or State action and have the text appear under the method/action name in a diagram.

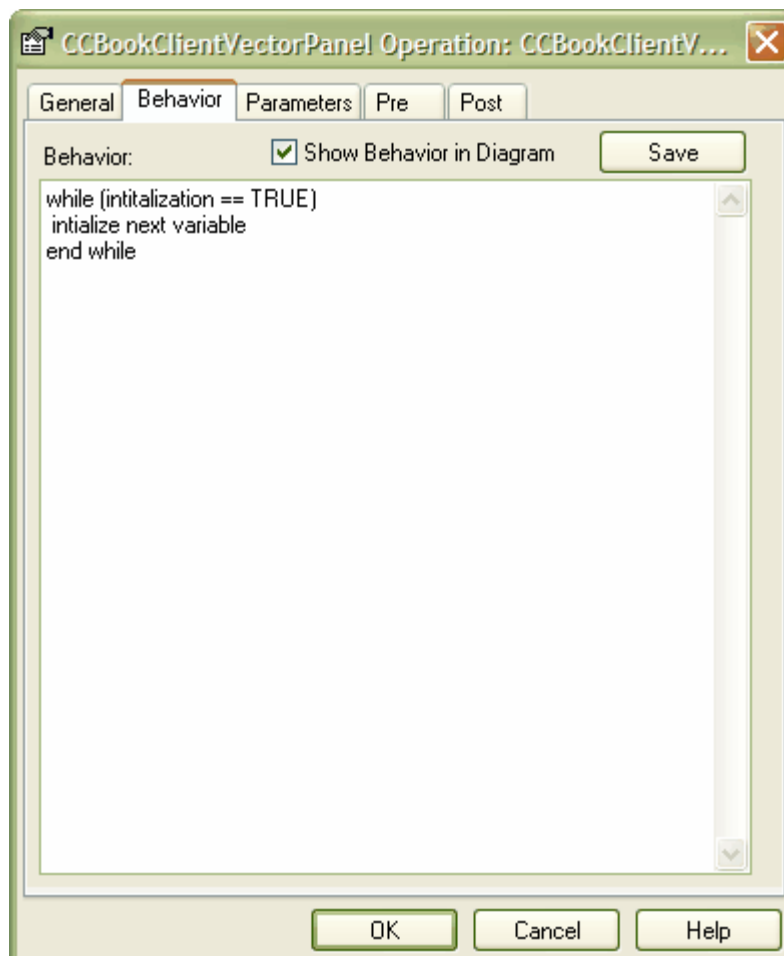
This example illustrates how to use this field to elaborate a method's function in a diagram.



Showing Behavior in a Diagram

To show behavior in a diagram, follow the steps below:

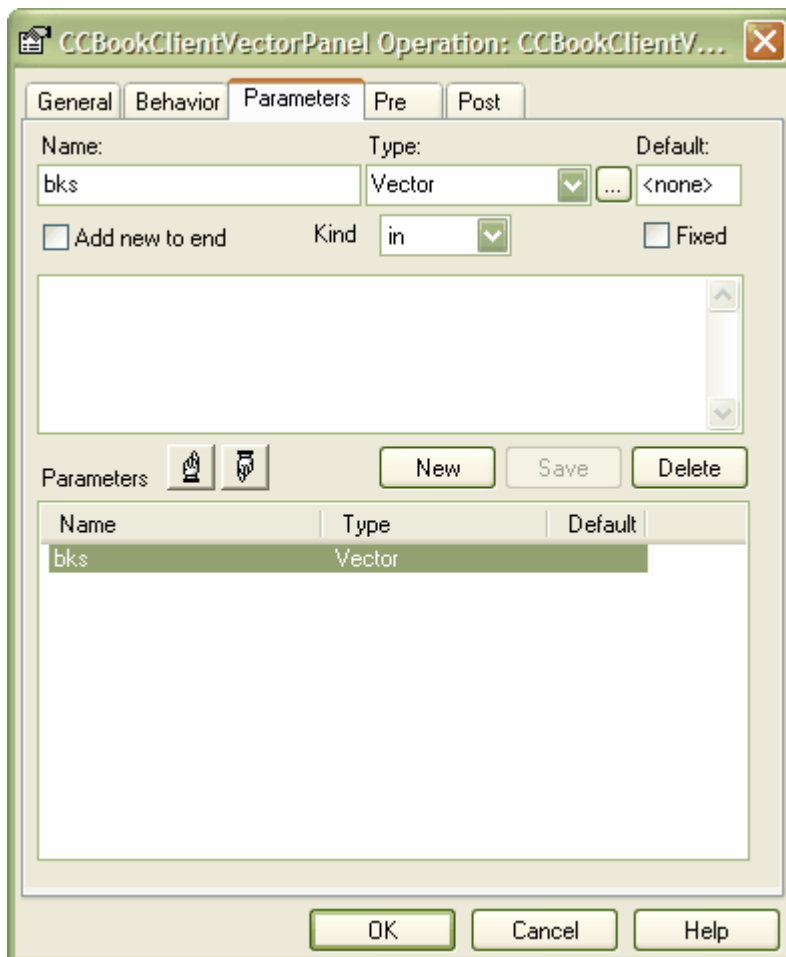
1. Create or locate the required operation.
2. Go to the *Behavior* tab of the *Operations* dialog.
3. Select the *Show Behavior in Diagram* option.
4. Press *Save*.



6.2.4.3.3 Operation Parameters

The *Parameters* tab in the *Operations* dialog lets you define the parameters an operation will have. The parameter list will be reproduced in code in the order they appear in the parameters list - so use the up and down arrows to move parameters into their required positions. Additionally, you may select the *Add new to end* option to force new parameters to appear at the end of the list instead of the head.

Tip: Set the amount of parameter detail to display in a specific diagram using the *Show Parameter Detail* drop down list on the *Diagram Properties* dialog. The setting applies only to the current diagram. The default is to show the type only.



Control	Description
Name	Parameter name
Type	Data type of parameter
Default	Optional default value
Kind	Indicates the way a parameter is passed to a function <ul style="list-style-type: none">• In = By Value• InOut = By Reference• Out is passed by Reference - but only the return value is significant
Fixed	The parameter is 'const' - even if passed by reference
Add new to end	Place new parameters at the end of the list instead of the start
Notes	Free text

6.2.4.3.3.1 Operation Parameter Tagged Values

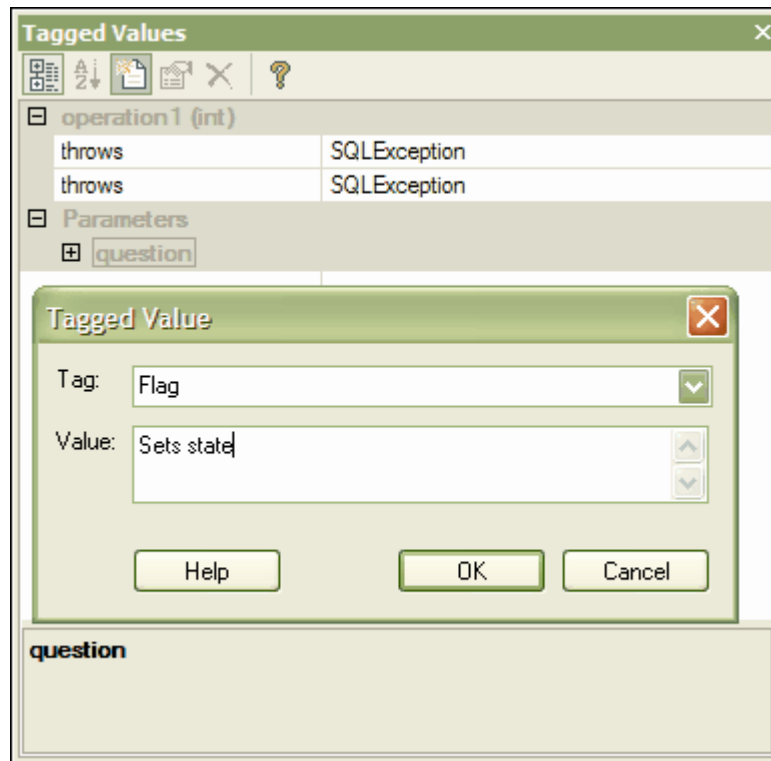
Operation parameters may have tagged values associated with them. Tagged values offer a convenient extension mechanism for UML elements - so you can define any tags you like - and then assign them values using this form.

Tagged values will be written to the XMI output, and may be input to other third party tools for code generation or other activity.

Add a Tagged Value

To add a tagged value for a parameter, use the following the steps :

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the operation by double clicking on the operation containing the parameter in the diagram or on the operation containing the parameter in the Project View.
3. The *Tagged Values* window will now have the operation with parameter selected,
4. Select the parameter from the Parameters compartment in the Tagged Values window press either the *New Tags* button or the *Ctrl + N* hotkey combination.
5. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
6. Then press *OK* the button to confirm the parameter tag.

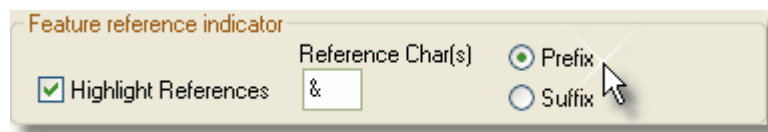


Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

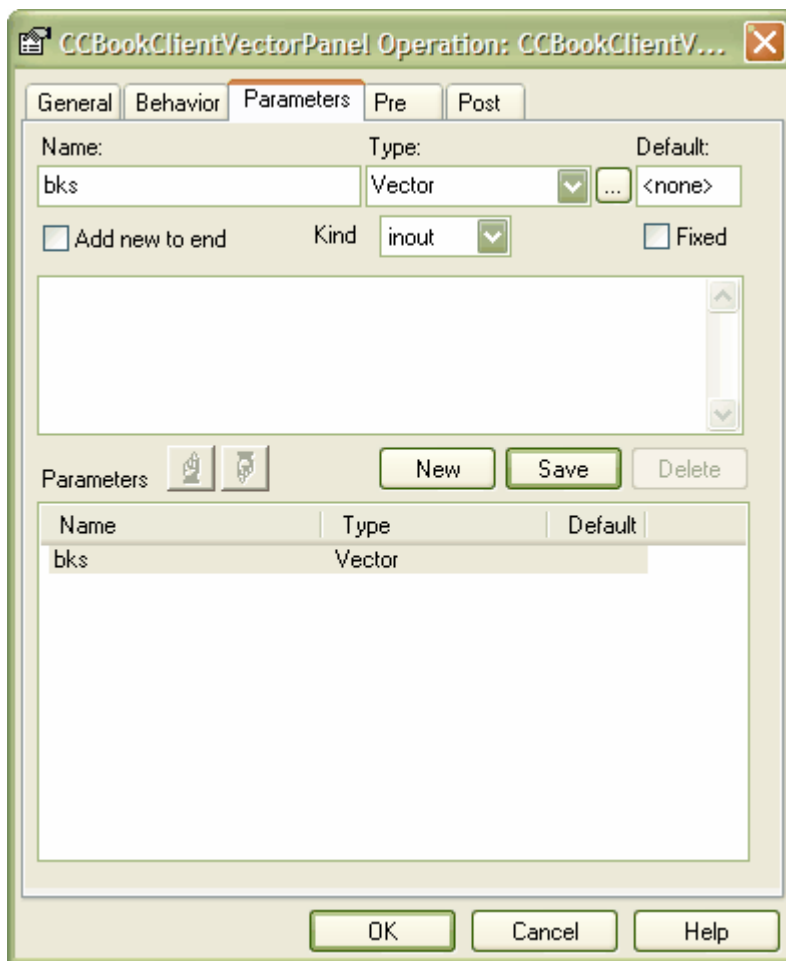
6.2.4.3.4 Operation Parameters by Reference

You can elect to highlight parameters declared as type 'inout' with an additional user-defined prefix or suffix. In the *Objects* section of the *Local Options* dialog (*Tools | Options*), there is a segment which allows you to set whether references are highlighted or not.

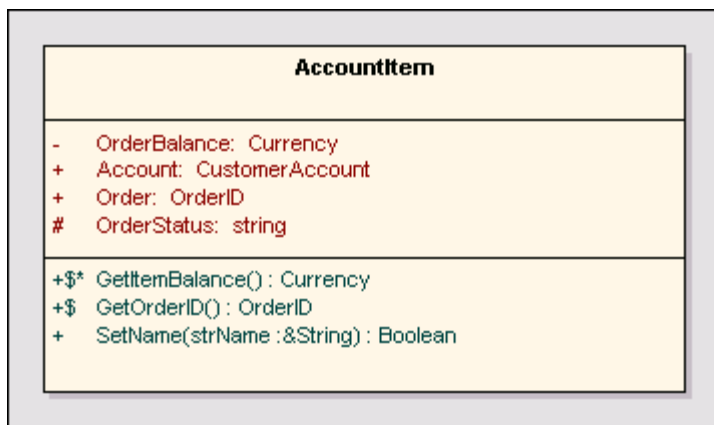
If you select the *Highlight References* option, you can also indicate whether a prefix or suffix will be used, and the actual character to use. In the example below, the '&' character as a prefix has been selected.



When you declare a parameter of type 'inout', it is assumed you are passing the parameter by reference, rather than by value. If you have elected to highlight references, then this will be displayed in the diagram view.



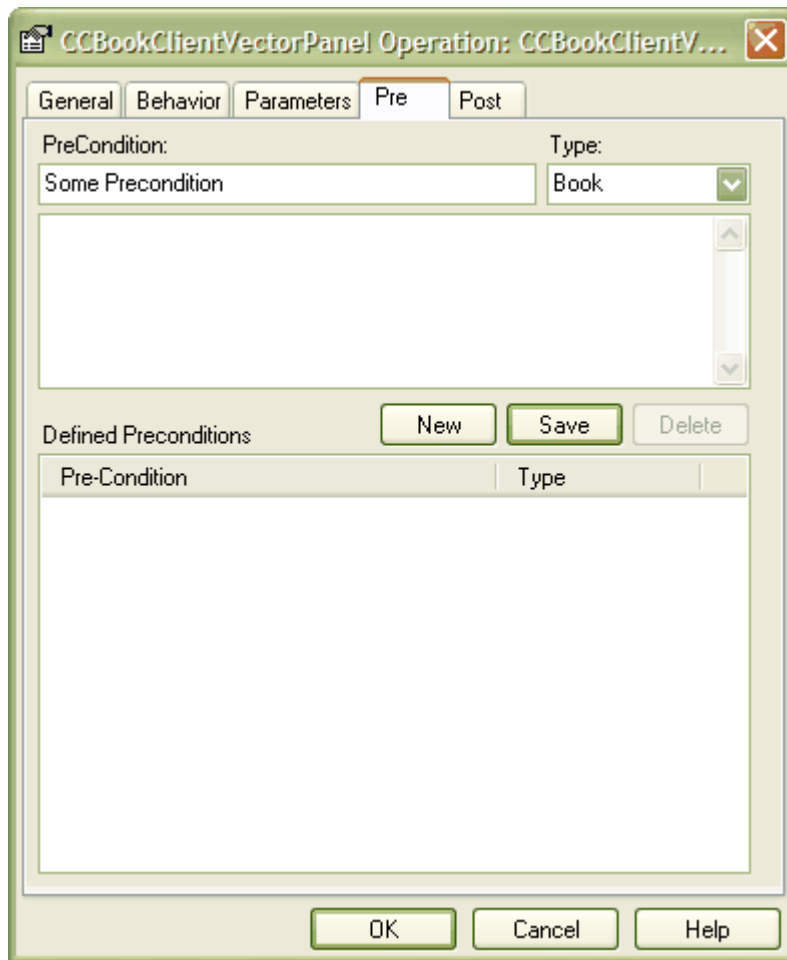
The example below shows that the parameter `strName` is a String reference - and is highlighted using the chosen character and position.



6.2.4.3.5 Operation Constraints

Operations may have pre- and post- conditions defined. For each type, give the condition a name, a type and enter notes.

Constraints define the contractual behavior of an operation - what must be true before they are called and what is true after. In this respect they are related to the state model of a class and can also relate to the guard conditions that apply to a transition.



6.2.4.3.6 Operation Tagged Values

Operations may have tagged values associated with them. Tagged values offer a convenient extension mechanism for UML elements - so you can define any tags you like - and then assign them values using this form.

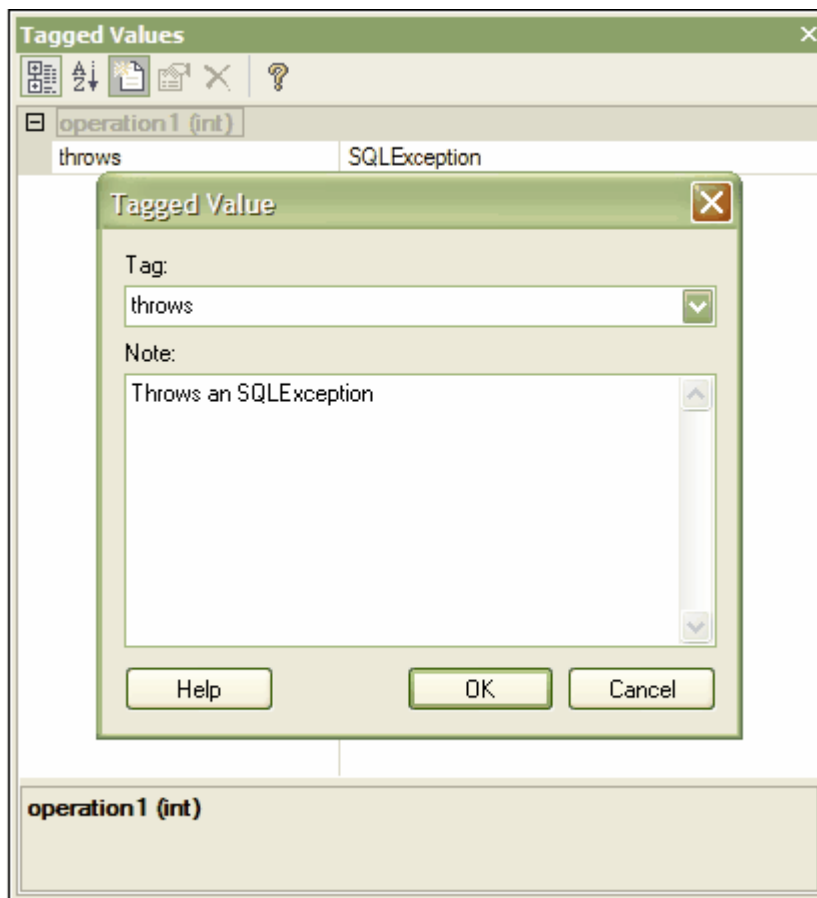
Tagged values will be written to the XMI output, and may be input to other third party tools for code generation or other activity.

Tip: Tagged values are supported for Attributes, Operations, Objects and Connectors.

Add a Tagged Value

To add a tagged value for an operation, use the following steps :

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the operation by double clicking on the operation in a diagram or on the operation in the Project View.
3. The *Tagged Values* window will now have the operation selected, press either the *New Tags* button or the *Ctrl + N* hotkey combination.
4. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
5. Then press *OK* the button to confirm the operation.

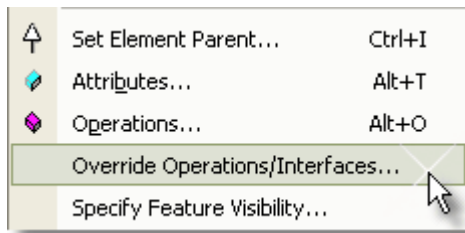


Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

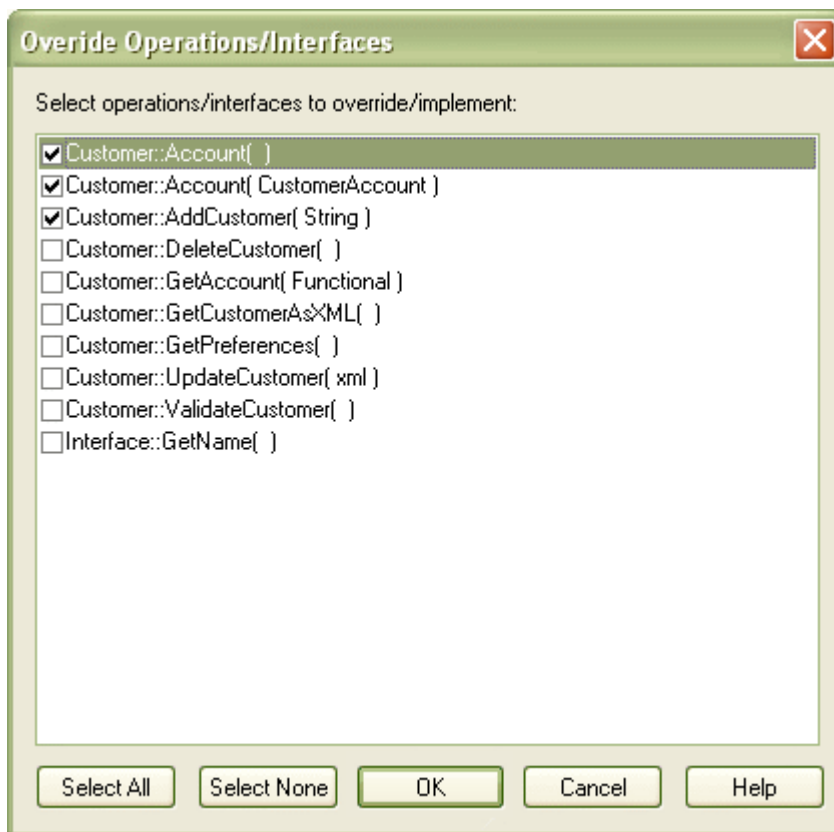
6.2.4.3.7 Override Parent Operations

It is possible in EA to automatically override methods from parent classes and from realized interfaces.

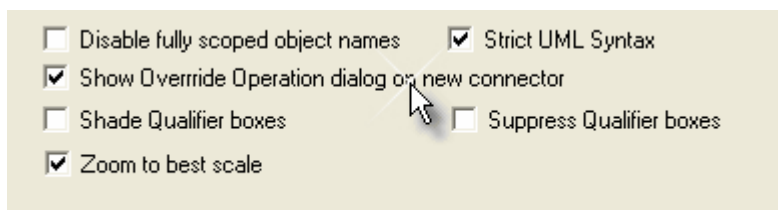
Select a class that has a parent or realized interface and choose *Override Implementation* from the *Element* menu.



In the *Override Operations/Interfaces* dialog, check the operations/interfaces that you wish to automatically override and press **OK**. EA will generate the equivalent function definitions in your child class.



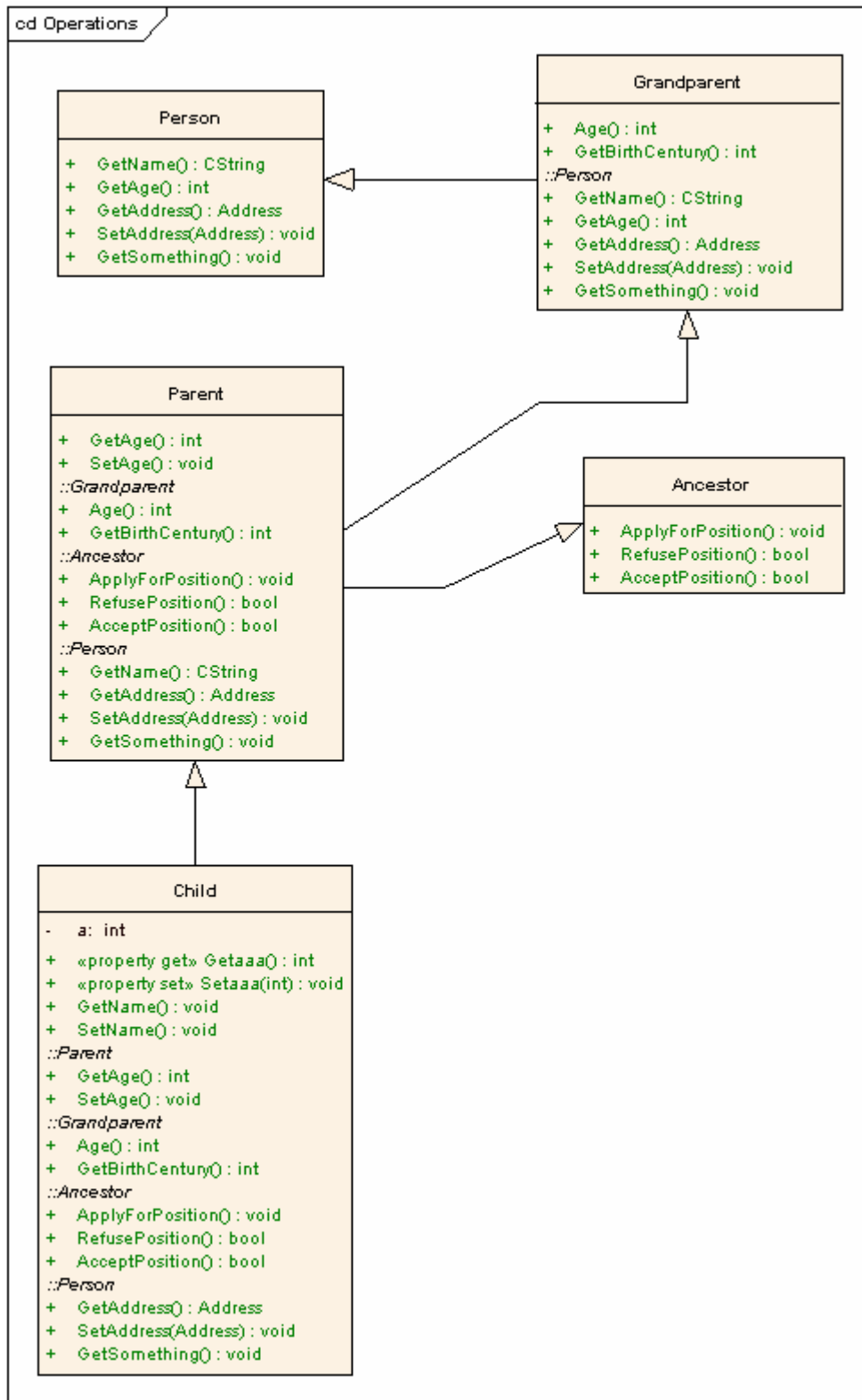
It is possible to configure EA to display this dialog each time you add a Generalization or Realization link between classes and their possible operations/interfaces to override/implement. Do this from the *Diagram* section of the *Local Options* dialog (*Tools | Options*).



6.2.4.3.8 Displaying Inherited Operations

It is possible to configure an element in a diagram to display the complete Operation set obtained from all ancestors in the elements type hierarchy, as well as those directly owned. To do this, use the [Specify Feature Visibility](#) function from the main menu, or use the Ctrl+Shift+Y shortcut key.

The following diagram illustrates this behavior when enabled for each element in a simple hierarchy.

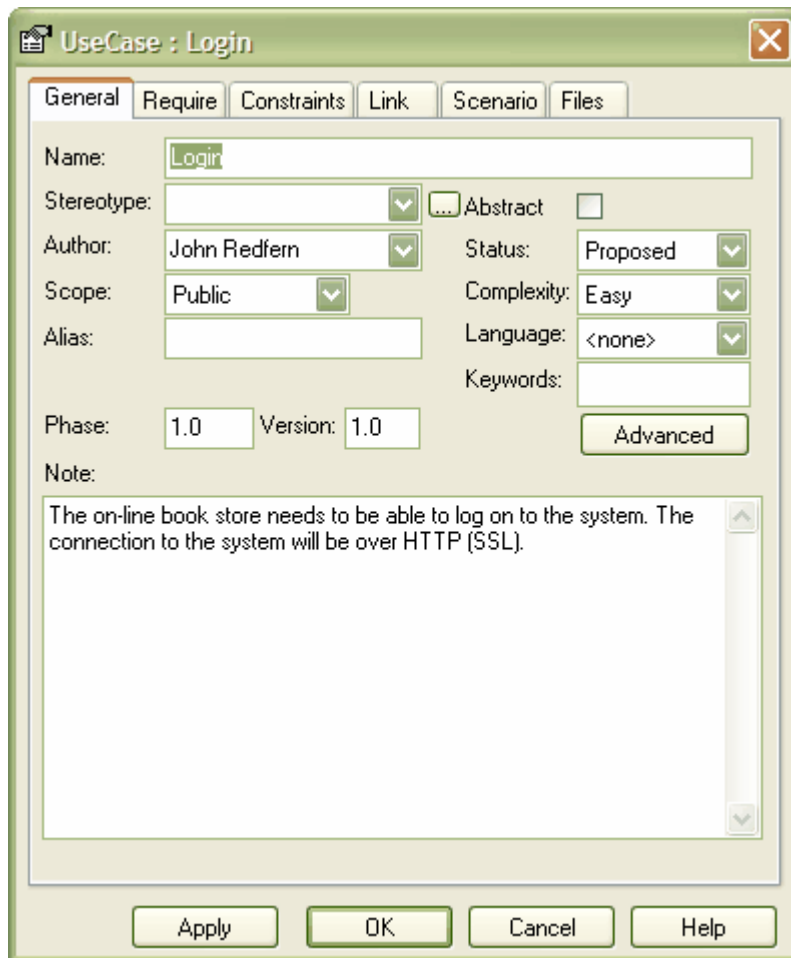


6.2.5 Element Properties

This topic covers element properties and their settings, responsibilities, constraints, links, scenarios, tagged values, associated files, object files and classifiers and boundary element settings.

6.2.5.1 Properties

Below is the *Element Properties* dialog:



Any one of the following options allows you to view this dialog:

- Select an element in the diagram view and select *Properties...* from the *Element* menu.
- Right click on an element in the diagram view, and select *<element type> Properties* from the context menu.
- Select an element in the diagram view, and press *Alt+Enter*.
- Double-click on an element in the diagram view.
- Right click on an element in the Project Browser, and select *Properties...* from the context menu.

The following topics describe each of the tabs in this dialog in detail.

6.2.5.2 General Settings

The *General* tab of the *Element Properties* dialog is shown below:

The screenshot shows a dialog box titled "Class : Message" with a close button in the top right corner. The dialog has several tabs: "General", "Detail", "Require", "Constraints", "Link", "Scenario", and "Files". The "General" tab is selected. The "Name" field contains "Message". The "Stereotype" field is empty with a dropdown arrow. To the right of the "Stereotype" field is an "Abstract" checkbox, which is unchecked. The "Author" field contains "John Redfern" with a dropdown arrow. The "Status" field contains "Proposed" with a dropdown arrow. The "Scope" field contains "Public" with a dropdown arrow. The "Complexity" field contains "Easy" with a dropdown arrow. The "Language" field contains "C++" with a dropdown arrow. The "Persistence" field is empty with a dropdown arrow. The "Keywords" field is empty. The "Phase" field contains "1.0" and the "Version" field contains "1.0". There is an "Advanced" button to the right of the "Phase" and "Version" fields. Below these fields is a "Note" section with a text area containing "@created 23-Aug-2004 02:54:36 PM". At the bottom of the dialog are four buttons: "Apply", "OK", "Cancel", and "Help".

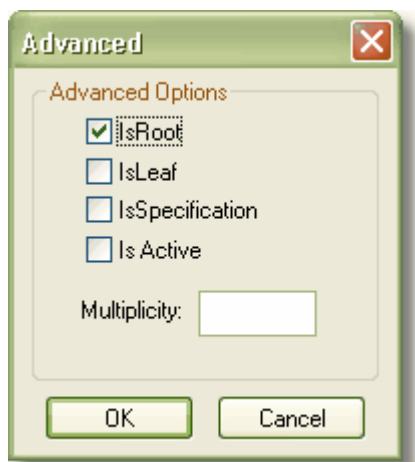
From here you can do the following:

Control	Description
Name	Change the element's name
Stereotype	Select a stereotype for the element (optional)
Abstract	Check to indicate element is abstract
Author	Enter or select the name of the original author
Status	Indicate the current status of the element (eg. Approved, Proposed...)
Scope	Indicate the element's scope (public, private, protected, package)
Complexity	Indicate the complexity of the element (used for project estimation). Assign Easy, Medium or Hard
Alias	Enter an alias (alternate display name) for the object.
Language	Select the programming language for the object
Keywords	A free text area that may be filtered in Use Case Metrics and Search dialogs - typically used for keywords, context information etc.
Advanced button	See Advanced Settings for details
Phase	Indicate the phase this element will be implemented in (eg. 1, 1.1, 2.0 ...)
Version	Enter the version of the current element
Notes	Enter any free text notes associated with the element

6.2.5.3 Advanced Settings

Some elements support some additional attributes. These are Generalizable elements - and by pressing *Advanced* on the *Element Properties* dialog you can set the following:

- *IsRoot* - the element is a root element and may not be descended from another
- *IsLeaf* - the element is 'final' and may not be a parent for other elements
- *IsSpecification* - the element is a specification
- *IsActive* - the element is active - eg. an [active class](#)
- *Multiplicity* - multiplicity setting for the element



6.2.5.4 Requirements

The *Require* tab of the *Element Properties* dialog is shown below. Use this page to create requirements that this element is designed to meet. Requirements are of two types - internal requirements (responsibilities) and external requirements (system requirements). EA will show both types, but you can only edit the internal type from this tab.

These form the functional requirements of the system to be built. The meaning of the requirement may vary depending on which element is the host - for example a business process requirements may mean something different to a Use Case requirement, which again may mean something different to a Class requirement. Use these as best suit your needs.

Use the [Specify Feature Visibility](#) function to show Requirements for an element on the diagram directly (it is also possible to show inherited Requirements in this fashion).

Note: External requirements are those linked to this element using a Realization link

UseCase : Login

General **Require** Constraints Link Scenario Files

Requirement: 1. Login to System Type: Functional

Status: Approved Difficulty: Medium Priority: Medium Last Update: 1/11/2004

Move External New Save Delete

Defined

Requirement	Type	External
1. Login to System	Functional	
1.01	Functional	Yes
2. Go to the Registered screen for non- ...	Functional	
3. Encrypt user details over HTTPS	Functional	Yes
4. Logout of system	Functional	

Apply OK Cancel Help

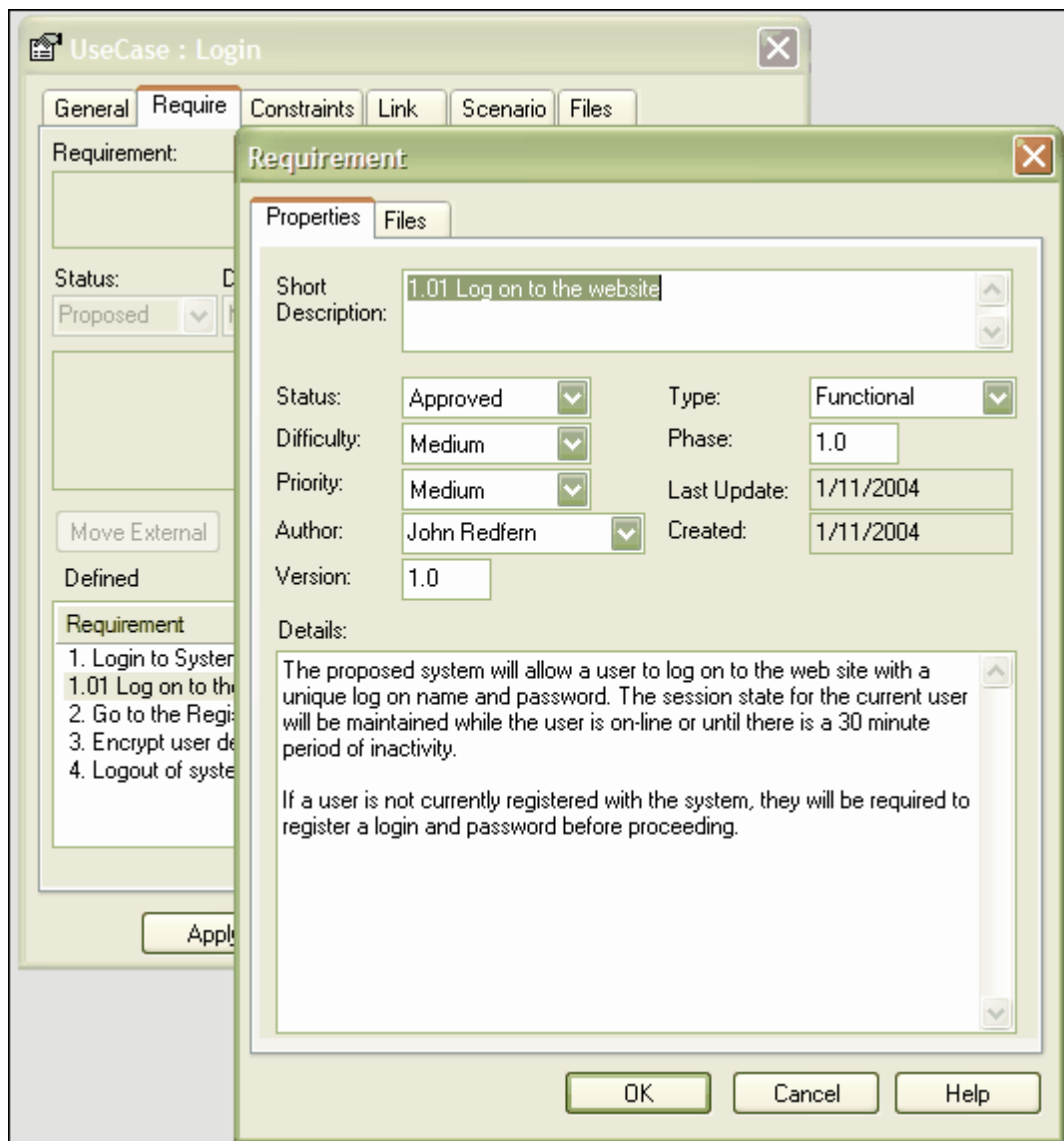
Control	Description
Requirement	Name and high level detail of requirement
Type	Functional, non-functional etc.
Status	Current status of requirement
Difficulty	Complexity of implementing current requirement
Priority	How urgent the requirement is
Last update	Date of last requirement update
Notes	Details of requirement
Defined requirements	List of defined requirements associated with element

6.2.5.5 External Requirements

External requirements are those requirement elements that have been linked to the current element using a Realization link. By creating the link from the element to the requirement, the element now has a responsibility that it must implement as part of the system solution.

In Enterprise Architect, linked requirements are shown in the *Require* tab of the *Element Properties* dialog - but they are marked external and cannot be directly edited (on selection, the tab fields are grayed out).

Double click an external requirement in the list to activate the *Properties* dialog for the associated requirement, where you can view and modify the requirement details and check the requirement hierarchy details.



6.2.5.6 Constraints

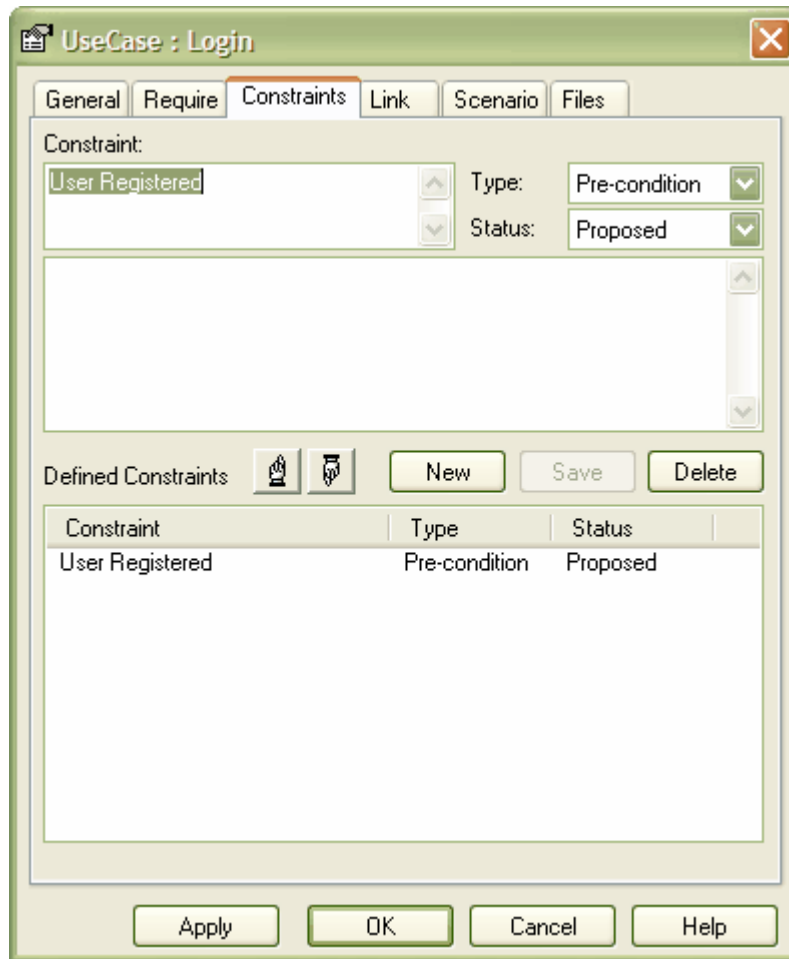
The *Constraints* tab of the *Element Properties* dialog is shown below. Elements may have associated constraints placed on them. These are conditions under which the element must exist and function. Typical constraints are pre- and post- conditions which indicate things that must be true before the element is created or accessed and things which must be true after the element is destroyed or its action complete. Use the *Specify Feature Visibility* function to show Constraints for an element on the diagram directly (it is also possible to show inherited constraints in this fashion).

Adding Constraints to a Model Element

To add constraints to a model element, follow the steps below:

1. Open the *Element Properties* dialog.
2. Select the *Constraints* tab.

3. Enter constraint details.
4. Enter the *Type* and the *Status*.
5. Enter optional notes.
6. Press *Save*.

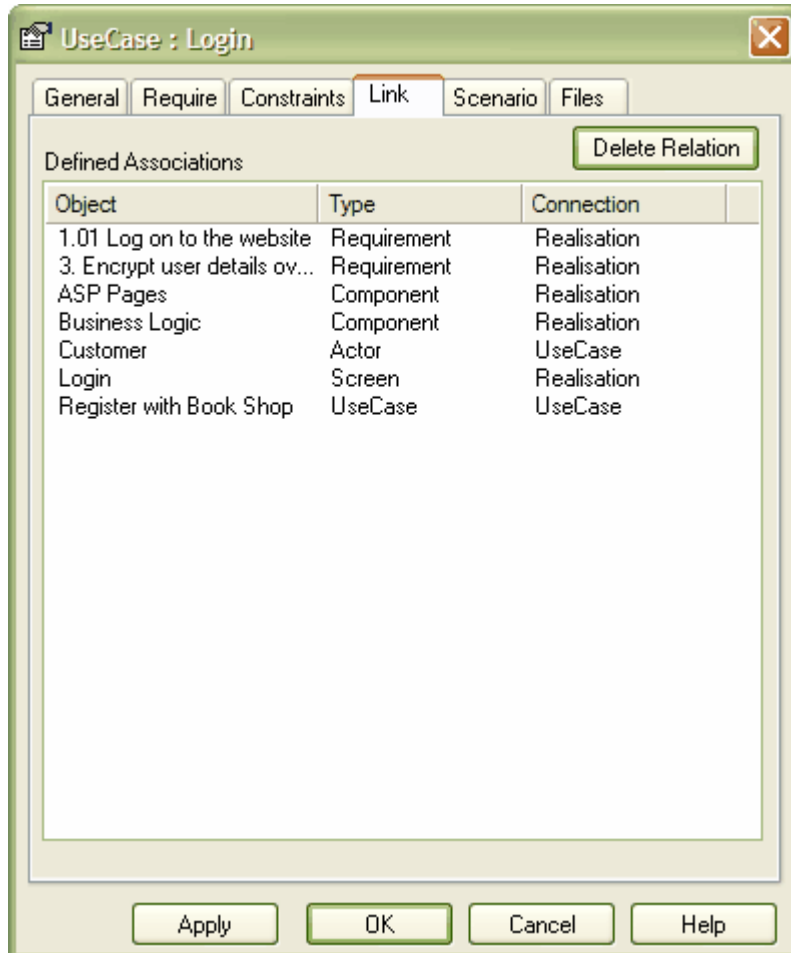


Constraints are used in conjunction with [responsibilities](#) to define the conditions and rules under which an element operates and exists.

Constraint	Details of Constraint
Type	Pre-condition, post-condition or invariant
Status	Current constraint status
Notes	Free text notes associated with constraint
Defined constraint	List of defined constraints

6.2.5.7 Links

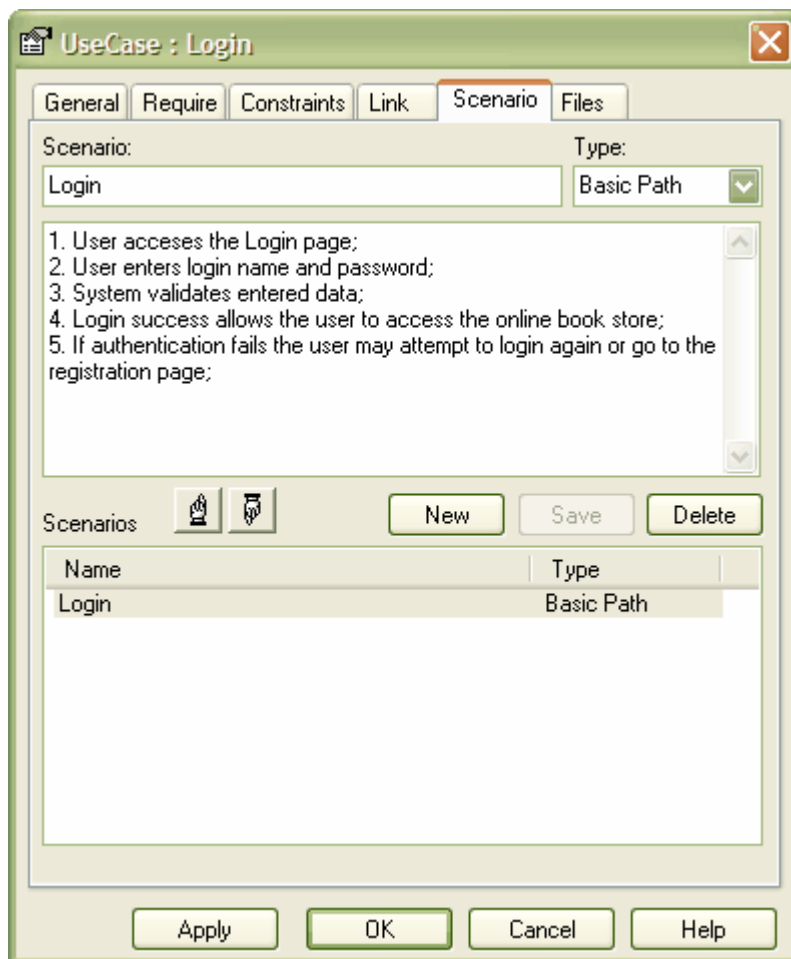
The *Links* tab of the *Element Properties* dialog is shown below. This displays a list of all connections active for the current element. You may delete a connection here if required. Using the right click context menu you may also locate the related element in the Project Browser.



Control	Description
Defined Associations	List of relationships this element has
Delete relation button	Delete the selected relation
Object	The element this element is related to
Type	The type of the related element
Connection	The relationship type

6.2.5.8 Scenarios

The *Scenario* tab of the *Element Properties* dialog is shown below. A scenario is a real world sequence of operations that describes how this element works in real-time. It may be applied to any element and can describe functional behavior, business work flows and end-to-end business processes.



Control	Description
Scenario	Name of scenario
Type	Basic path, alternate path etc.
Notes	Textual description of the scenario - usually depicted in steps of how the user will use the current element
Scenario list	List of defined scenarios

6.2.5.9 Tagged Values

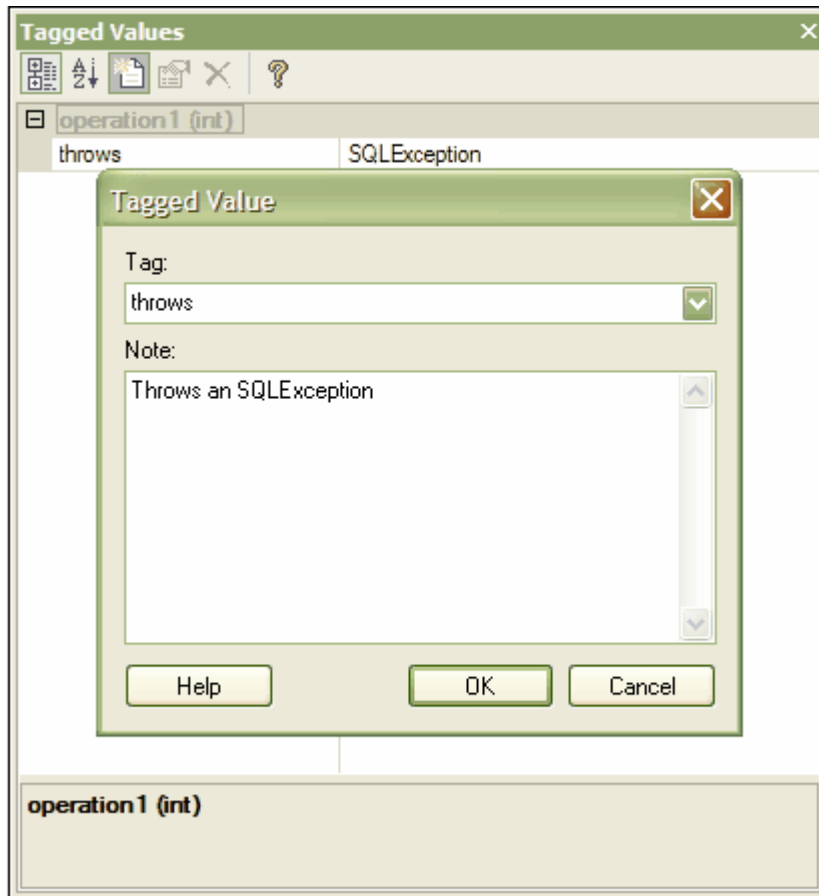
Tagged values are a convenient way of adding additional information to an element outside that directly supported by UML. The UML provides the Tagged Value element for just this purpose. Often these are used during code generation or by other tools to pass information or operate on elements in particular ways. For more information relating to using tags see [The Tagged Values Window](#) topic.

Add a Tagged Value

To add a tagged value for an element, use the following steps :

1. Ensure the **Tagged Values** window is open by selecting **View | Other Windows | Tagged Values** (or press the **Ctrl + Shift + 6** hotkey combination).

2. Select the element by double clicking on the element in a diagram or on the element in the Project View.
3. The *Tagged Values* window will now have the operation selected, press the *New Tag* button.
4. Define the tag in the *Tag* field (or select a custom defined tag from the drop down list), then add notes as appropriate to the *Note* text entry field.
5. Then press *OK* the button to confirm the operation.



Tip: Custom tags may be defined by creating a custom tagged value type for more information refer to the [Creating a Custom Tagged Value Type](#) section.

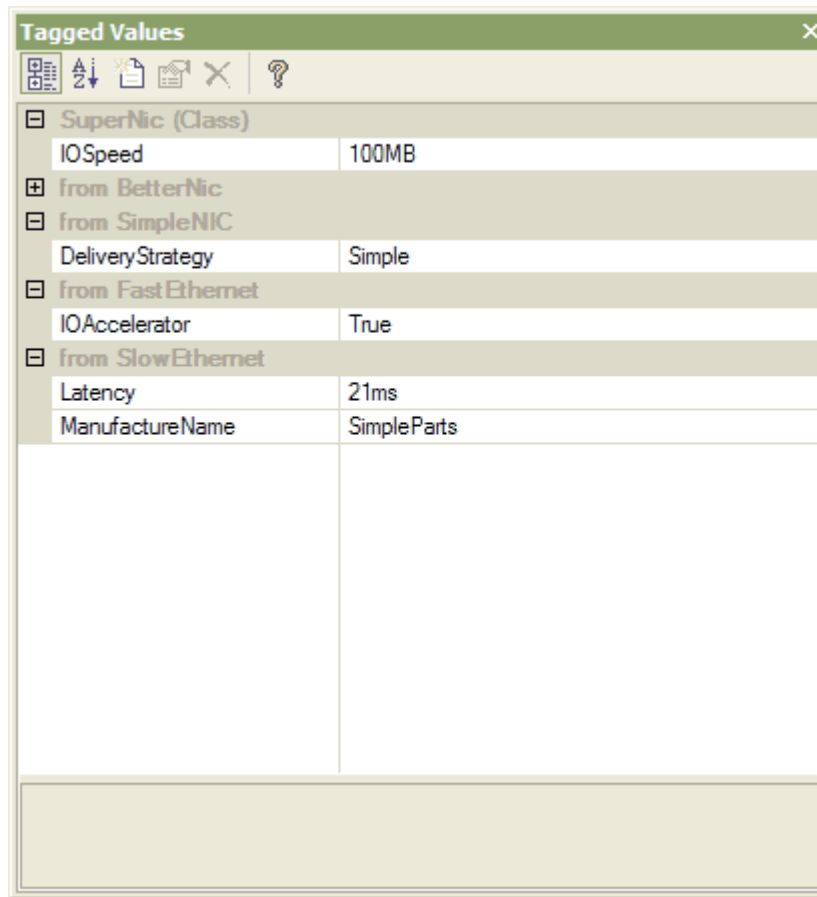
Tagged Values are the preferred method of extending the code generation capabilities of the CASE tool on a per element / per language basis. In future, EA will support more tagged values for different languages.

6.2.5.9.1 Advanced Tag Management

Tagged Values can also be managed within a type hierarchy and with respect to element instances. This additional management is done using the *Tag Management* dialog.

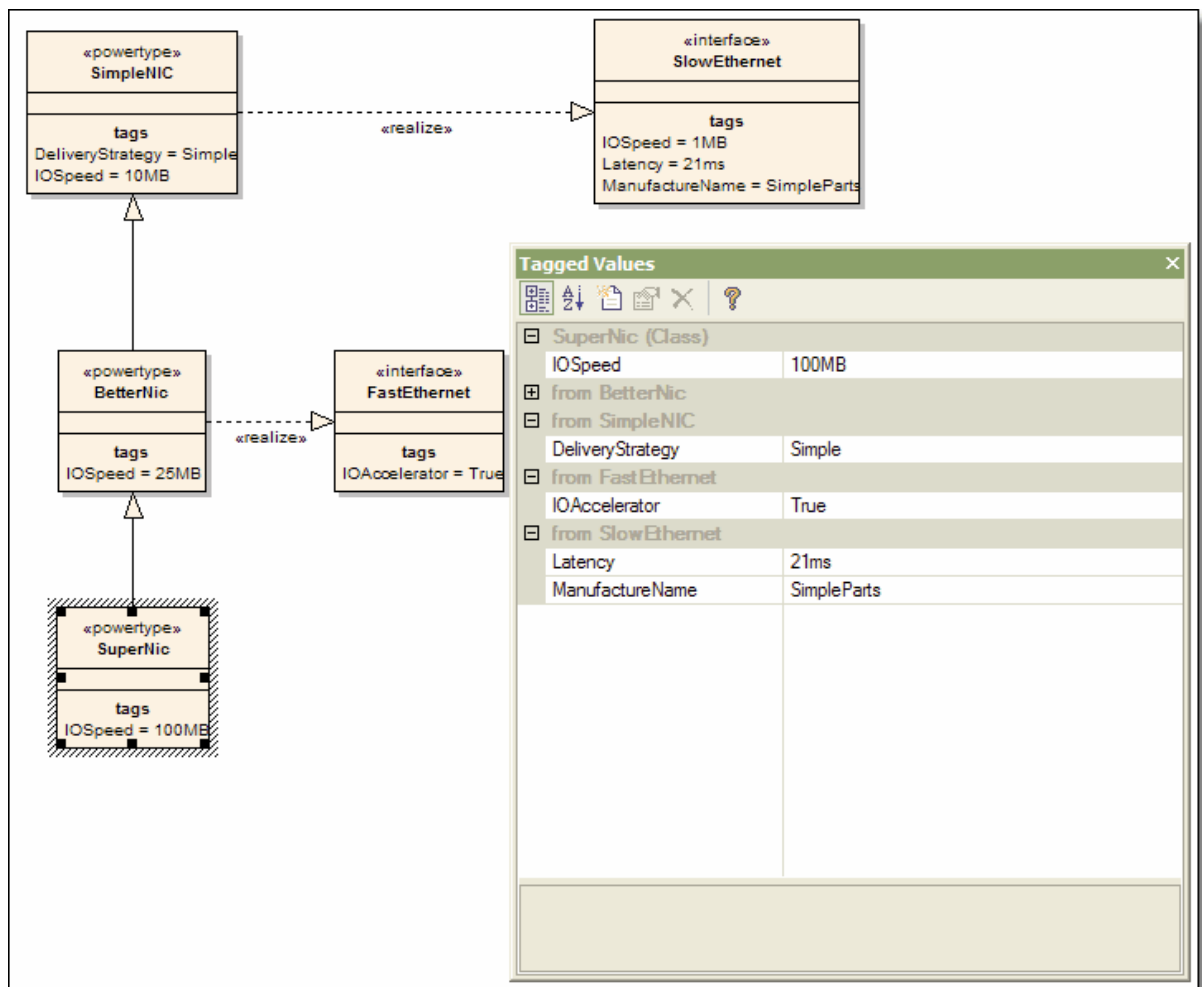
Using the *Tag Management* dialog it is possible to:

1. View tagged values inherited from parent classes or realized interfaces or applied stereotypes.
2. Override Tagged Values derived from parents or applied stereotypes with a unique value for the current element.
3. Delete tagged values from the current element (if a parent version of the Tagged Value exists, it will re-appear in the list after the override is deleted).



The diagram below illustrates a complex tag hierarchy and the way tagged values can be either inherited or overridden in specialized classes to create the final tagged property set for an element.

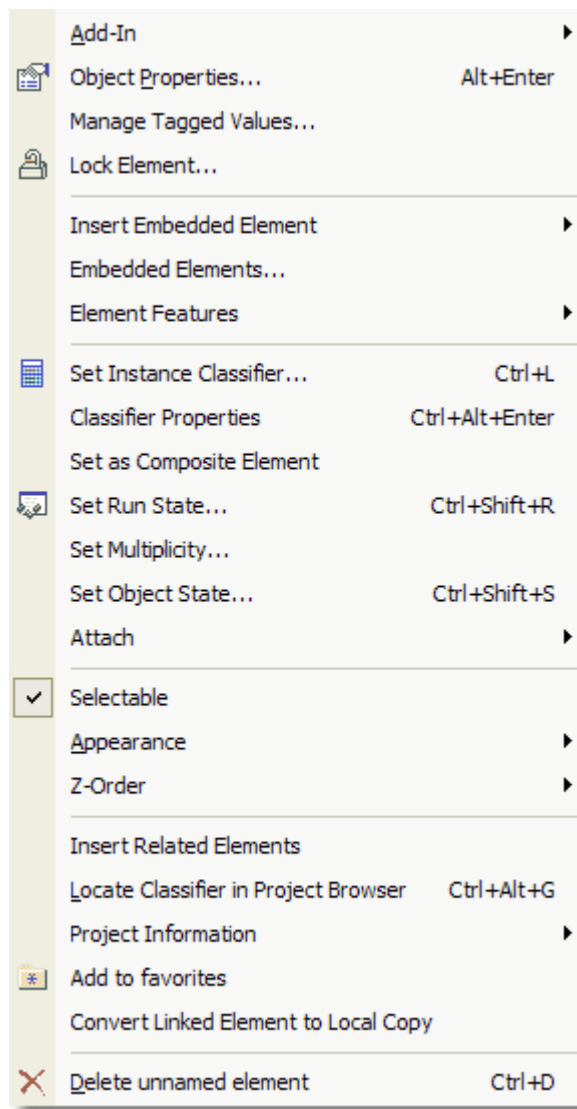
Note also that a similar concept applies to instances - in which case the full tag set is created from the directly owned tags, plus all of those merged in from the classifiers type hierarchy, additional stereotypes and realized interfaces.



6.2.5.9.2 Quick Add of Tagged Values

It is possible to add a single tagged value to one or more elements with a special shortcut.

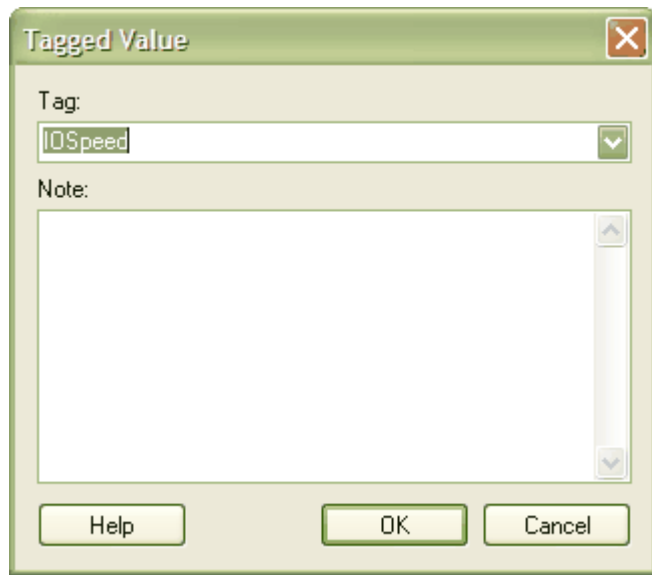
1. From an element context menu (or the context menu of a multi-selection) - choose the *Element Features | Add Tagged Value* option. (Alternatively, select one or more elements and press *Shift+Ctrl+T*).



2. The Tagged Values window will appear - press the New Tag button,
2. This will allow the entry of a *Name for the tag* (required) and *Notes*.
- 4.
5. To add a value for the tag by entering a value in the values section of the *Tagged Values Window* (which will be initially blank).
3. Press **OK** to add your new tagged value to all the currently selected elements.

Note: You can also use the Current Element toolbar ... the last button is a shortcut to the *Add Tagged Value* function.

To delete this property you must open the element property dialog, go to the *Tagged Values* tab and manually delete the item. There is currently no shortcut to delete tags from multiple elements at one time.



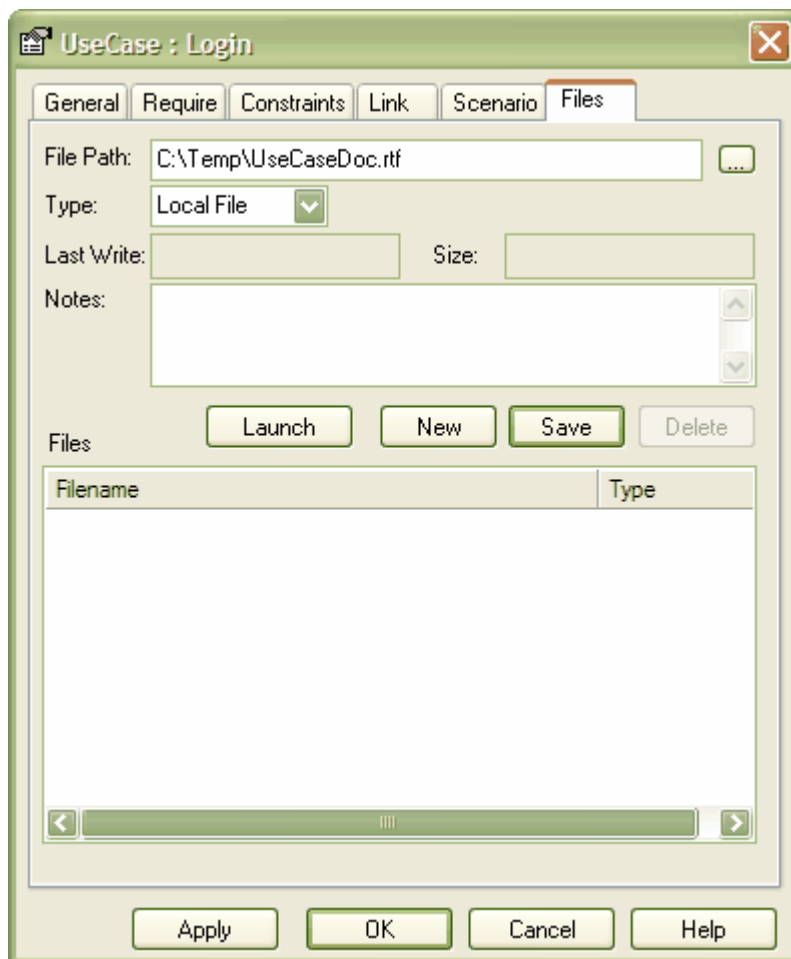
The Notes section will appear in the Tagged Values window in the Info section at the bottom Tagged Values window.

6.2.5.10 Associated Files

The *Files* tab of the *Element Properties* dialog is shown below. An element may be linked to files held somewhere. Use this tab to set associated files for the current element.

Tip: *Linked files are a good way to link elements to additional documentation and/or source code.*

Note that you can also insert [hyperlinks](#) in diagrams to other files - and launch them directly from the diagram. This is an alternative method to that described here.



Control	Description
File path	Name of file
Type	Local file or web address
Notes	Free text about the file
Attached files	List of file
Launch	Open the selected file - local files will open with their default application and web files will open in the default browser

6.2.5.11 Object Classifiers

Many elements in UML model instances of classes - for example, objects, actors and sequence diagram objects. These elements represent real things in a run-time scenario - eg. a Person element named Joe Smith. In UML this is written as "Joe Smith : Person".

As the model develops from a rough sketch to a detailed design, many objects will become examples of defined classes - so in the early analysis phase you may model a "Joe Smith" and a "Jane Smith" - and later a "Person" class from which Joe and Jane will be instantiated.

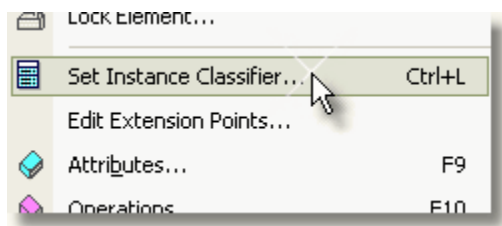
Enterprise Architect allows you to associate an object with its template or class (its classifier). Doing this

greatly increases the descriptive power of the model in capturing the functionality and responsibility of objects at run-time and their associated state. For example, if we describe a Person class with attributes such as Age, Name, Address and Sex, and functions such as GetAge, GetName etc., then when we associate our object with the Person class it is seen to have all the Person class behavior and state (as well as inherited state and behavior from Person's ancestors).

Tip: This is a powerful means of moving your model from the analysis phase into detailed design.

6.2.5.11.1 Using Classifiers

Objects which support classifiers have the option *Set Instance Classifier* in their diagram view context menu. Select this option to choose a single class as the classifier or template for this object.



When you do this, the object name will be displayed as "Object : Class" - for example a Person object named Joe Smith will be displayed as "Joe Smith : Person".

Several Changes Occur if an Object has a Classifier:

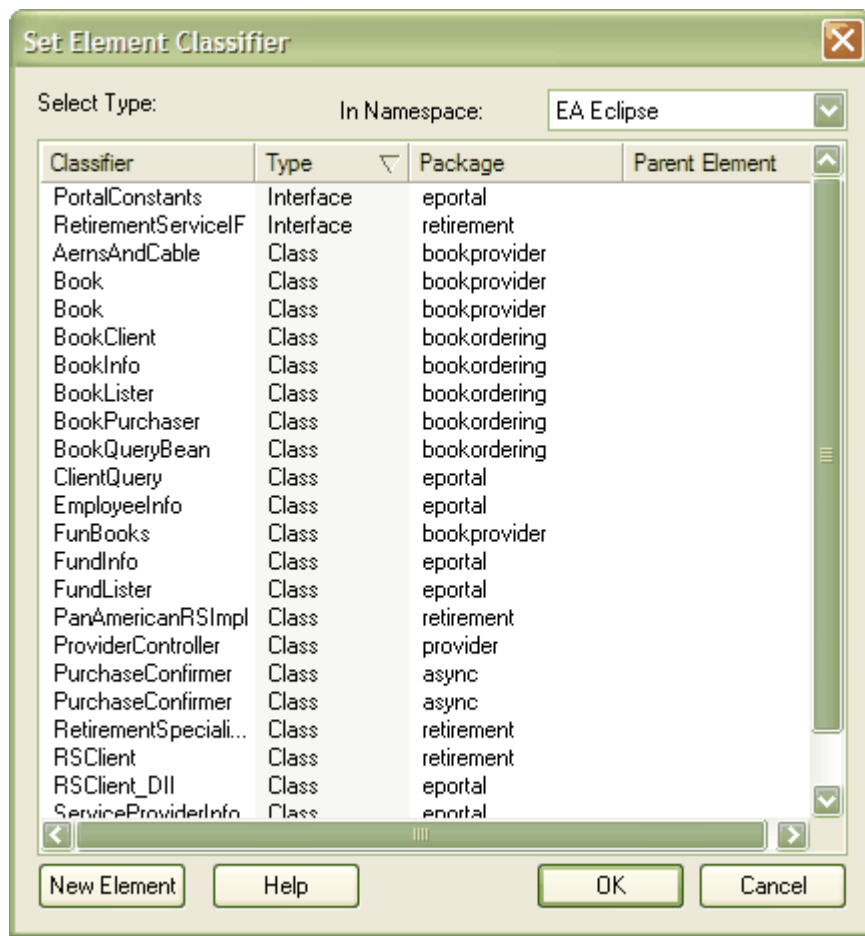
In the context menu for the object, the *Element Features | Attributes* and *Element Features | Operations* menu items will appear. If you select either of these, you will be given the *Attributes* or *Operations* dialog for the classifier, not the object. It is important to remember that the object is only an instance of a class at runtime, so the appropriate attributes and operations are those of the classifier, not the object.

If you set the classifier for an object in a sequence diagram, when you add a message the drop down list of available messages derived from the target object will come from the classifier - not the object selected. This allows you to associate sequence diagram objects with classes and use the defined behavior of the class to model actual behavior at run time.

You may also select a message for a state flow connection. The same rules apply as for sequence objects.

Note that in the message dialog you may also elect to include messages defined in the target classifiers inheritance hierarchy.

The *Set Element Classifier* dialog is shown below. Select the desired item in the list and press *OK* to set the instance classifier.



6.2.5.12 Boundary Element Settings

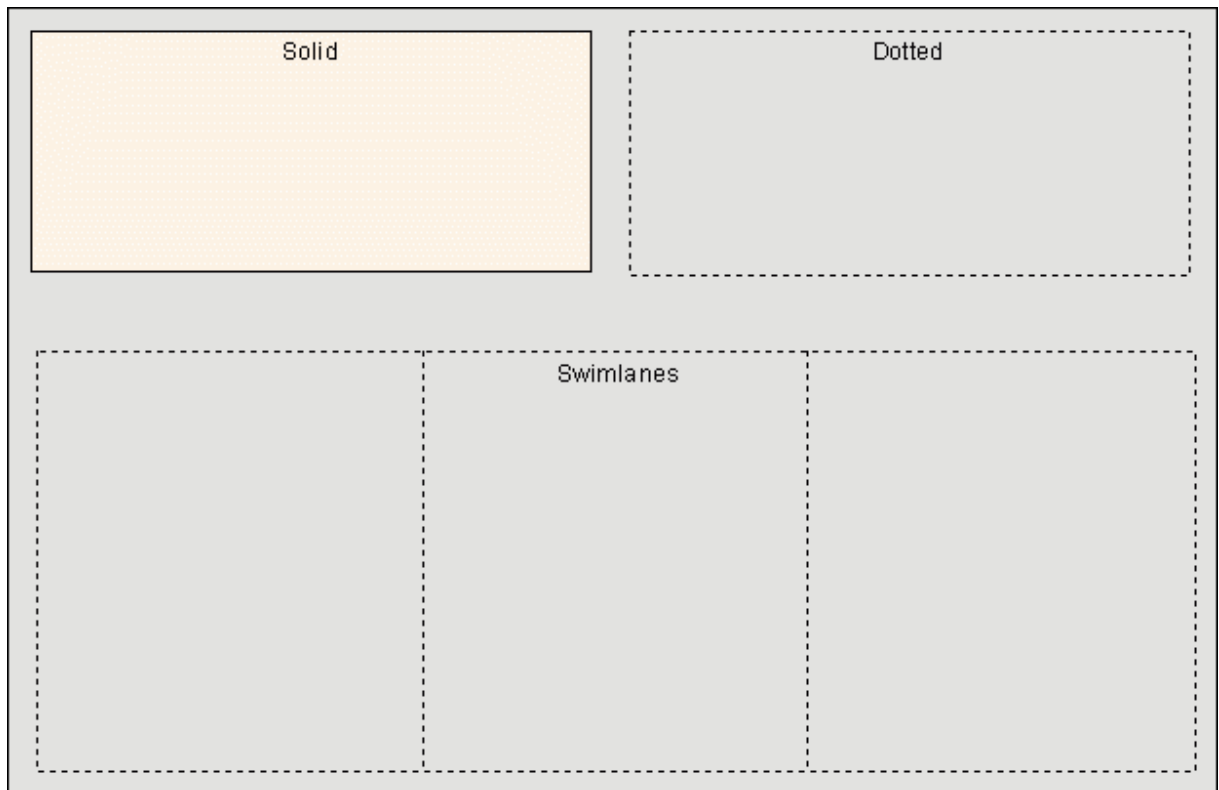
A system boundary element is a non-UML element used to define conceptual boundaries. You can use boundaries to help group logically related elements (from a visual perspective, not as part of the UML model)

Configuring Boundary Elements

Boundary elements may be configured to display in different ways. The main differences are:

- Solid border
- Dotted Border

With horizontal or vertical 'swim lanes' - swim lanes are used to group elements in a vertical or horizontal context (eg. Client, Application and Database tiers could be represented in swim lanes)

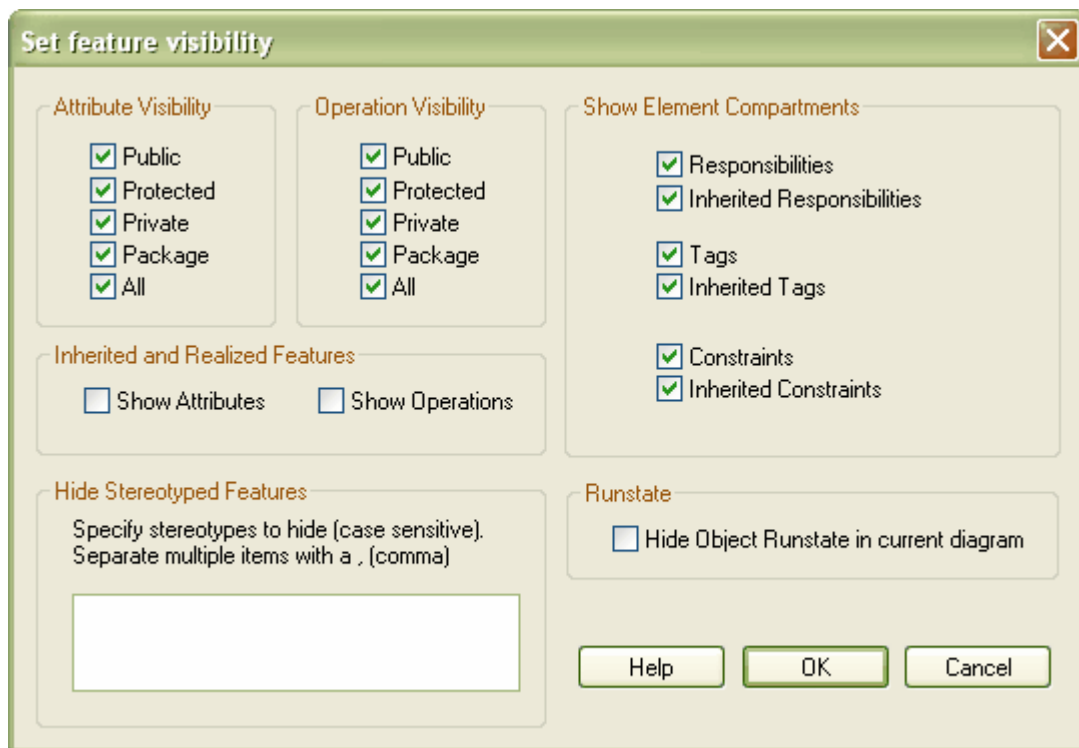


6.2.6 Compartments

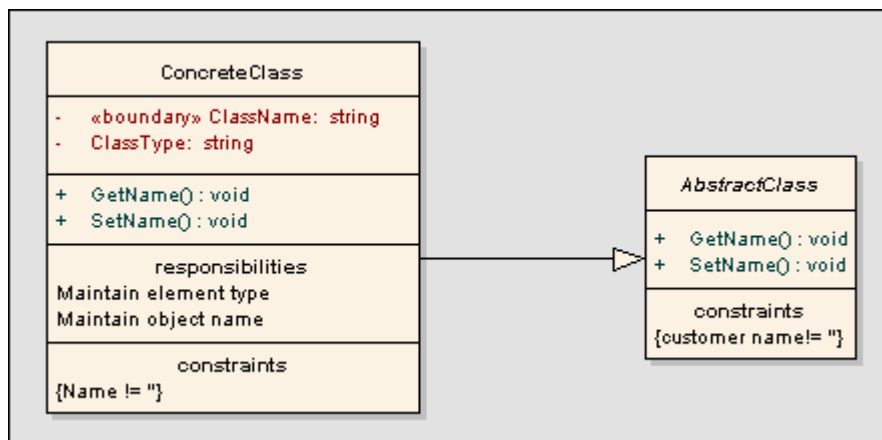
In addition to the two main compartments shown in a class element (attributes and operations), EA also supports three other compartments that may be optionally displayed. These are the Responsibility, Constraint and Tag compartments.

To display one or more of these compartments, follow the steps below:

1. In the diagram view, right click on the element for which you wish to change the display.
2. From the context menu, select *Element Features* | *Specify Feature Visibility*.
3. In the *Set Feature Visibility* dialog, select the options for the compartments you wish to show.



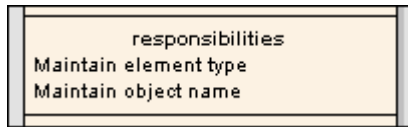
The display will be updated to show the additional compartments - if that element supports compartments.



Note: You can set the compartment visibility of more than one class at the same time by selecting multiple elements, then specifying the compartment visibility for all at the same time.

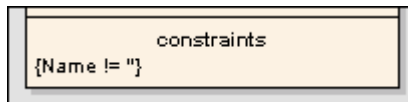
6.2.6.1 Responsibility Compartment

The responsibility compartment shows a list of responsibilities as entered on the *Responsibility* tab of the *Element Properties* dialog.



6.2.6.2 Constraint Compartment

The constraint compartment shows a list of element constraints as entered in the *Constraint* tab of the *Element Properties* dialog.



6.2.6.3 Tag Compartment

The Tag compartment lists all tagged values for an element as entered in the *Tagged Values* tab of the *Element Properties* dialog.



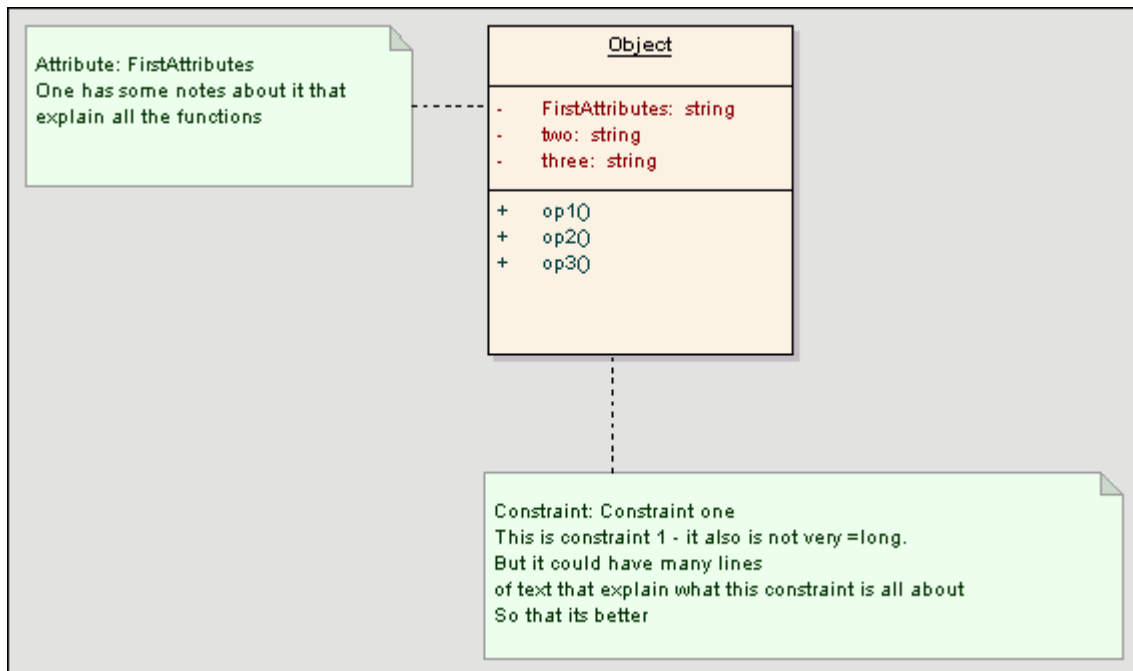
6.2.7 Linking Notes

In Enterprise Architect you can link notes to internal documentation or to an element.

6.2.7.1 Linking Notes to Internal Documentation

It is possible to link a note element to another element's internal documentation. This allows you to externalize model documentation to the diagram level, and as EA keeps the note and the internal structure in synch, you do not have to worry about updating the note contents - this is done automatically.

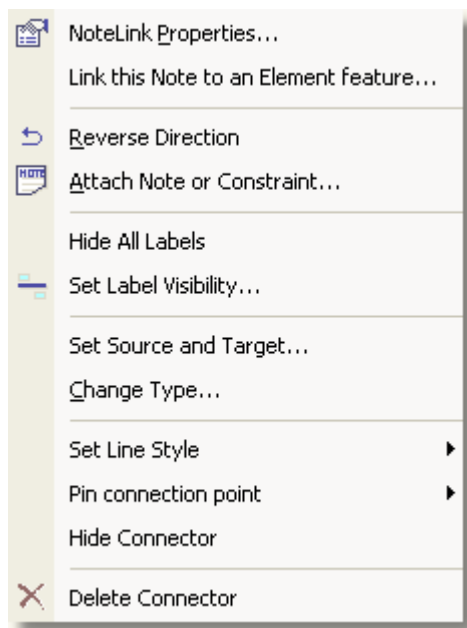
In the example below, two notes are linked into an element's internal structures. One is linked to an attribute, and displays the attribute name and notes. The other is linked to a constraint - showing the constraint name and documentation.



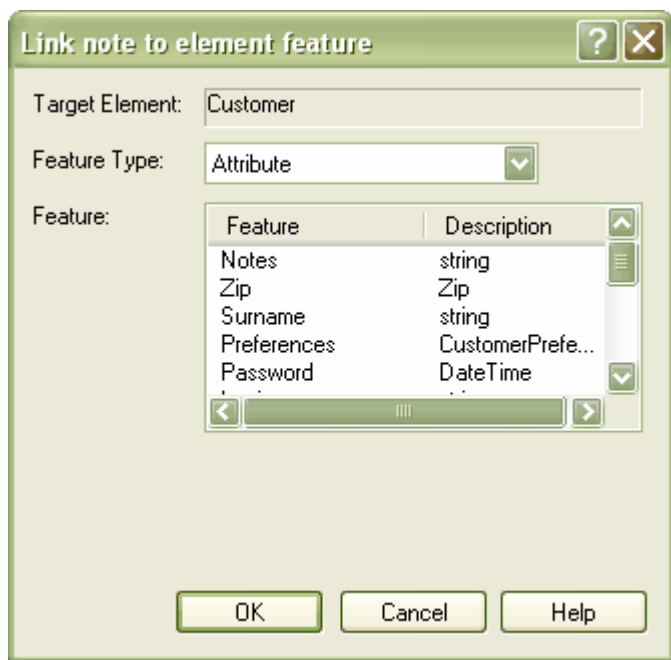
6.2.7.2 Link a Note to an Element

To link a note element (graphical text) to another design element, do the following:

1. Insert the target element into a diagram.
2. Add a note to the diagram, but do not add any text to it.
3. Link the note and the element with a note link connection. The note link connection is located in the [UML Elements toolbar](#) under the *Link* drop down menu - it is the last item on the list.
4. Right click on the note link and select *Link this Note to an Element Feature*.



- In the *Link note to element feature* dialog, select the *Feature Type* to link to.



- Select the exact *Feature* you want from the provided list.
- Press *OK*.

The note will now automatically derive its contents from the target element.

6.3 Working with UML Connections

UML connections, along with elements, form the basis of a UML model. Connections 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.

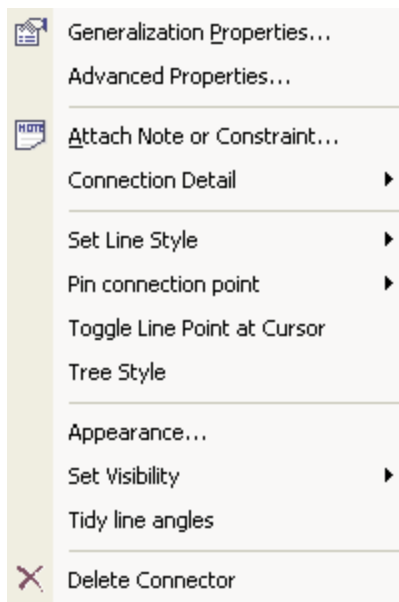
6.3.1 Connection Context Menu

If you right click on a connection in a diagram, the connection context menu opens. This provides quick access to some important functions. The menu is split into seven distinct sections:

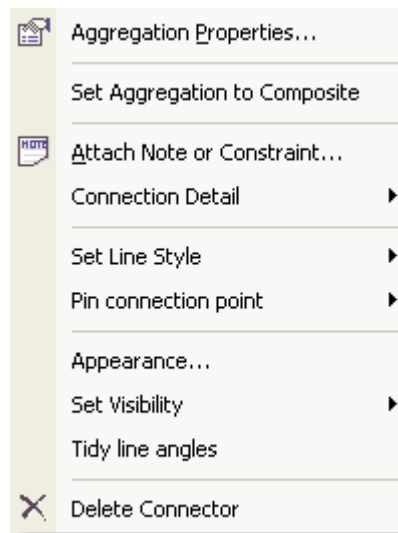
- [Zicom Mentor](#) - this will only show in the first section if you have Zicom Mentor installed and registered.
- [Properties](#)
- [Type Specific](#)
- [Common Actions](#)
- [Style](#)
- [Appearance](#)
- Delete -delete the connection with this option.

Note: Not all menu options will be present on all connection context menus. Context menus vary slightly between connection types. The type specific menu option is not always included, for example.

Example Context Menu for a Generalization:



Example Context Menu for an Aggregation:



6.3.1.1 Properties Menu Section

The Properties menu section on the Connections context menu contains the following options:

Menu Option	Description
<Connection type> Properties	Opens the Properties window for the selected connection.
Advanced Properties	Opens the Advanced Properties dialog.

Note: Not all menu options will be present on all connection context menus. Context menus vary slightly between connection types. The type specific menu option is not always included, for example.

6.3.1.2 Type Specific Menu Section

The Type Specific menu content is specific to the object, and only appears on a few different connections. Some examples are below:

Connection	Menu Option	Description
StateFlow	Message	Set the value of the message.
Aggregation	Set Aggregation to Composite	Change the aggregation to composite.
Aggregation	Set Aggregation to Shared	Appears after Set Aggregation to Composite has been selected. Sets the aggregation to shared.
Dependency	Dependency Stereotypes	Provides a sub-menu to select a stereotype for the dependency.
Trace	Dependency Stereotypes	Provides a sub-menu to select a stereotype for the dependency.
Role Binding	Dependency Stereotypes	Provides a sub-menu to select a stereotype for the dependency.
Represents	Dependency Stereotypes	Provides a sub-menu to select a stereotype for the dependency.
Occurrence	Dependency Stereotypes	Provides a sub-menu to select a stereotype for the dependency.

Note: Not all menu options will be present on all connection context menus. Context menus vary slightly between connection types. The type specific menu option is not always included, for example.

6.3.1.3 Common Actions Menu Section

The Common Actions menu section on the Connections context menu contains the following options:

Menu Option	Description
Attach Note or Constraint	Attach a note or attach a constraint to the connection.
Connection Detail	See table below for sub-menu options.

Element Features Sub-Menu

Menu Option	Description
Set Source and Target	Change the source and/or target of the connector.
Change Type	Change the connector type .
Reverse Direction	Reverse the direction of the connector - eg. if the connector is an arrow, the arrowhead will swap to the other end.

Note: Not all menu options will be present on all connection context menus. Context menus vary slightly between connection types. The type specific menu option is not always included, for example.

6.3.1.4 Style Menu Section

The Style menu section on the Connections context menu contains the following options:

Menu Option	Description
Set Line Style	Set the connector line style - options are direct, auto routing or custom line.
Pin Connection Point	Pin the start and/or connector ends to a position on the target element.
Toggle Line Point at Cursor	Inserts an anchor point on the line at the point of the cursor so you can change the shape of the line.
Tree Style	Create a tree style link (for a generalization connection).

Note: Not all menu options will be present on all connection context menus. Context menus vary slightly between connection types. The type specific menu option is not always included, for example.

6.3.1.5 Appearance Menu Section

The Appearance menu section on the Connections context menu contains the following options:

Menu Option	Description
Appearance	Set the line color and line thickness of the connector.
Set Visibility	See table below for sub-menu options.
Tidy Line Angles	Tidy the line angles of a custom connector.

Set Visibility Sub-Menu

Menu Option	Description
Hide Connector	Hide the connector. To show the connector again, follow the instructions on the Hide/Show Connector page.
Hide Connector in Other Diagrams	Hide or show the connector in other diagrams .
Hide All Labels	Hide or show all labels attached to the connector.
Set Label Visibility	Hide or show labels attached to the connector on an individual basis.

Note: Not all menu options will be present on all connection context menus. Context menus vary slightly between connection types. The type specific menu option is not always included, for example.

6.3.2 Common Connection Tasks

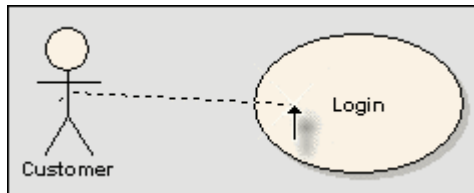
This section details some of the common tasks associated with managing model connections.

6.3.2.1 Connecting Elements

To connect classes, use cases, objects, components or other elements, follow the steps below:

1. In the UML Toolbox, locate the type of association you want - use case link, association, aggregation, etc. Left-click once to select the relationship.
2. Place the mouse cursor over the first diagram element. Press and hold down the left mouse button (the cursor should change to a vertical arrow).
3. Drag to the second diagram element and release the mouse button. During this process a dotted line will indicate from which element and to where you are creating a link.
4. To repeat the last connection that you have made press the **F3** key.

Tip: You may also drag a connection to position it: left click on the link, then while holding the mouse button down, drag the link to where you want it to appear. Note that there are some limitations on how far or to where you can drag a connection.



Double click on an association to [change properties](#), or right click to [change connection type](#) and [direction](#) if you want.

6.3.2.2 Connector Styles

Connectors come in three different routing styles:

Direct

The line is drawn from element A to element B in a straight line. You may move the line back and forth, up and down etc. to a limited degree.

Auto Routing

The line makes an attempt to route from A to B in a vertical and horizontal manner with 90 degree bends. Line can be pulled around a bit to give good results, but location of bends and number of bends is not configurable.

Custom

Most flexible option. You can add one or more line points and bend and push the line into virtually any shape by using the *Toggle Line Point at Cursor* option..

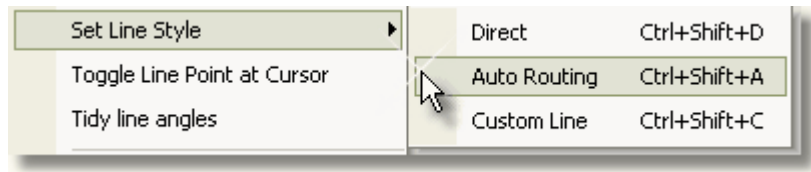
Set the Connector Style

How to set the connector style, follow the steps below:

1. Right click on the connector you wish to change, to open the context menu.
2. Select *Set Line Style*.
3. From the submenu, select the style you require.

-OR-

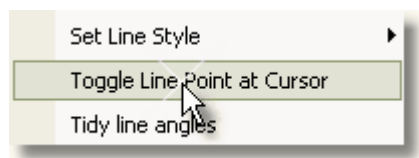
1. Select the connector you wish to change.
2. Use the following keyboard shortcuts to change the style: *Ctrl+Shift+D* for Direct, *Ctrl+Shift+A* for Auto Routing, and *Ctrl+Shift+C* for Custom.



Add or Delete Line Points

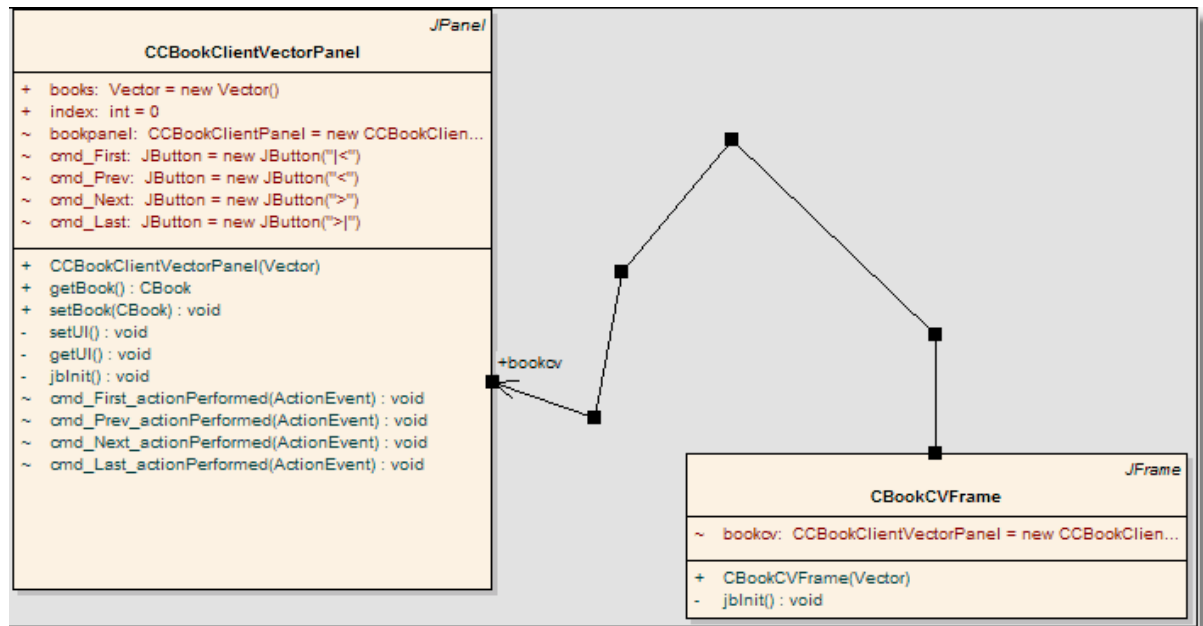
To add or delete line points (custom connector), follow the steps below:

1. Right click on the connection to open the context menu.
2. Set the line style to Custom Line (*Ctrl+Shift+C*), this will enable the *Toggle Line Point at Cursor* option in the context menu.
3. Select the *Toggle Line Point at Cursor* option to add/delete a line point.



-OR-

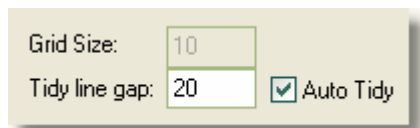
1. Hold down the *Ctrl* key and select a point on the connector the drag the connector into the desired position. This gives the developer the ability to quickly and easily route connectors in the layout that they desire.



Tidy Line Angles

To tidy line angles (custom connector), follow the steps below:

1. Right click on the connection to open the context menu
2. Select the *Tidy Line Angles* option - this will 'nudge' the custom line into good horizontal and vertical increments, saving you the time of trying to get good layouts by hand.



Tip: The tidy line function may also be set to operate by default on the *Diagram Behavior* page of the *Local Options* dialog (*Tools | Options*).

6.3.2.3 Arranging Connections

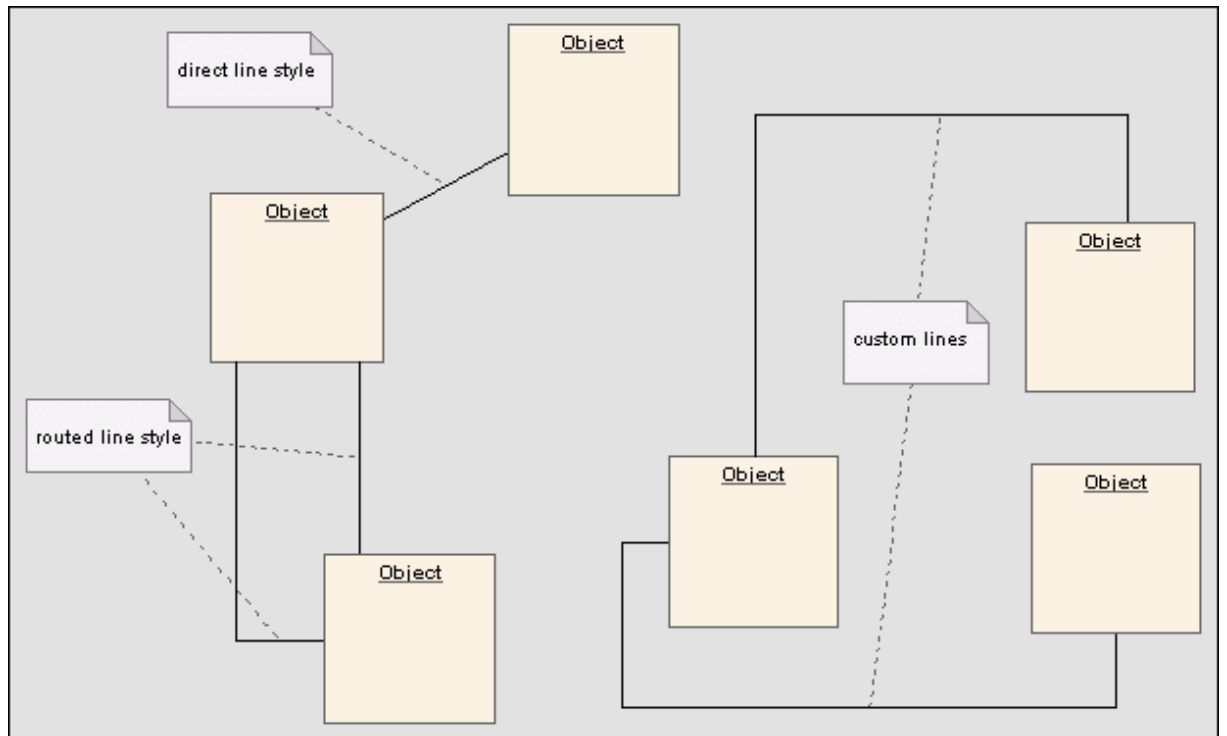
Connections between elements may be moved around so as to facilitate a good layout. There is a limit to how much a connection can be moved around - but generally it is very easy to find an acceptable layout. For the best layouts, use the Custom style - this allows you to add as many line points and bends as you desire to create a clean and readable diagram.

Move a Connection

To move a connection, follow the steps below:

1. Left click once on the connection to select it.
2. Holding the left mouse button down, move the mouse in the direction you wish to move the connection.
3. To refine the movement, click and hold very near to one end of the connection - this will produce a slightly different movement range.

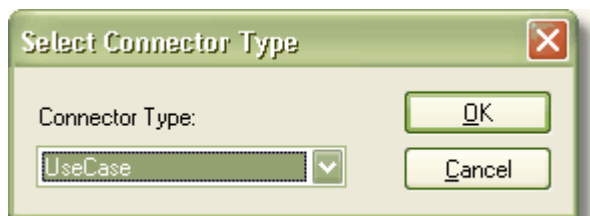
4. To further refine the movement and range, select either a routed, direct or custom line style - each will behave slightly differently (see also: [Connector Styles](#)).



6.3.2.4 Change Connector Type

To change a connector type, follow the steps below:

1. In the diagram view, right click on the connector you wish to change, to open the context menu
2. From the *Connection Detail* submenu, select *Change Type*.
3. From the drop down list, select the *Connector Type* you require.
4. Press *OK* to apply changes.



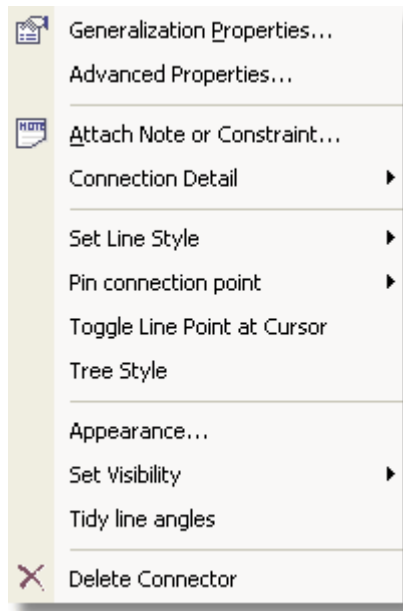
6.3.2.5 Reverse Connector

You can reverse the direction of a connection without having to delete and re-create it. This is helpful if your design changes or you add the connector wrongly to begin with.

Reverse a Connection

To reverse a connection, follow the steps below:

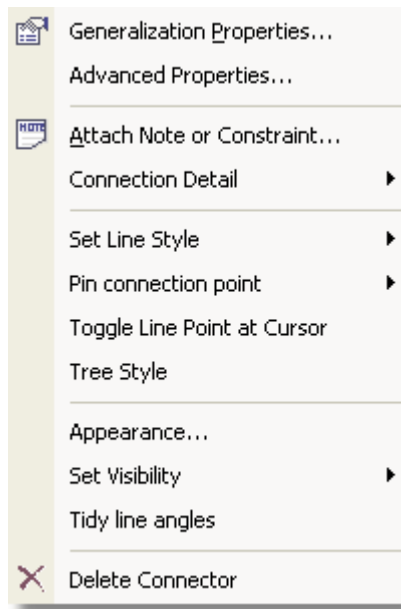
1. Right click on the incorrect connector to open the context menu.
2. From the *Connection Detail* submenu, select *Reverse Direction*.



6.3.2.6 Deleting Connections

To delete a connection, follow the steps below:

1. Right click on the connector to open the context menu.
2. Select *Delete Connector*.



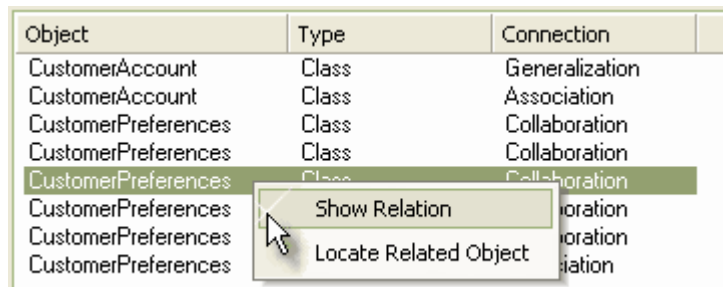
6.3.2.7 Hide/Show Connectors

Connectors/relations which appear in multiple diagrams may be selectively shown or hidden. This allows for easier to read diagrams where elements may have many connections, but not all are relevant in the context of the current diagram.

Hide or Show a Connector in the Current Diagram

To hide or show a connector in the current diagram, follow the steps below:

1. Select a diagram element in the diagram view.
2. Open the *Element Properties* dialog and select the *Links* tab
3. Right click on the connection you wish to hide or show. The context menu will allow you to *Show* hidden connections or *Hide* visible ones.

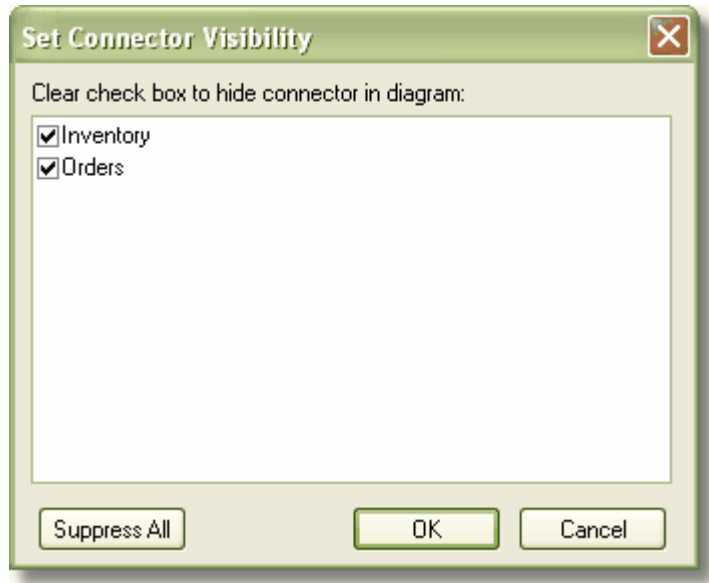


Tip: Alternatively, hide a connector by right clicking on it and selecting *Set Visibility | Hide Connector* from the context menu.

Hide or Show a Connector in Other Diagrams

To hide or show a connector in the other diagrams, follow the steps below:

1. Right click on the connector in the diagram to open the context menu.
2. From the *Set Visibility* submenu, select *Hide Connector in Other Diagrams*. This will open the *Set Connector Visibility* window.



3. If the two connected elements have been included in other diagrams, these diagrams will be listed here. All diagrams in the list which are checked will show the connector. Uncheck any diagrams in which you want the connector hidden.

Tip: If you want the connector hidden in all of the diagrams listed, press *Suppress All*.

4. When you are happy with your list, press *OK*.

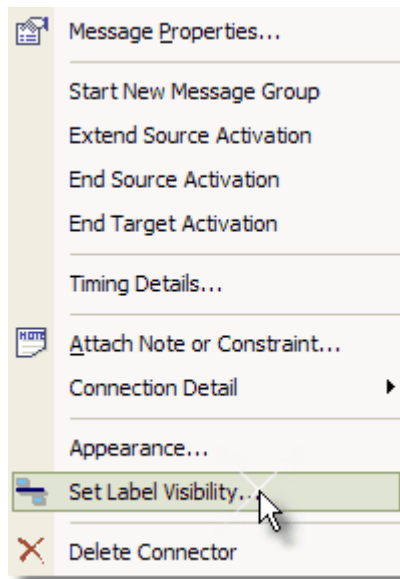
6.3.2.8 Hide/Show Labels

You can elect to hide or display one or more labels on a connection.

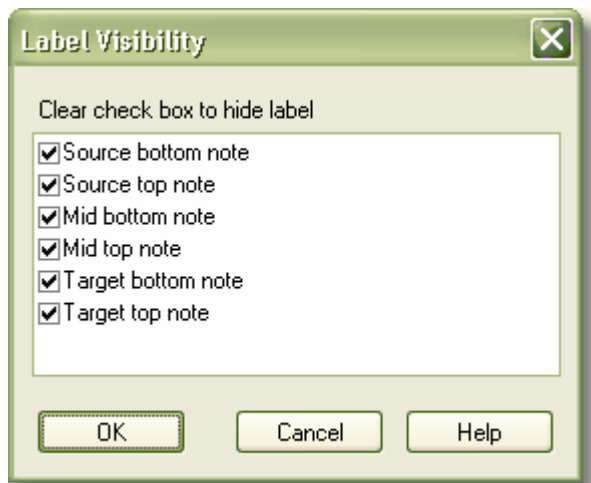
Hide/Show Labels

To hide/show labels, follow the steps below:

1. Right click on the connection to open the context menu.
2. From the *Set Visibility* submenu, select *Set Label Visibility*.



3. In the *Label Visibility* dialog that appears, check the labels to display and clear those you want to hide.
4. Press *OK* when done.



6.3.2.9 Change the Source or Target Element

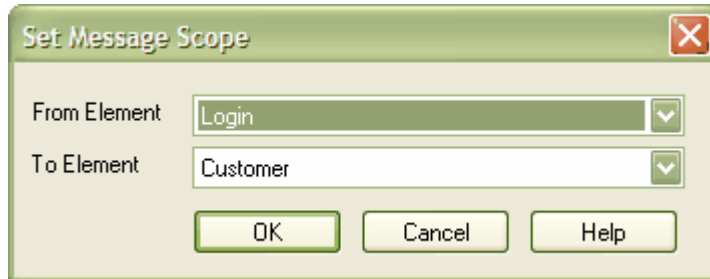
Once you have created a link between two elements, there may come a time when you want to change either the source or target. Instead of deleting and re-creating the link, EA lets you change either the source or target.

Change the Source or Target Element for a Connection

To change the source or target element of a connection, follow the steps below:

1. Right click on the connection to open the context menu.

2. From the *Connection Detail* submenu, select *Set Source and Target* to open the *Message scope* dialog.
3. Set the source and target elements using the *From Element* and *To Element* drop down lists.
4. Press *OK* to apply changes.



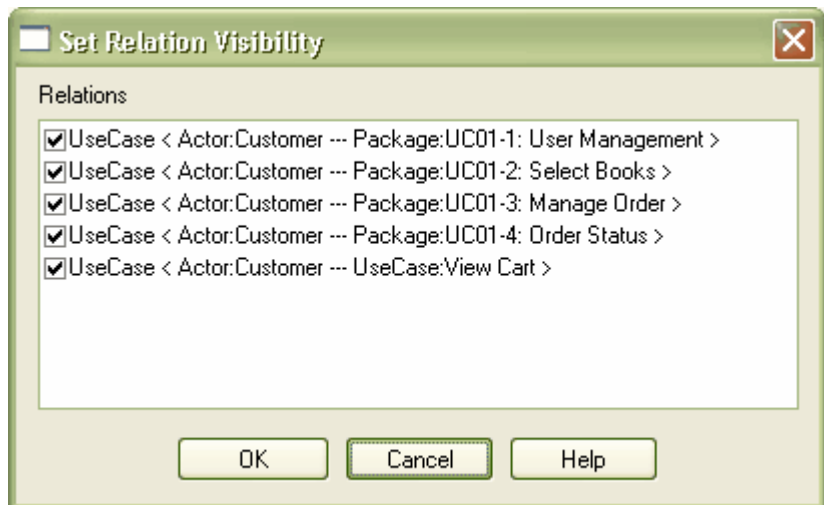
6.3.2.10 Set Relation Visibility

The visibility of individual links/connectors/relations may be changed on a per diagram basis.

Set Relation Visibility

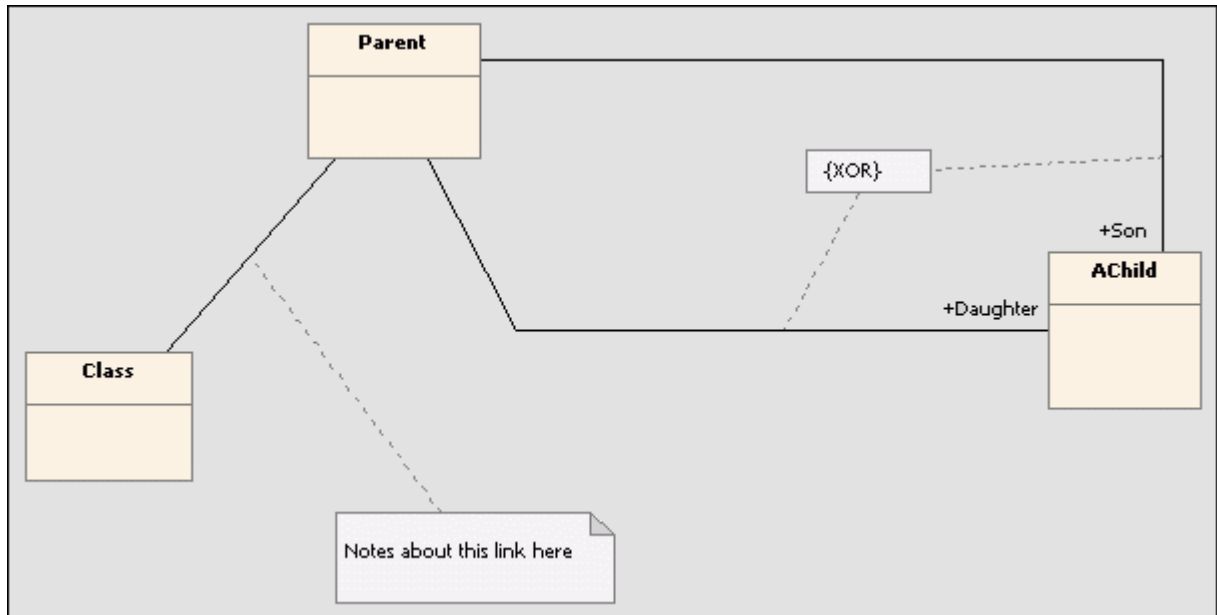
To set relation visibility, follow the steps below:

1. Open the diagram you wish to change
2. From the *Diagram* menu, select *Set Visible Relations*. Alternatively, press *Ctrl+Shift+I*.
3. Check the list items you want to show and clear those you wish to hide.
4. Press *OK* to apply changes.



6.3.2.11 Add a Note to a Link

You can link notes and constraints to graphical relationships. Notes let you provide explanation and further detail for one or more connections on a diagram, with a visible note element, as in the example below.

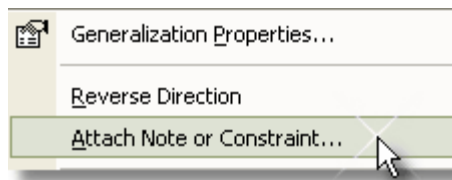


Constraints let you specify a logical or informal constraint against a set of links - for example the {XOR} constraint in the image above indicates that only one of the links in the specified set can be true at any one time (exclusivity).

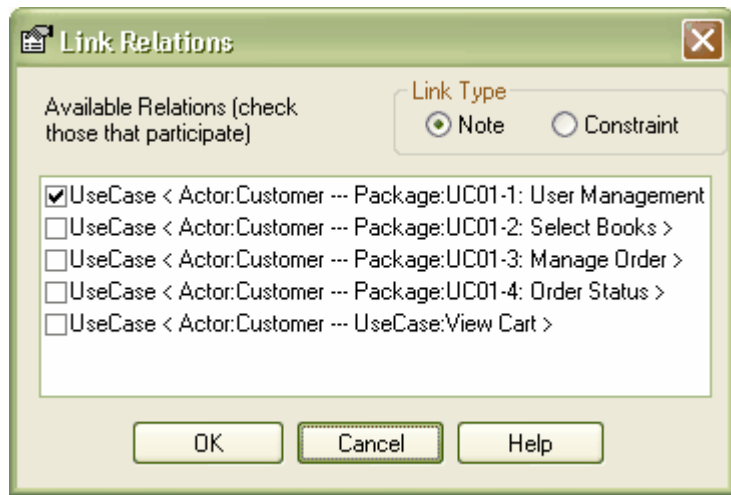
Attach a Note or Constraint to a Link

To attach a note or constraint to one or more links, do the following:

1. Right click on one of the links to attach a note to.
2. From the context menu, select *Attach Note or Constraint*.



3. In the *Link Relations* dialog, check all those links which participate in the set. In the example below, two links have been checked to participate in a logical constraint.

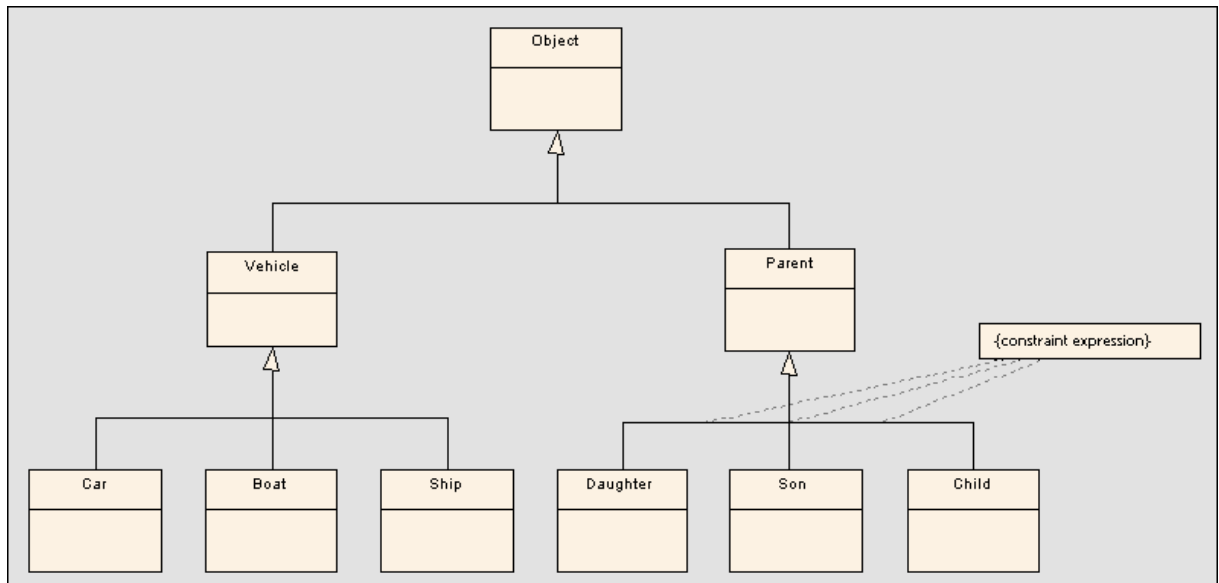


4. Press **OK** to complete the note or constraint creation.
5. You may then use the normal *Note dialog* to enter the appropriate text for the note or constraint.

Note: The constraint note is drawn slightly differently to a regular note, and has a { and } automatically added to visually indicate the constraint form.

6.3.2.12 Tree Style Hierarchy

It is possible in EA to create a tree style inheritance diagram, using a special form of the Generalization link. The example below illustrates this idea.

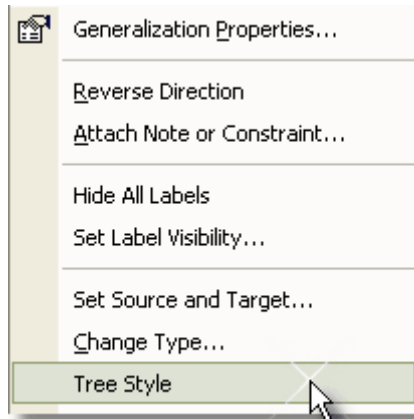


This style of diagram provides a clearer layout for inheritance hierarchies and is easy to work with.

Create a Tree Style Link

To create a tree style link, follow the steps below:

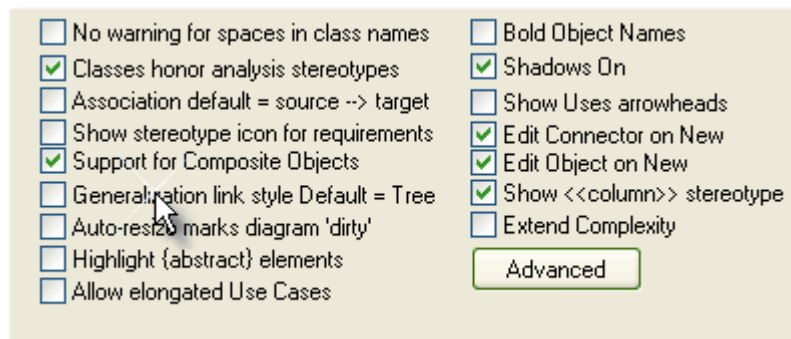
1. Create a normal Generalization between two elements, and then right click on the link to open the context menu.
2. Select the *Tree Style* option.
3. EA will automatically make the Generalization layout conform to a specific shape. By adding more Generalization links, and checking their Tree Style option, the appearance as in the diagram above is possible. You can slide the root and child classes left and right to achieve a desirable result - EA will maintain the conformity of the branch links.



Setting the Default Link Style

To set this style of link as default, follow the steps below:

1. Open the *Local Options* dialog by selecting *Options* from the *Tools* menu .
2. Select the *Objects* section.
3. Check the *Generalization link style Default = Tree* check box to make this branching style the default style for inheritance links.



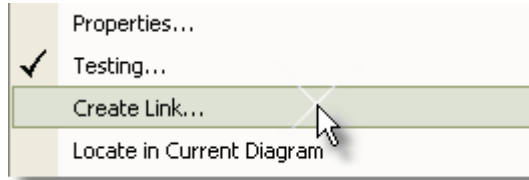
6.3.2.13 Create Link

You can create a link from one element to another directly in the Project Browser.

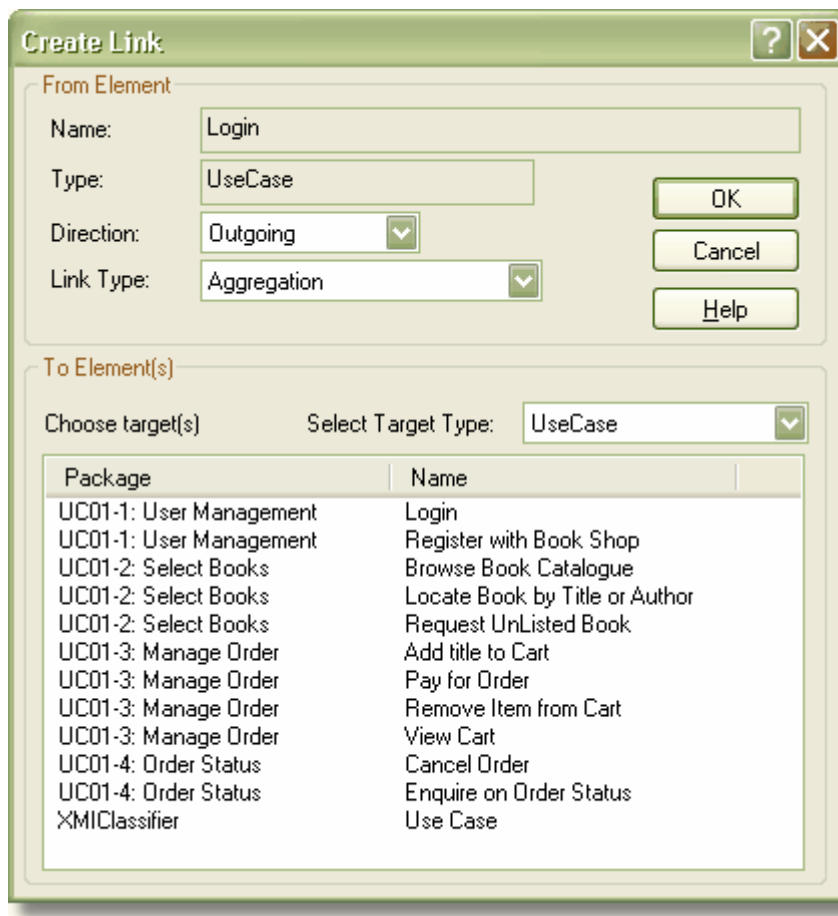
Link Elements from the Project Browser

To link elements from the Project Browser, follow the steps below:

1. In the Project Browser, right click on the element you wish to create a link for.
2. From the context menu, select *Create Link* to open the *Create Link* dialog.



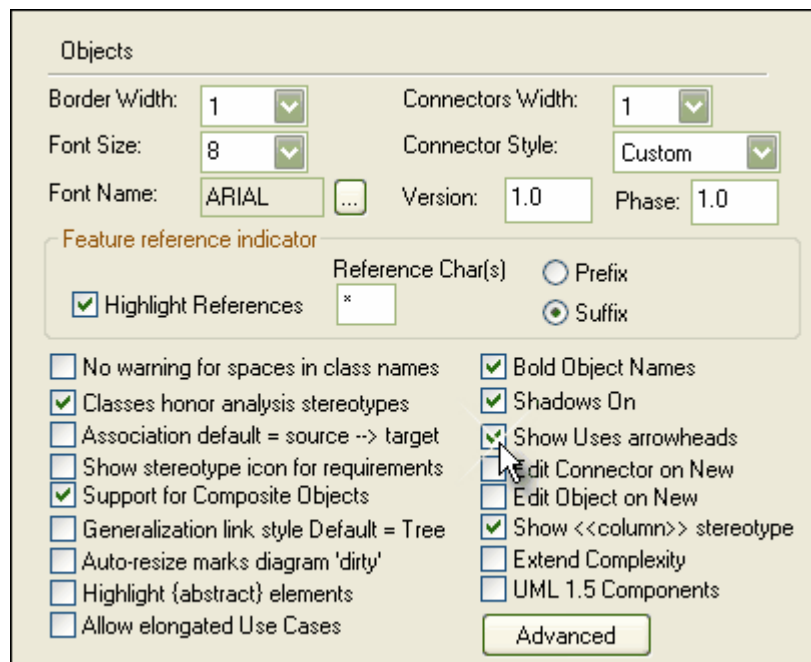
3. Select the *Direction* of the new link (Outgoing means this element is the source).
4. Select the *Link Type*.
5. Select the target in the *Choose Target* list .
6. To filter the list, use the drop down list *Select Target Type*.
7. Press *OK* to create the link.



6.3.2.14 Show Uses Arrow Head

By default EA sets the Use connector in use cases to have no arrow head, to change this behavior use the following steps.

1. Open the Tools Menu and click on the Options item.
2. In the Local Options dialog box select the Objects item.
3. Ensure that the Show Uses arrowhead item is checked.



4. When the Use Case diagram is saved the Use connectors will change to display arrowheads.

6.3.3 Association Properties

To access the *Connection Properties* dialog, double click with the mouse on a connection in a diagram. Several characteristics of connections may be changed from this dialog.

The Connection properties dialog has several tabs. The *General* tab allows you to configure the name of the connection (Link Name), the direction, the line style (routed or auto), the optional stereotype and a comment.

The screenshot shows the 'UseCase Properties' dialog box with the following details:

- Source:** Login
- Target:** Register with Book Shop
- Link Name:** (empty text box)
- Direction:** Source -> Destination
- Style:** Custom
- Stereotype:** (empty dropdown)
- Object Flow:**
- Notes:** (empty text area)

Control	Description
Link Name	An optional name for the link. The name will be displayed on the diagram if you enter one.
Direction	Direction details - from source to destination, reverse or bi-directional. Some links have arrow heads which depend on this setting. Some links are logically dependent on this (eg. inheritance)
Style	Connection style - can be direct (straight line from source to destination) or custom (user defined)
Stereotype	An optional stereotype for the link(will be displayed on the diagram if entered)
Note	An optional note about the link. The note will be displayed in documentation if desired

Tip: If you set a stereotype, this will display in a diagram and over-ride the link type in the RTF documentation.

6.3.3.1 Connection Constraints

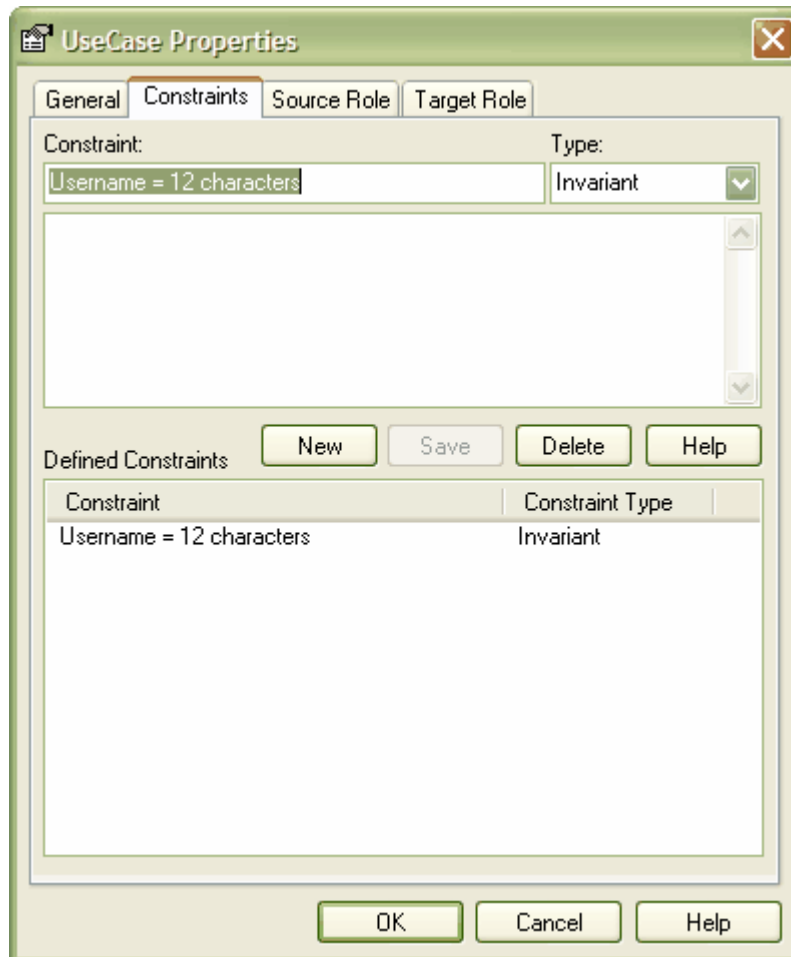
A UML connection may also have associated constraints placed on it. Constraints tell us something about the rules and conditions under which a relation operates. For example, it may be a pre-condition that the a customer is of a certain type before an association link to an Account is allowed.

Tip: Constraints about an association (connection) may be added to further refine the model. Constraints detail the business and operational rules for the model.

Set Constraints on a Link

To set constraints on a link, follow the steps below:

1. Double click on a connection to open the *Connection Properties* dialog.
2. Select the *Constraints* tab.
3. Fill in details about the constraint(s) that apply and press *Save*.



Control	Description
Constraint	Name of constraint
Type	The type of constraint (eg. pre-condition)
Notes	Notes about the link
Constraint List	A list of constraints for this list
New/Save/Delete	Add, Modify and Delete buttons

6.3.3.2 Source Role

A link may have certain properties assigned to one end, and associated with a particular role that element will play in the relationship.

You may enter details about this role to further develop your model.

Set Source Role Details

To set the source role details, follow the steps below:

1. Double click on a connection to open the *Connection Properties* dialog.
2. Select the *Source Role* tab.
3. Enter the required details and press *OK*.

The screenshot shows the 'UseCase Properties' dialog box with the 'Source Role' tab selected. The 'Login Role' is 'm_PersonList' and 'Access' is 'Protected'. The 'Role Notes' field contains 'A list of staff in order'. Under 'Containment', 'Not Specified' is selected. 'Multiplicity' is '1..*' and 'Multiplicity is Ordered' is checked. 'Aggregation' is 'none' and 'Changeable' is set to 'none'. The 'Member Type' is 'PersonList'. Buttons for 'OK', 'Cancel', and 'Help' are at the bottom.

Control	Description
Source Role	The name of the source role to be played
Access	Access level for role
Multiplicity	The role multiplicity (eg. 1..n)
Role Note	A note about the role
Containment	The nature of the containment at the source (reference, value...)
Role Constraints	Free text constraint
Aggregate	Check if role is an aggregate
Multiplicity is ordered	Check if the role is a list and the list is ordered
Qualifier	A qualifier or restriction on the role
Stereotype	A stereotype that applies to this end of the association
Type	A type that can be used when generating collection classes for multiplicity > 1

Note: Source role details are displayed at the start end of a connection. If you have drawn the link the wrong way, you can always use the Reverse Direction feature from the connection context menu.

6.3.3.3 Destination Role

A link may have certain properties assigned to one end, and associated with a particular role that element will play in the relationship.

You may enter details about this role to further develop your model.

Set Destination Role Details

To set the destination role details, follow the steps below:

1. Double click on a connection to open the *Connection Properties* dialog.
2. Select the *Destination Role* tab.
3. Details and appearance of this tab are identical to the *Source Role* tab.

Note: *Destination Role* details will be displayed at the terminating end of a connection on the diagram.

Control	Description
Destination Role	The name of the Destination role to be played
Access	Access level for role
Multiplicity	The role multiplicity (eg. 1..n)
Role Note	A note about the role
Containment	The nature of the containment at the Destination (reference, value...)
Role Constraints	Free text constraint
Aggregate	Check if role is an aggregate
Multiplicity is ordered	Check if the role is a list and the list is ordered
Qualifier	A qualifier or restriction on the role

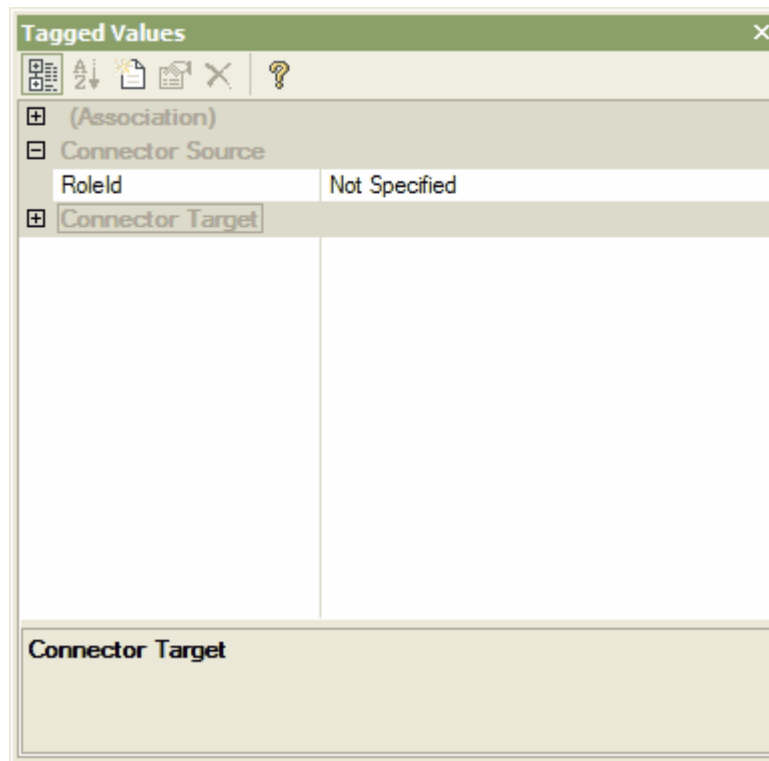
6.3.3.4 Role Tagged Values

For Association and Aggregation connection types you can set additional tagged values on the Source and/or Target Role.

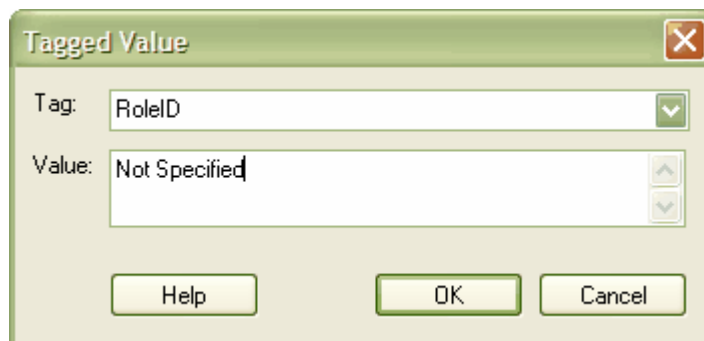
Setting Tagged Values

To set tagged values for the connection, follow the steps below:

1. Ensure that the Tagged Values window is open by pressing the *Ctrl + Shift + 6* hotkey combination, or from the *View | Other Windows | Tagged Values*.
2. Click on the connector in the diagram to display the tagged values information for the connector in the *Tagged Values* window.
3. Select *Connector Source* or *Connector Target* in the *Tagged Values* window as required.



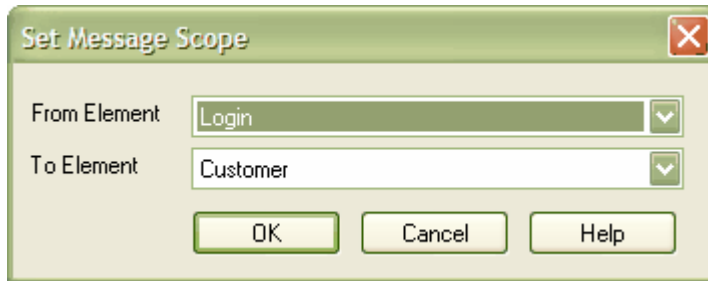
4. In the *Tagged Values* window press either the *New Tags* button or the *Ctrl + N* hotkey combination. In the *Tagged Value* dialog enter the tag name and values or select a [Predefined Tagged Value](#) type from the dropdown list.



5. Press the *OK* button to save changes.

6.3.3.5 Message Scope

A message in a sequence diagram represents a dynamic interaction from one element to another. Sometimes when you are designing your model you may need to change either the start or end point of a message as the responsibilities of elements change during design. For this reason, EA lets you change the message scope by setting a new start or end element.



Change Message Scope

To change message scope, follow the steps below:

1. Select the message in the sequence diagram.
2. Right click on the message to open the context menu.
3. Select *Set Message Scope*.
4. In the pop up dialog, select the required *From* and *To* elements from the drop down lists.
5. Press *OK* to save changes.

The message has now been re-routed to meet your changed requirements.

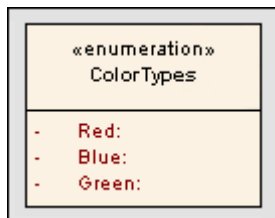
6.4 UML Stereotypes

UML supports a large number of stereotypes, which are an inbuilt mechanism for logically extending or altering the meaning, display and syntax of a model element. Different model elements have different stereotypes associated with them.

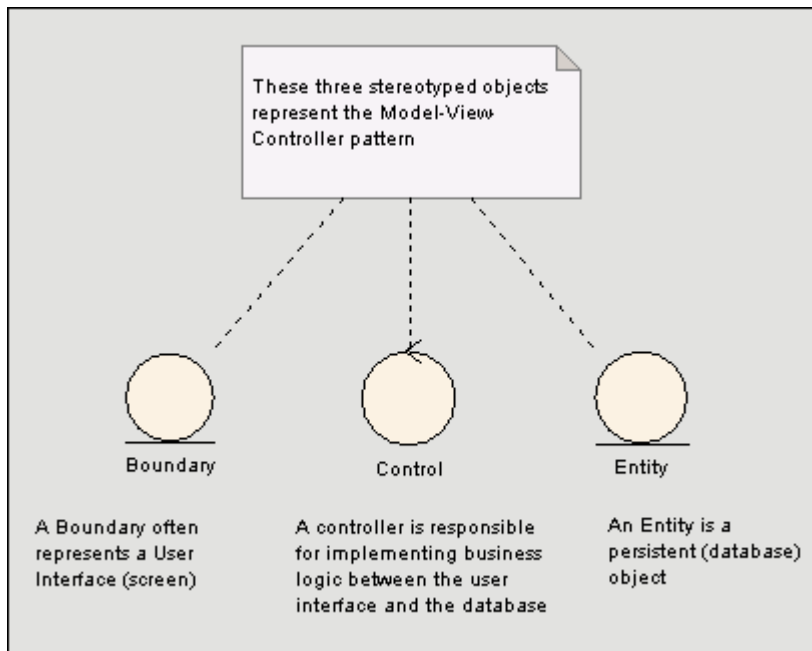
The OMG UML specification states:

"A stereotype is, in effect, a new class of metamodel element that is introduced at modeling time. It represents a subclass of an existing metamodel element with the same form (attributes and relationships) but with a different intent. Generally a stereotype represents a usage distinction. A stereotyped element may have additional constraints on it from the base metamodel class. It may also have required tagged values that add information needed by elements with the stereotype. It is expected that code generators and other tools will treat stereotyped elements specially. Stereotypes represent one of the built-in extensibility mechanisms of UML."

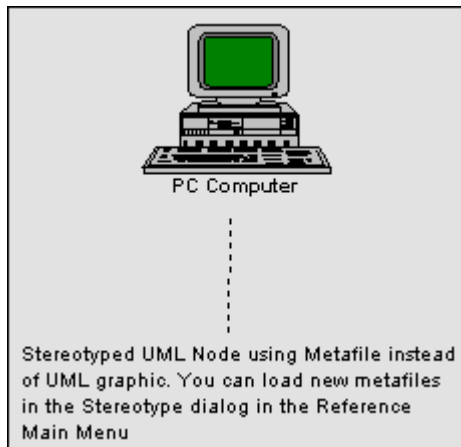
The stereotype is generally displayed as in the example below (where <<enumeration>> is the stereotype).



In some cases, the stereotype will cause the element to be drawn differently - as below:



A metafile may be associated with the applied stereotype, as in the examples below:



Add Your Own Stereotypes

To add your own custom stereotypes, follow the steps below:

1. From the *Configuration | UML* submenu, select *Stereotypes*.
2. Enter a *Stereotype* name.
3. Select a *Base Class* from the drop down list.
4. If you wish to associate a *Metafile* with this stereotype, press the *Browse [...]* button and locate the required .emf or .wmf file.
5. Enter optional *Notes* and select *Default Colors* for this stereotype
6. Press *Save* to save stereotype



6.4.1 Standard Element Stereotypes

Below is a list of standard element stereotypes:

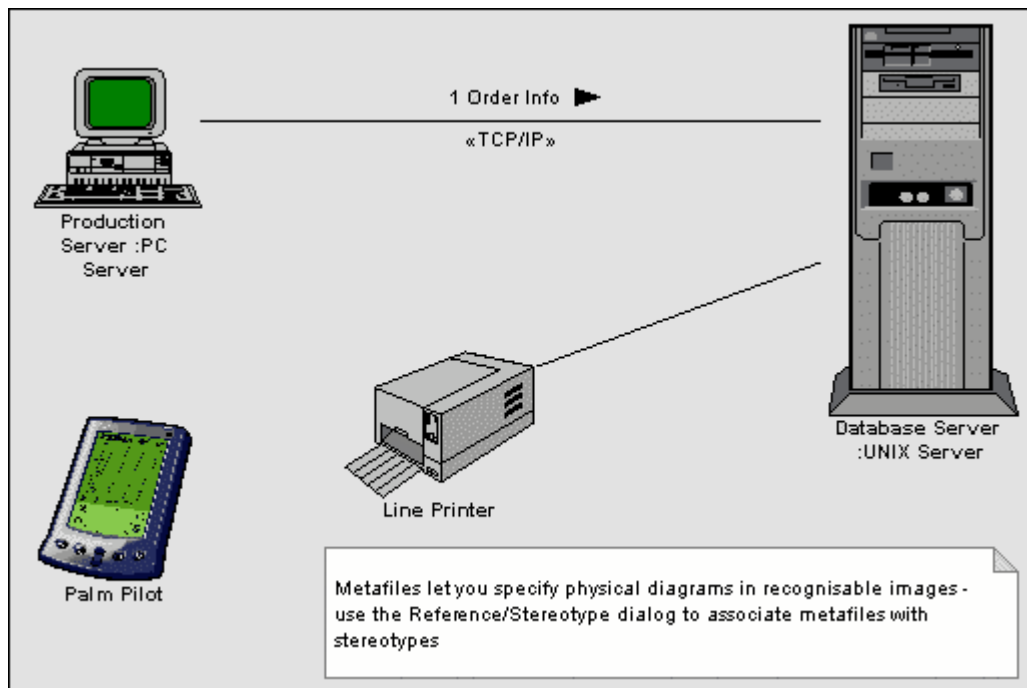
Stereotype	Base Class	Type
«access»	Permission	Stereotype
«appliedProfile»	Package	Stereotype
association	Association	Constraint
«association»	AssociationEnd	Stereotype
«auxiliary»	Class	Stereotype
«become»	Flow	Stereotype
«call»	Usage	Stereotype
complete	Generalization	Constraint
«copy»	Flow	Stereotype
«create»	BehavioralFeature	Stereotype
«create»	CallEvent	Stereotype
«create»	Usage	Stereotype
«derive»	Abstraction	Stereotype
derived	ModelElement	Tag
«destroy»	BehavioralFeature	Stereotype
«destroy»	CallEvent	Stereotype
destroyed	Association	Constraint
destroyed	Association	Constraint
disjoint	Generalization	Constraint
«document»	Abstraction	Stereotype
documentation	Element	Tag
«executable»	Abstraction	Stereotype
«facade»	Package	Stereotype
«file»	Abstraction	Stereotype
«focus»	Class	Stereotype
«framework»	Package	Stereotype
«friend»	Permission	Stereotype
global	Association	Constraint
«global»	AssociationEnd	Stereotype
«implementation»	Class	Stereotype
«implementation»	Generalization	Stereotype
implicit	Association	Stereotype
«import»	Permission	Stereotype
incomplete	Generalization	Constraint
«instantiate»	Usage	Stereotype
«invariant»	Constraint	Stereotype

«library»	Abstraction	Stereotype
local	Association	Constraint
«local»	AssociationEnd	Stereotype
«metaclass»	Class	Stereotype
«metamodel»	Package	Stereotype
«modelLibrary»	Package	Stereotype
«modelLibrary»	Package	Stereotype
new	Association	Constraint
new	Association	Constraint
overlapping	Generalization	Constraint
parameter	Association	Constraint
«parameter»	AssociationEnd	Stereotype
persistence	Association	Tag
persistence	Attribute	Tag
persistence	Classifier	Tag
persistent	Association	Tag
«postcondition»	Constraint	Stereotype
«powertype»	Class	Stereotype
«precondition»	Constraint	Stereotype
«process»	Classifier	Stereotype
«profile»	Package	Stereotype
«realize»	Abstraction	Stereotype
«refine»	Abstraction	Stereotype
«requirement»	Comment	Stereotype
«responsibility»	Comment	Stereotype
self	Association	Constraint
«self»	AssociationEnd	Stereotype
semantics	Classifier	Tag
semantics	Operation	Tag
«send»	Usage	Stereotype
«signalflow»	ObjectFlowState	Stereotype
«source»	Abstraction	Stereotype
«stateInvariant»	Constraint	Stereotype
«stub»	Package	Stereotype
«systemModel»	Package	Stereotype
«table»	Abstraction	Stereotype
«thread»	Classifier	Stereotype
«topLevel»	Package	Stereotype

«trace»	Abstraction	Stereotype
transient	Association	Constraint
transient	Association	Constraint
«type»	Class	Stereotype
usage	Association	Tag
«utility»	Classifier	Stereotype
xor	Association	Constraint

6.4.2 Stereotypes with Alternate Images

You can alter the appearance of elements quite dramatically using stereotypes. If the stereotype has an associated metafile specified, then when the stereotype is applied to a class or other element which supports alternate graphical format, then EA will draw the alternate image instead of the standard one.



6.5 MDG Technologies

The Model Driven Generator (MDG) Technologies allow for a logical collection of resources pertaining to a specific technology to be bundled into one centralized location in Enterprise Architect. With MDG Technologies the user has the option of granular importation of UML Profiles, UML Patterns, Code templates and Language types to be contained in a single, easy to access area contained in the Enterprise Architect Resource View. To get you started with MDG Technologies Sparx Systems offer MDG Technologies for download from http://www.sparxsystems.com.au/mdg_technologies.htm.

Typical Tasks

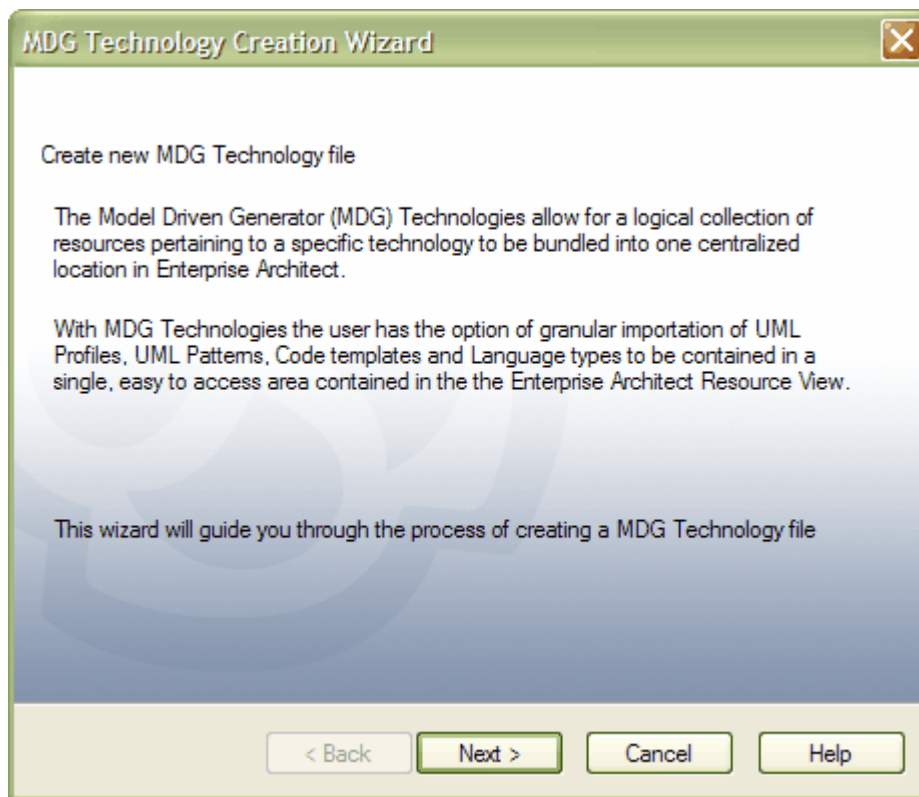
Typical tasks you might want to perform with MDG Technologies include:

- [Creating MDG Technologies](#)
- [Working with MDG Technologies](#)
- [Adding MDG Technologies to the UML Toolbox](#)
- [Import MDG Technologies](#)

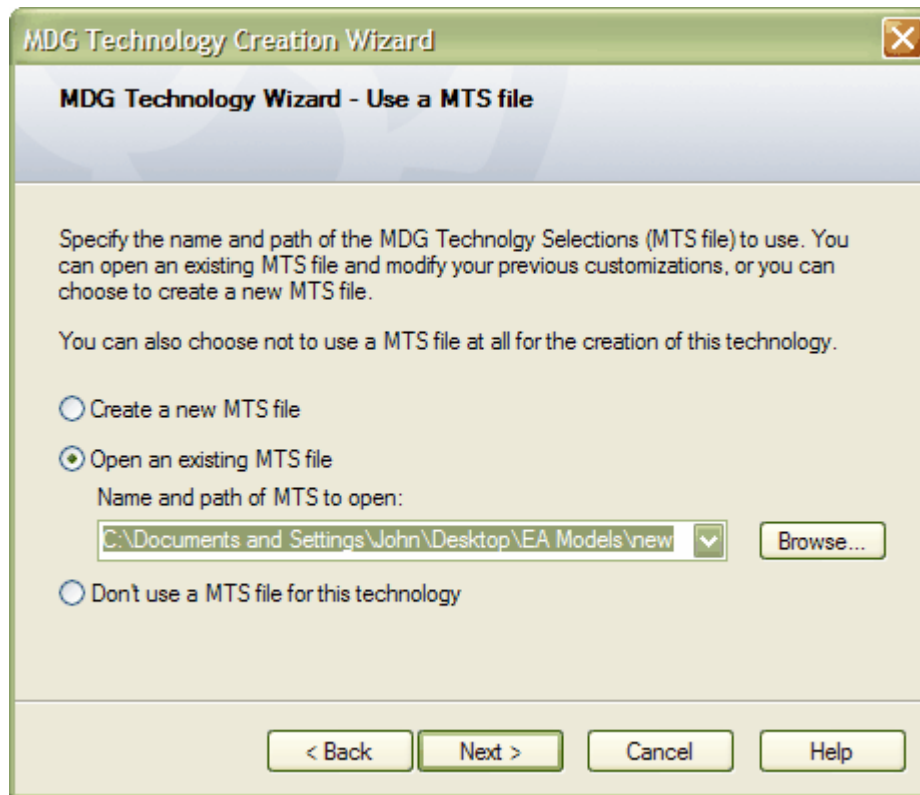
6.5.1 Creating MDG Technologies

MDG Technology files may be created using the MDG Technology wizard. By using the Technology wizard it is possible to create and MDG Technology files which may include [UML Profiles](#), Code Modules, [Patterns](#), Images and [Tagged Value Types](#). To create an MDG Technology file use the following instructions:

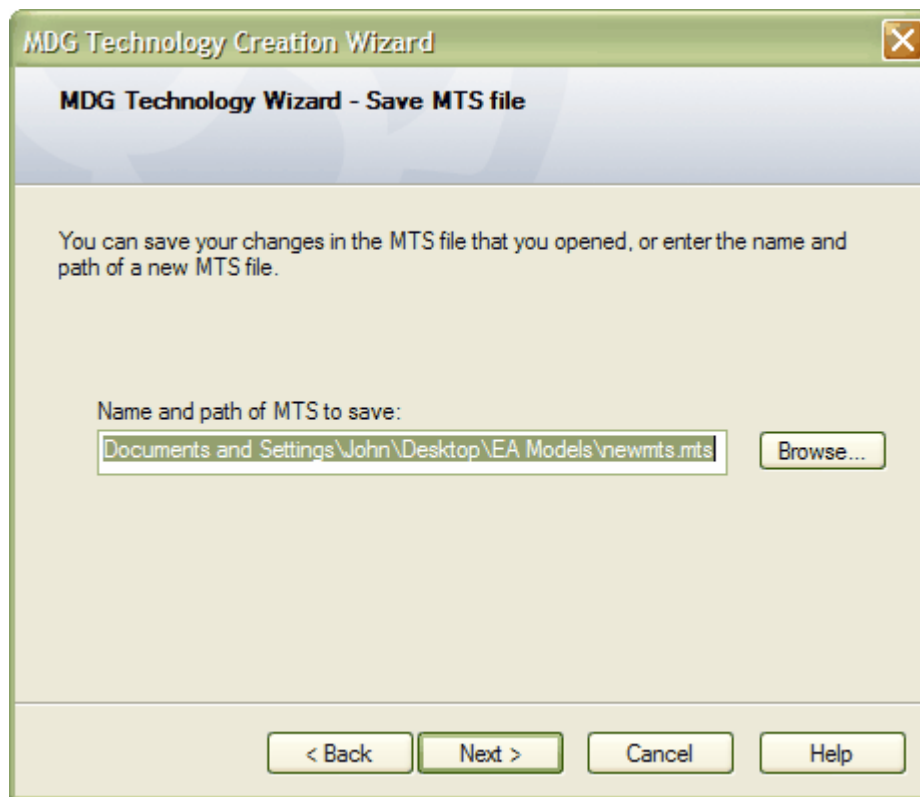
1. To start the wizard go to the *Project* menu and select *Generate MDG Technology File*.
2. This will open the *MDG Technology Creation Wizard*, press the *Next* button to proceed.



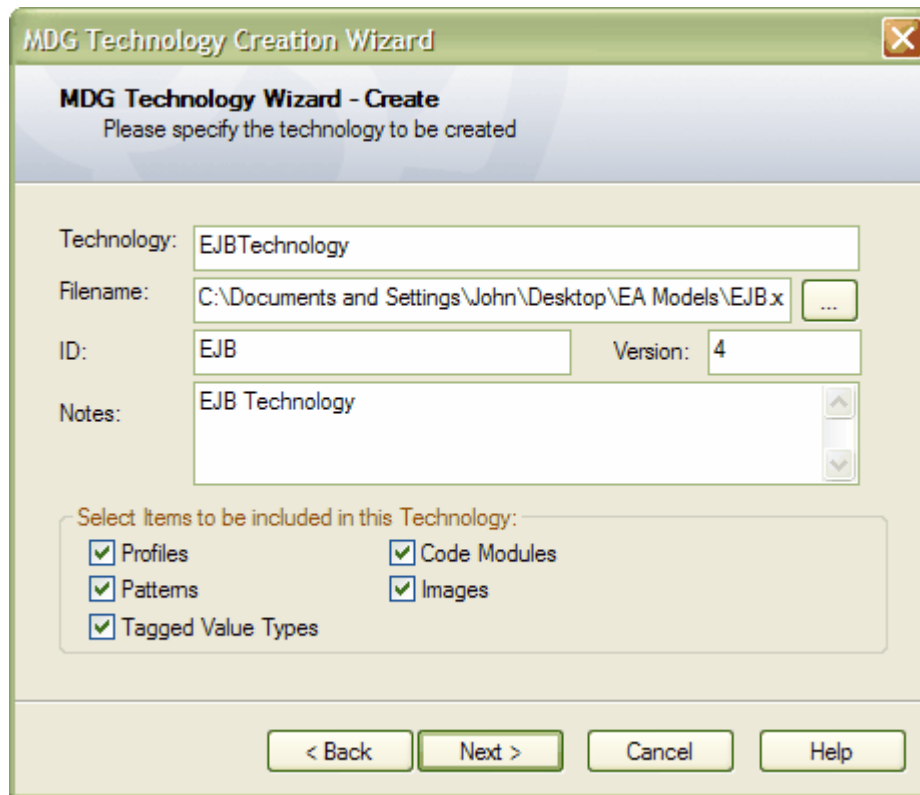
3. The wizard will then allow the user to specify the creation of a new MDG Technology Selection (MTS) file, or use an existing MTS file or the option of not using an MTS file. An MTS file stores the selected options that a user defines during the creation of an MDG technology file and if created will allow the user to perform modifications to the MTS file, allow the user to add or remove specific items to the MDG technology file. Select the appropriate option and then press the *Next* button to continue the Wizard.



4. If a MTS file is selected the user will be prompted to save the changes in the existing MTS file or given the opportunity to save the changes into a new MTS file, this allows the user to preserve the existing MTS file and create a modification based on the existing MTS file. Select the appropriate option and press the *Next* button to proceed with the Wizard.



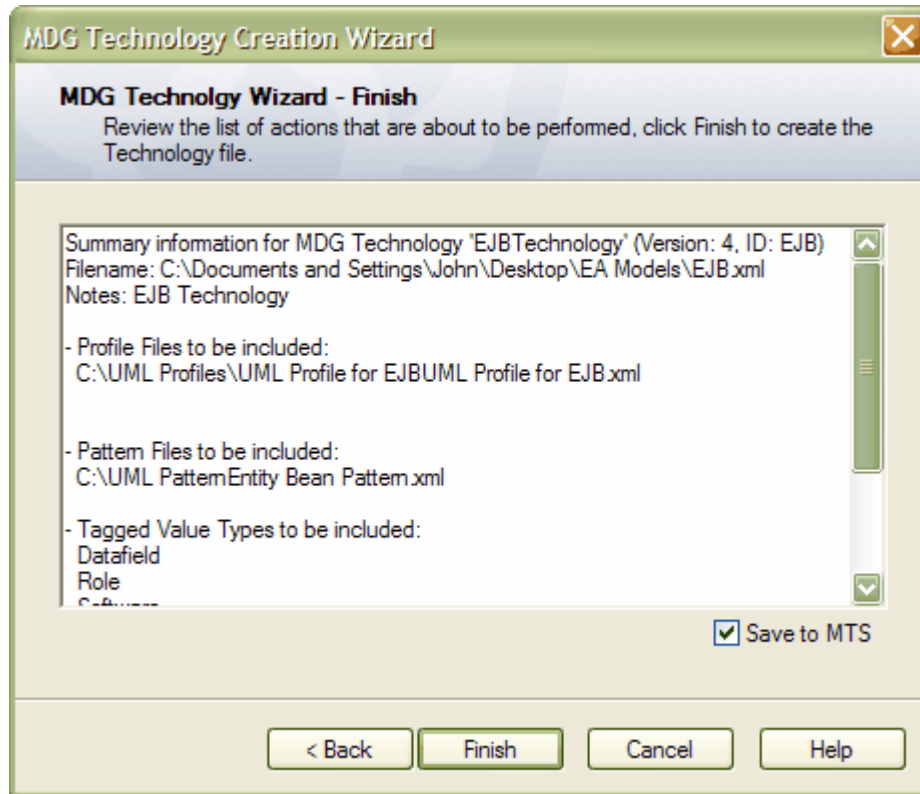
5. From the Create section of the MDG Technology Wizard the options detailed in the table below may be selected by the user, choose the appropriate options and then press the *Next* button to continue.



Field	Description
Technology	The Technology field is used to define the name of the Technology.
Filename	The Filename is the path to the MDG Technology file (the file extension for the MDG Technology File is .xml).
ID	The ID fields is a reference to the MDG Technology file, this field is restricted to a length of 12 characters.
Version	Specify the version of the MDG Technology file.
Notes	Add an explanation regarding the functionality of the MDG Technology.
Select Items to be included in this Technology	This section of the Wizard gives the user control over the items to be included in the MDG Technology, ensure the checkbox is selected next to the items that are to be included in the Technology file.

6. The items selected in the Select Items to be included in this Technology will run the specific dialogs to enable the selection of the specific items that are to be included in the MDG Technology. The method used for selection of specific items are found in the following topics:
- [Profiles](#)
 - [Patterns](#)
 - [Tagged Values Types](#)
 - [Code Modules](#)
 - [Images](#)

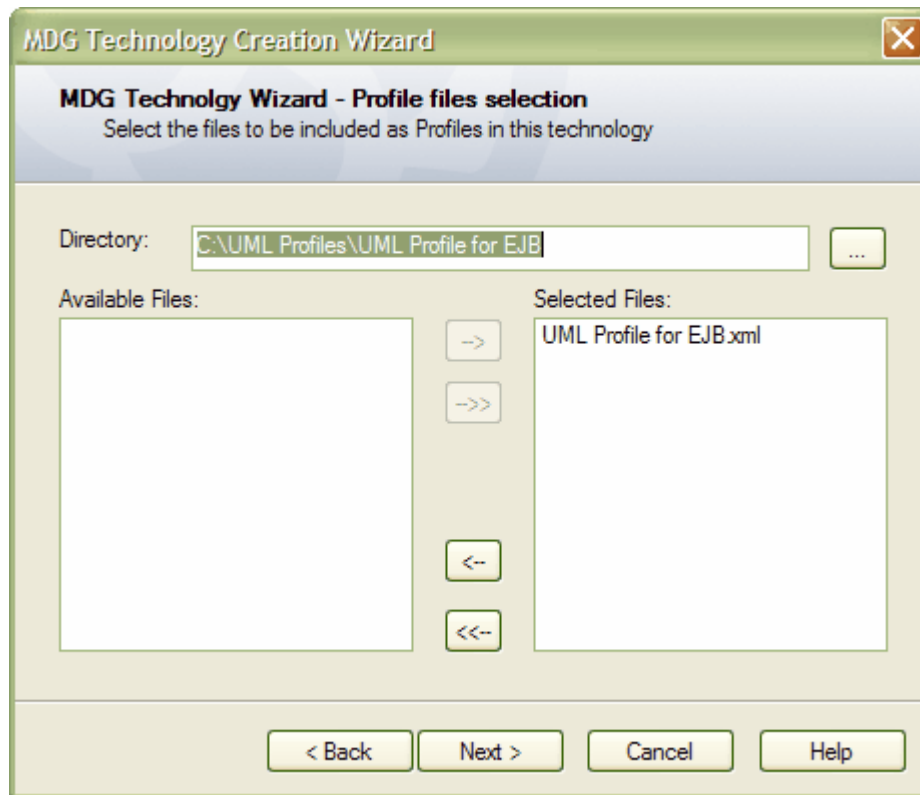
7. The final Wizard screen will then provide information about the items that are included in the MDG Technology file, if the user is satisfied with the selection of items press the *Finish* button. To update the MTS file ensure the Save to MTS checkbox is selected. To import the Technology file into an EA model see the [Import MDG Technology](#) topic.



6.5.1.1 Adding Profile in MDG Technology Wizard

During the process of creating an MDG Technology file the user may select the option of including UML 2 compliant profiles. To use the Profiles section of the wizard use the following instructions:

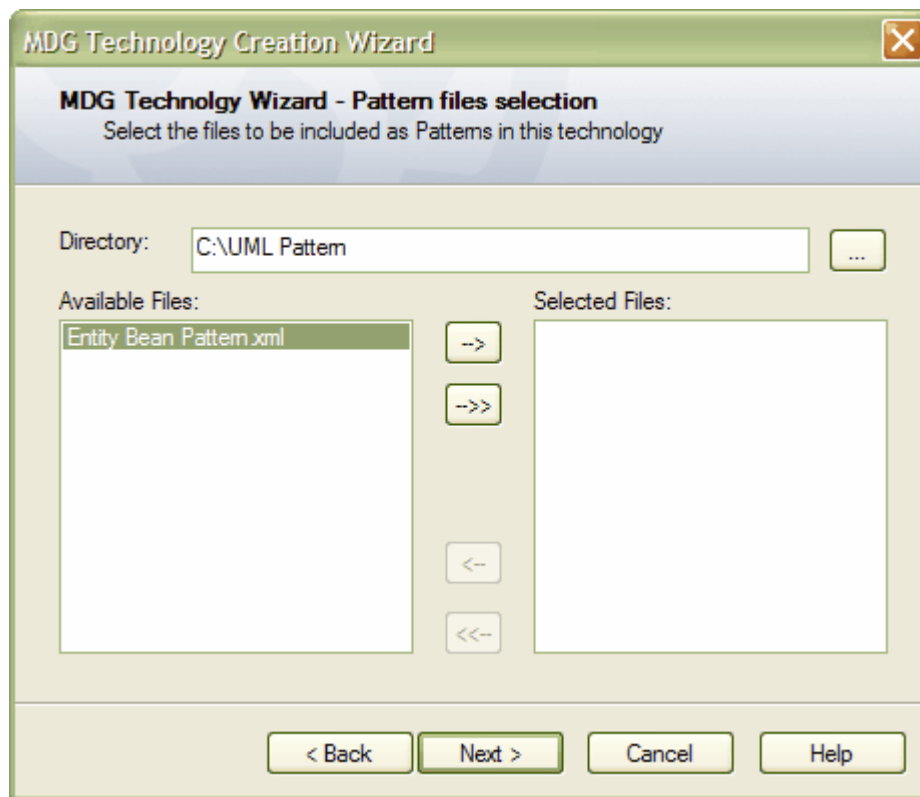
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Profiles* item selected.
2. The *Profile files selection* wizard dialog will then allow the user to navigate to the directory containing the Profile/s. Once the profile/s have been located select the desired profile/s by highlighting them in the *Available Files* section and pressing the --> button to select the Profile individually or the -->> button to select all of the available Profiles. After the appropriate selections have been made press the *Next* button to proceed.



6.5.1.2 Adding Pattern in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including Patterns. To use the Patterns section of the wizard use the following instructions:

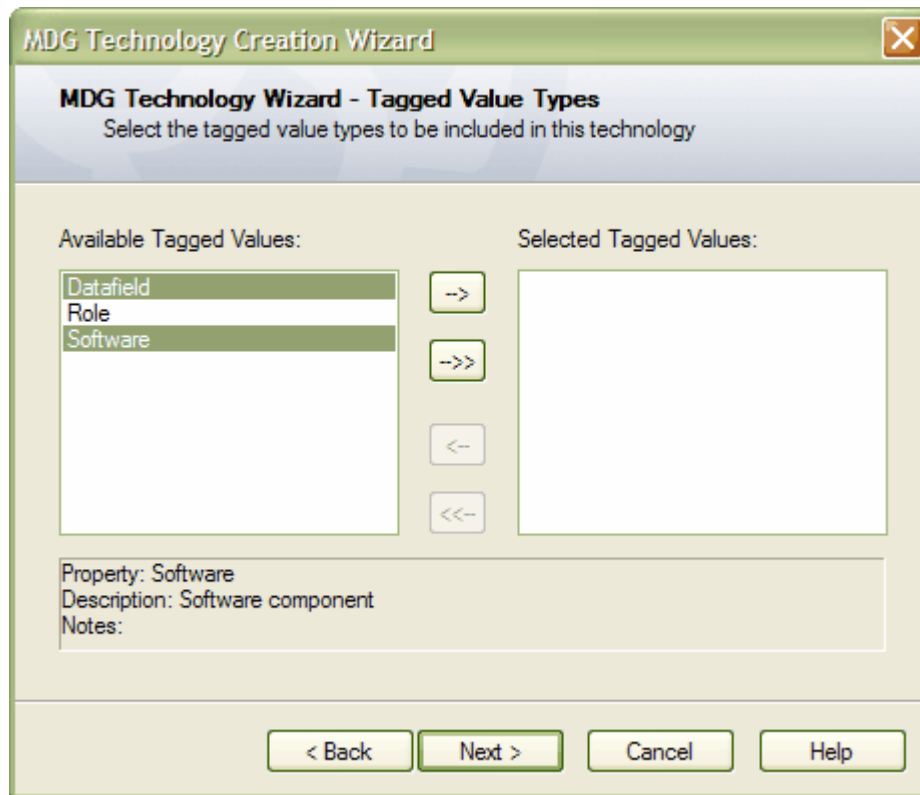
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Pattern* item selected.
2. The *Pattern files selection* wizard dialog will then allow the user to navigate to a the directory containing the Pattern/s. Once the Pattern/s have been located select the desired Pattern highlighting them in the *Available Files* section and pressing the *-->* button to select the Pattern individually or the *-->>* button to select all of the available Patterns. After the appropriate selections have been made press the *Next* button to proceed.



6.5.1.3 Adding Tagged Values in MDG Technology Wizard

During the process of creating the an MDG Technology file the user may select the option of including Tagged Value Types. To use the Tagged Value Types section of the wizard use the following instructions:

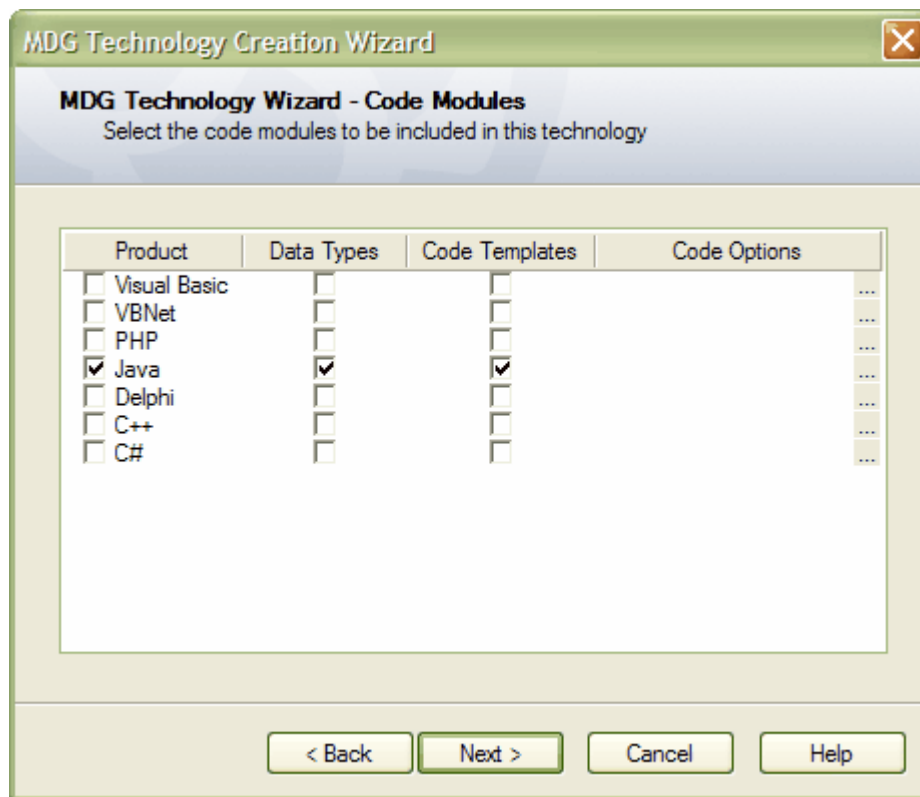
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Tagged Values Type* item selected.
2. The *Tagged Value Type* wizard dialog will then allow the user select the Tagged Values Types that have been defined in the model. Select the appropriate Tagged Value Types by pressing the --> button to select the Tagged Values Type individually or the -->> button to select all of the available Tagged Values Type. After the appropriate selections have been made press the *Next* button to proceed.



6.5.1.4 Adding Code Modules in MDG Technology Wizard

During the process of creating an MDG Technology file the user may select the option of including Code Modules. To use the Code Modules section of the wizard use the following instructions:

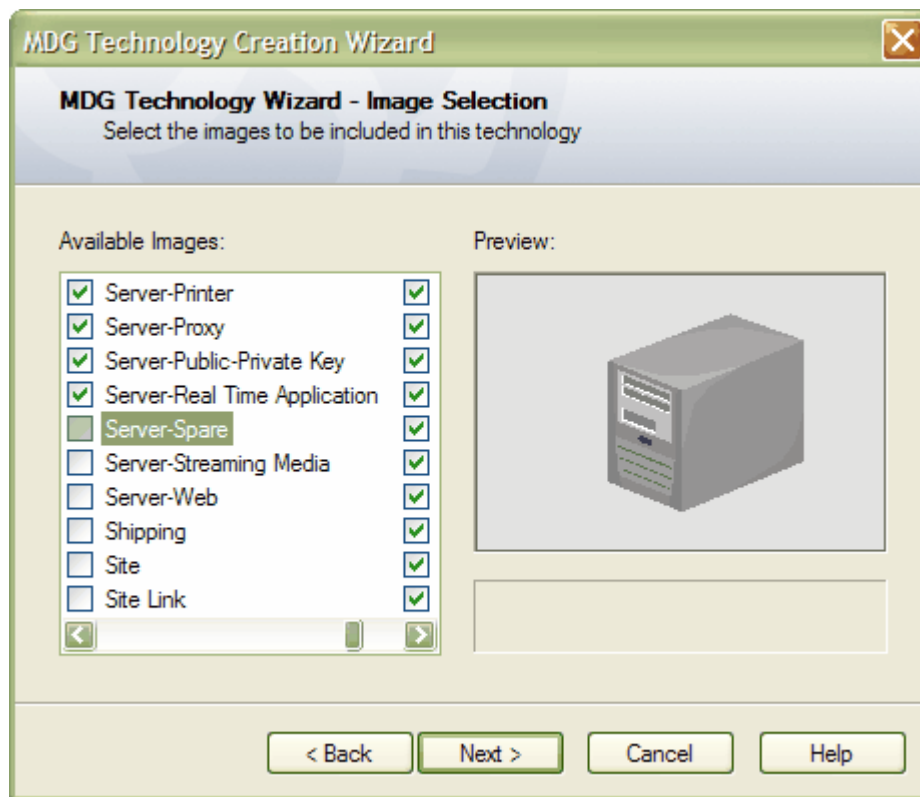
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Code Modules* item selected.
2. The *Code Modules* wizard dialog will then allow the user select the from the available Code Models that are present in the model. Select the appropriate Code Module/s by checking the required Product, Data Types, Code Templates checkboxes. To select the Code Options for each product press the ... button to select the appropriate code options. After the selections have been made press the *Next* button to proceed.



6.5.1.5 Adding Images in MDG Technology Wizard

During the process of creating an MDG Technology file the user may select the option of including the images that have been imported into the model. To use the Image Selection section of the wizard use the following instructions:

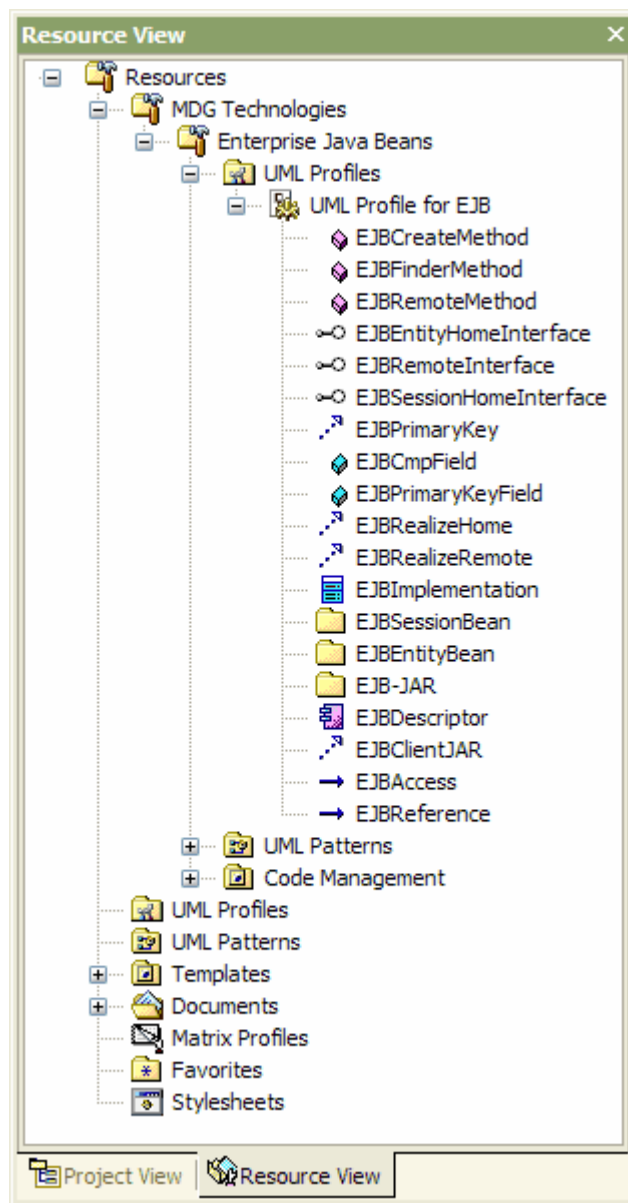
1. Use the steps detailed in the [Creating MDG Technologies](#) topic up until [Step 5](#), ensure that in the *Select Items to be included in this Technology* section has the *Images* item selected.
2. The *Image Selection* wizard dialog will then allow the user select the available model images that are present in the current model. Select the appropriate images by selecting the checkbox next to the image name. After the selections have been made press the *Next* button to proceed.



6.5.2 Working with MDG Technologies

The Resource Window

The Resource window is the second tab of the docked Project Browser window. It contains a tree structure with entries for items such as MDG Technologies, Documents, Stylesheets, Matrix profiles and UML Profiles. The MDG Technologies node initially contains no entries - to activate a MDG Technology you must import them into EA from supplied XML files. Once they have been imported, you can open the MDG Technology and work with its elements.



MDG Technologies can bundle up the functionality provided by UML Profiles, UML Patterns, Model Types and Code Templates

Profiles contained in the MDG Technologies can be applied to:

- Elements such as classes and interfaces can be dragged directly from the resource window to the current diagram.
- Attributes can be dragged over a host element (eg. Class) - they will automatically be added to the element feature list.
- Operations are like Attributes - drag over a host element to add the operation.
- Links such as Association, Generalization, and Dependency are added by selecting them in the browser, then clicking the on start element in a diagram and dragging to the end element (in the same manner as adding normal links). The link will be added with the new stereotype and tagged value information.
- Association Ends can be added by dragging the link end element over the end of an Association in the diagram.

Patterns contained in the MDG Technologies can be used to:

- Enable reuse in a model.
- Build in robustness.

Code Templates can be used to:

- Specify the transformation from UML elements into various parts of a given programming language.

Model Types can be used to:

- define the data types for the model.

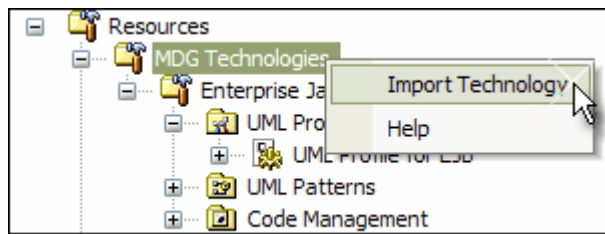
6.5.3 Import MDG Technologies

To import a MDG Technology you will need a suitable MDG Technology XML file. If the MDG Technology includes references to any Metafiles, they should be in the same directory as the XML MDG Technology.

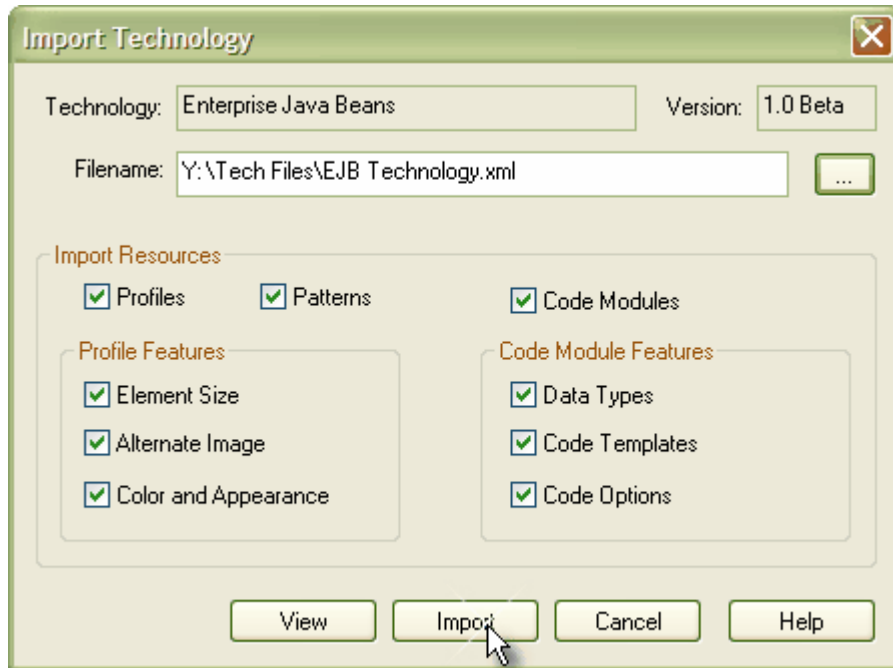
Import a MDG Technology

To import a MDG Technology, follow the steps below:

1. Right click on the MDG Technologies tree node and select *Import Technology* from the context menu in the resource view - as in the example below.



2. The *Import Technology* dialog will open.



Note: Options that are greyed out (such as the Model Types in the image above) indicate that no

examples of that type exist in the MDG Technology XML file.

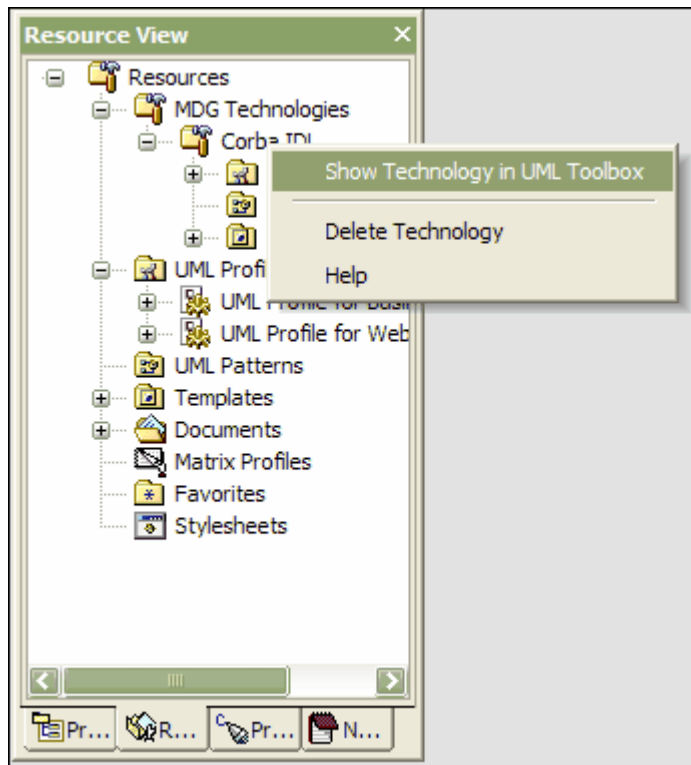
3. Locate the MDG Technology file to import using the *Browse [...]* button.
4. Set the required import options for all of the resources defined in the MDG Technology - you can select to import:
 - Element Size yes/no - check this to import the element size attributes.
 - Color and Appearance yes/no - check this to import the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image yes/no - check this to import the metafile image.
 - Patterns yes/no - check this to import patterns if they exist.
 - Profiles yes/no - check this to import profiles if they exist.
 - Code Modules yes/no - check this to import the various languages that are associated with the technology if they exist.
 - Data Types yes/no - check this import the data types.
 - Code Templates yes/no - check this to import the code templates if they exist.
 - Code Options yes/no - check this to import the options which include items such as default file extensions and default file paths.
5. Press *Import*.

If the MDG Technology already exists, EA will offer to overwrite the existing version and import the new one - or cancel. Once the import is complete, the MDG Technology is ready to use.

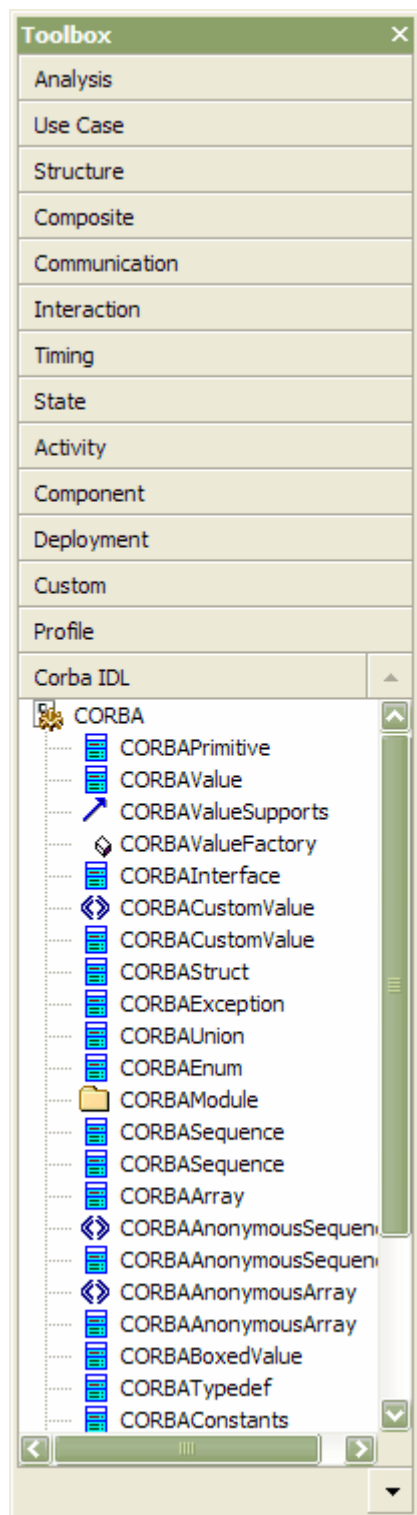
6.5.4 Add MDG Technologies to UML Toolbox

MDG technologies and UML profiles can be added to the UML tool box to enable the quick use of the items contained in the profile or technology file. To add a technology to the UML toolbox use the following instructions:

1. Locate the MDG Technology file in the Resource View.
2. Right click on the technology file in the resource window to bring up its context menu and select the *Show Technology in UML Toolbox* option.



3. This will add the Technology to the UML Toolbox, from here the user can drag the items contained in the Technology file directly into a diagram.



6.6 UML Profiles

What are UML Profiles?

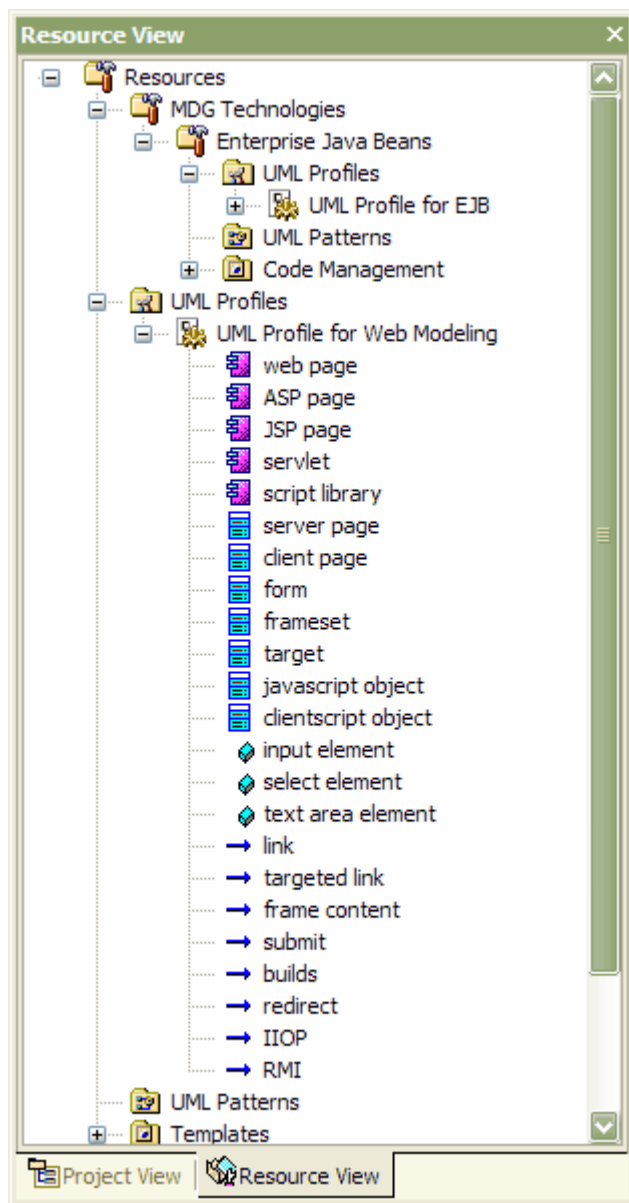
UML Profiles provide a means for extending the UML Language, which allows for building UML models in

particular domains. They are based on additional Stereotypes and Tagged values that are applied to Elements, Attributes, Methods, Links, Link Ends, etc. A profile is a collection of such extensions that together describe some particular modeling problem and facilitate modeling constructs in that domain. For example the UML Profile for XML as defined by David Carlson in the book "Modeling XML Applications with XML" pp. 310, describes a set of extensions to basic UML model elements to enable accurate modeling of XSD Schemas.

Enterprise Architect has a generic UML Profile mechanism for loading and working with different Profiles. UML Profiles for Enterprise Architect are specified in XML files, with a specific format - see the examples in this section. These XML files can be imported into EA in the Resource page of the Project Browser. Once imported, you can drag and drop Profile elements onto the current diagram. EA will attach the stereotype, tagged values and default values, notes and even metafile if one is specified, to the new element. You can also drag and drop attributes and operations onto existing classes and have them immediately added with the specified stereotype, values etc.

Profiles in the Resource View

The Resource window is the second tab of the docked Project Browser window. It contains a tree structure with entries for items such as MDG Technologies, Documents, Stylesheets, Matrix profiles and UML Profiles. The UML Profiles node initially contains no entries - to be able to use profiles you must import them into EA from supplied XML files.



Items in the Profile represent stereotypes. These can be applied to UML elements in the following ways:

- Stereotypes that apply to elements such as **classes** and **interfaces** can be dragged directly from the resource window to the current diagram, automatically creating a stereotyped element.
- Stereotypes that apply to **attributes** can be drag-and-dropped onto a host element (eg. class) - they will automatically be added to the element's feature list.
- Stereotypes that apply to **operations** are like those that apply to attributes - drag-and-drop onto a host element to add the stereotyped operation.
- Stereotypes that apply to **connectors** such as **associations**, **generalizations** and **dependencies** are added by selecting them in the browser, then clicking on the start element in a diagram and dragging to the end element (in the same manner as adding normal links). The stereotyped link will be added.
- Stereotypes that apply to **association ends** can be added by dragging the link end element over the end of an association in the diagram.

To get you started, some profiles are supplied on the Sparx Systems website at www.sparxsystems.com.au/uml_profiles.htm. You can download these and import them into EA. Over time we

will expand the range of Profiles, the content of each profile and the degree of customization possible in each profile. Remember, you can always create your own profiles to describe modeling scenarios specific to your development environment.

See also: [Creating Profiles](#), [Using Profiles](#), [Profile References](#)

6.6.1 Creating Profiles

This section describes how to create profiles and profile items. These creation tasks include creating the profile stereotypes, defining the metaclasses they apply to, as well as defining tagged values and constraints. This section also describes how to export a profile for use in UML modeling.

Perform the following steps to create a UML Profile:

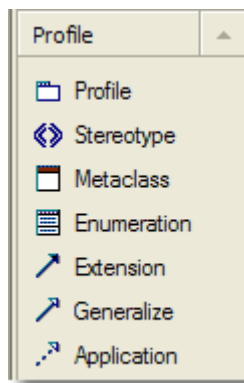
1. [Create a Profile](#)
2. [Add Stereotypes and Metaclasses](#)
3. [Define Tagged Values for Stereotypes](#)
4. [Define Constraints for Stereotypes](#)
5. [Add Enumerations](#)
6. [Export the Profile](#)

6.6.1.1 Create a UML Profile

Enterprise Architect allows you to create profiles in your model and use them directly in your project.

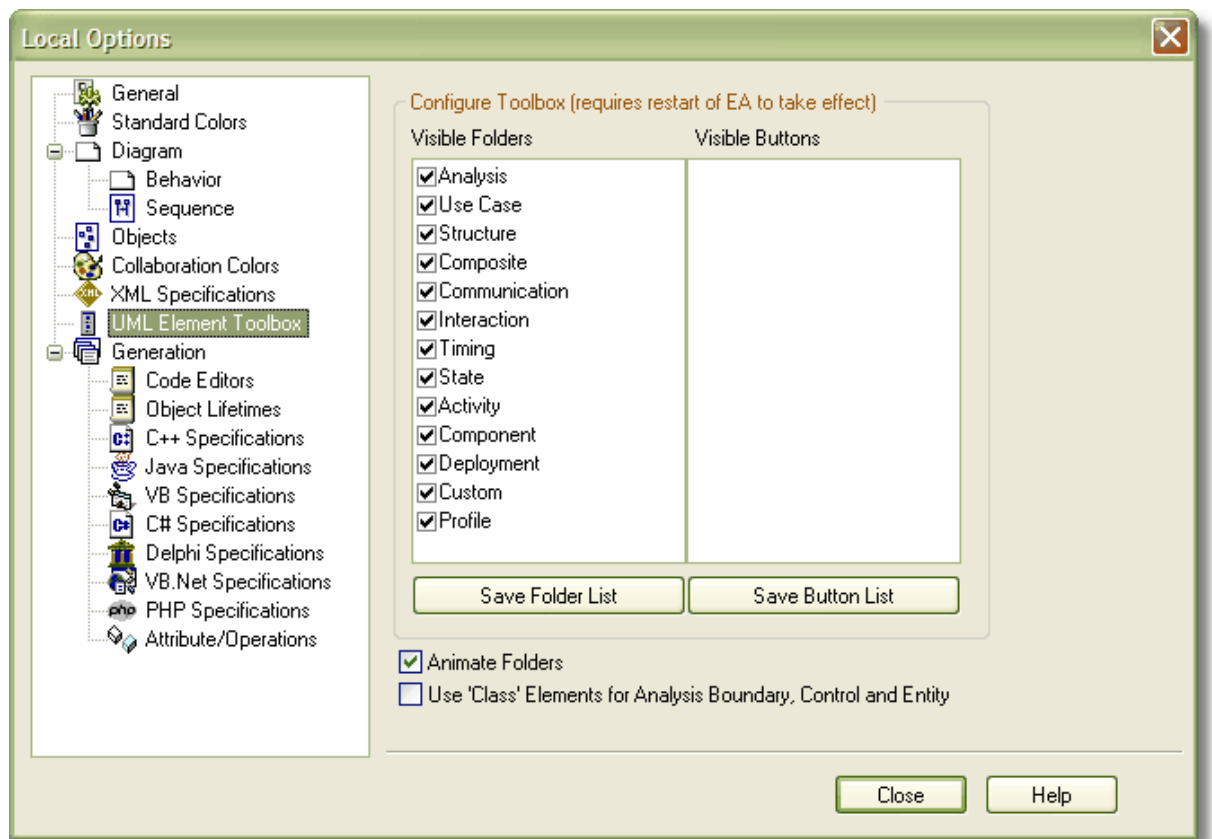
Create a Profile

To create a profile, drag a *Profile* item from the Profile section of the UML toolbox onto a diagram.



Once you have entered a name for the profile, a package will be created with the «profile» stereotype and a child diagram beneath it. This child diagram will be used to [add stereotypes](#) to the profile.

Note: The *Profile* section of the UML toolbox is not shown by default. To show this section, select *Tools | Options | UML Element Toolbox* on the main menu, tick *Profile* at the bottom of the *Visible Elements* list, then press the *Save Folder List* button. Close the *Local Options* window. The *Profile* section should now show in the UML toolbox as below.



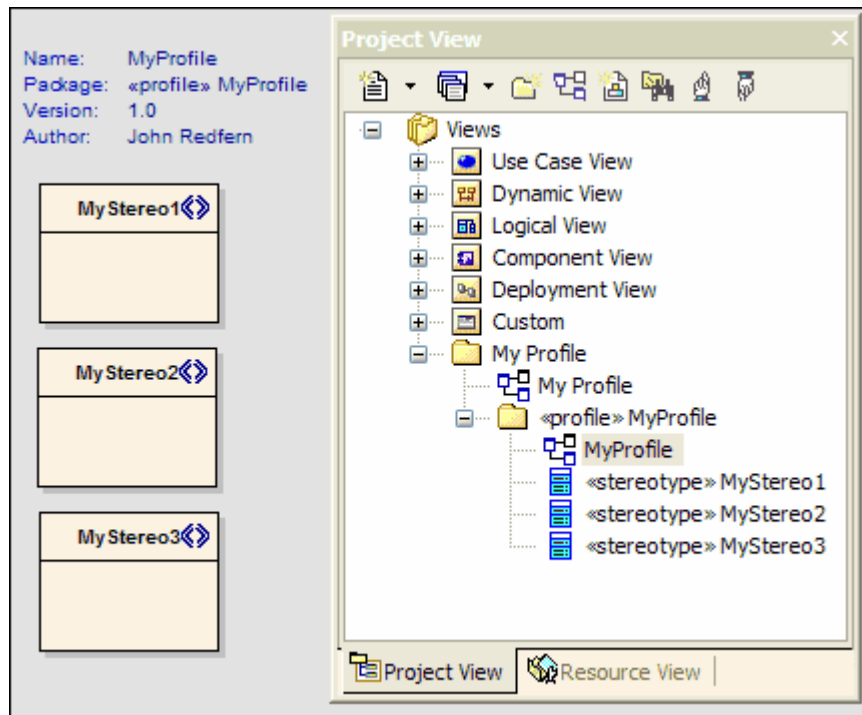
6.6.1.2 Add Stereotypes and Metaclasses to UML Profiles

Once a Profile has been created it is possible to define new stereotype elements and metaclasses.

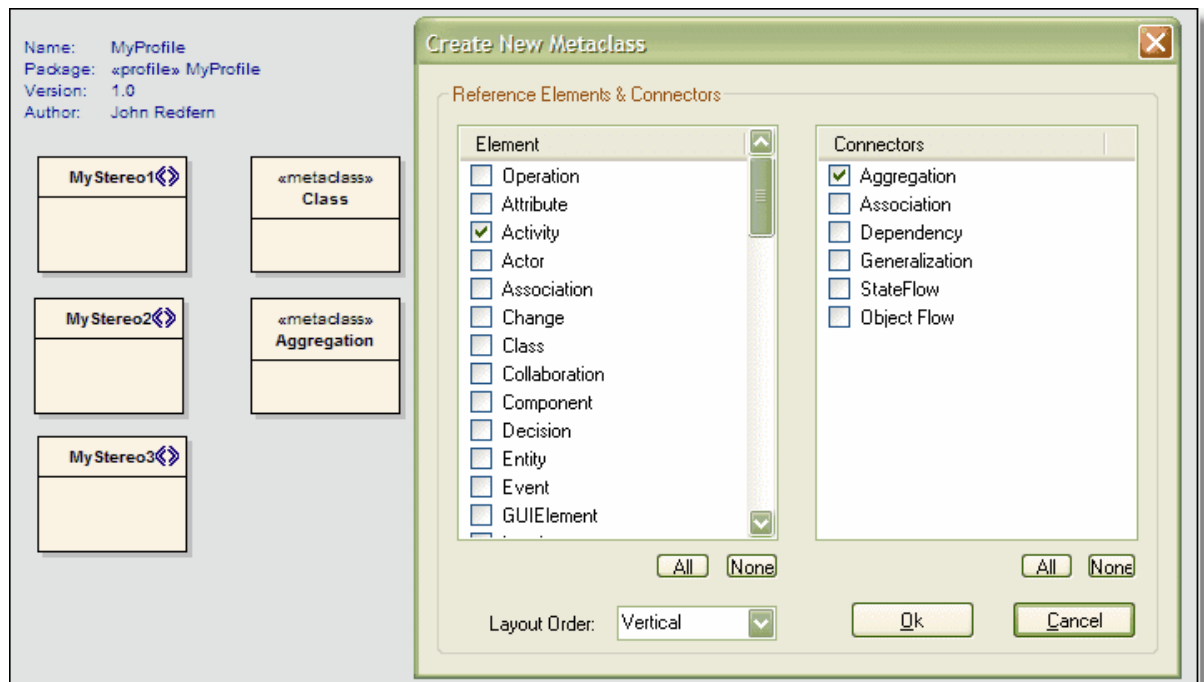
Adding Stereotypes and Metaclasses to a Profile

To add stereotypes to a profile, follow the steps shown below:

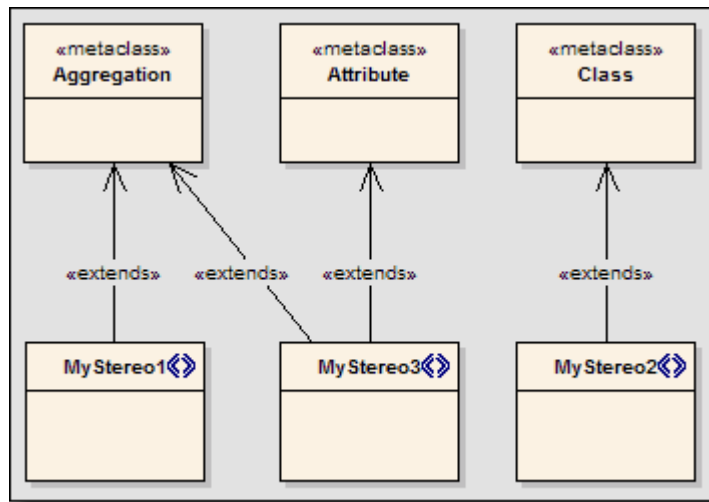
1. Open up the diagram contained in the profile package.
2. Create your *Stereotype* element(s) in the diagram belonging to the profile package by dragging the *Stereotype* tool from the *Profile Toolbox*.



3. Create the *Metaclass*(es) which your profile stereotypes will extend by dragging the *Metaclass* tool from the *Profile Toolbox*.



4. *Extend* your metaclasses with your profile stereotypes by creating *Extension* associations in your profile model.



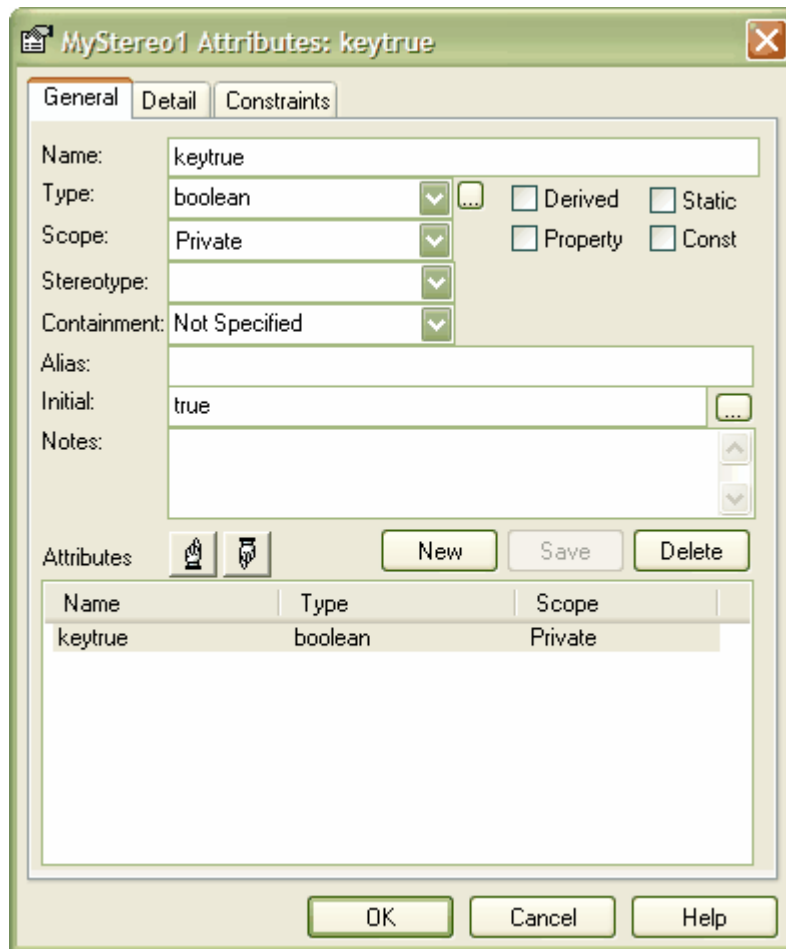
6.6.1.3 Define Stereotype Tags

Stereotypes within a UML Profile may have one or more associated tagged values. In creating a UML Profile, these tagged values are defined as attributes of the stereotype class.

Defining Stereotype Tags

To define Tagged Values for a stereotype, follow the steps shown below:

1. Open the [Attributes](#) dialog for the stereotype element.

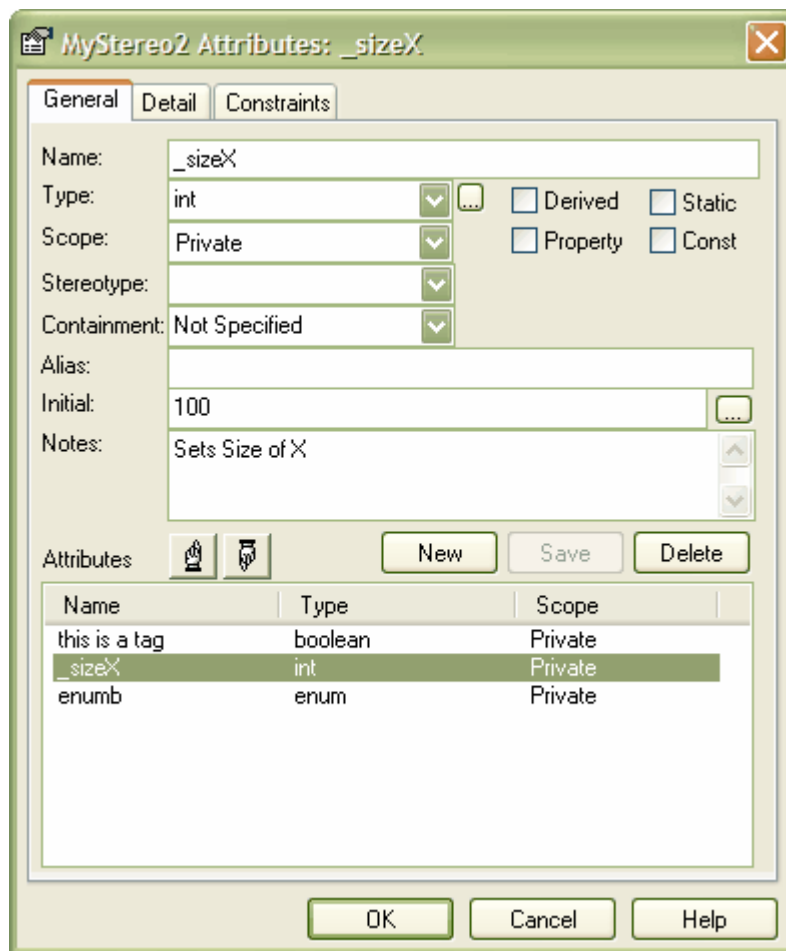


2. Press **New** to create a new attribute.
3. Enter the name of the stereotype tag in the **Name** field.
4. Set the **Type** and the **Initial** value of the tag.
5. Add a description of the tag into the **Notes** field.
6. Press **Save**

Defining Stereotype Tags with Supported Attributes

Supported stereotype attributes tags are special tags which set the default behavior of stereotyped elements such as the initial size of the element and the default location of any image files associated with the stereotype, etc. To define Tags for a stereotype with the supported attributes, follow the steps shown below (for a list of supported attributes go to the [supported attributes](#) section):

1. Open the [Attributes](#) dialog for the stereotype element.



Note: For supported attributes set only the *Name* (which needs to match the attributes listed in the supported attributes section) and the *Initial* value, do not set the other values.

2. Enter the name of the stereotype Tag in the *Name* field.
3. Set the initial value of the tag.
4. Press *Save*

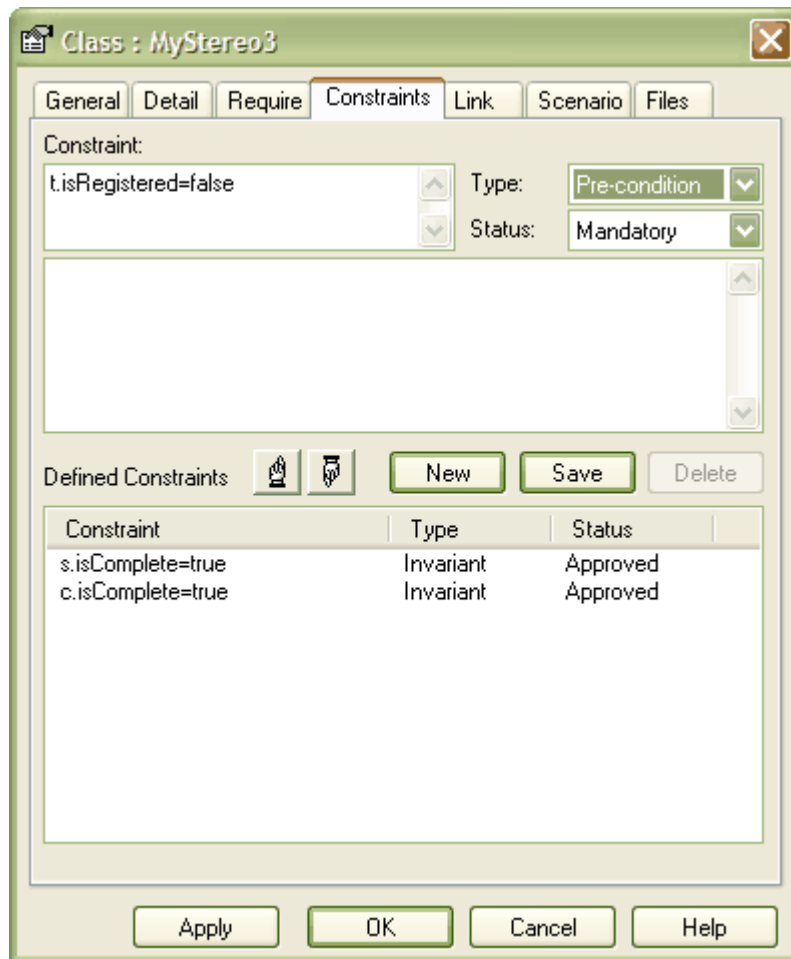
6.6.1.4 Define Stereotype Constraints

Defining constraints for stereotypes follows the same procedure as defining constraints for any class.

Defining Stereotype Constraints

To define constraints for a stereotype, follow the steps shown below:

1. Open the Class Properties dialog of the stereotype element in a diagram.
2. Open the *Constraints* tab and press *New* to create a new constraint.



3. Enter the value (in the *Constraint* field), the *Type* and the *Status* of the Constraint.
4. Enter any additional information in the Notes field.
5. Press *Save*.

6.6.1.5 Add Enumerations to UML Profiles

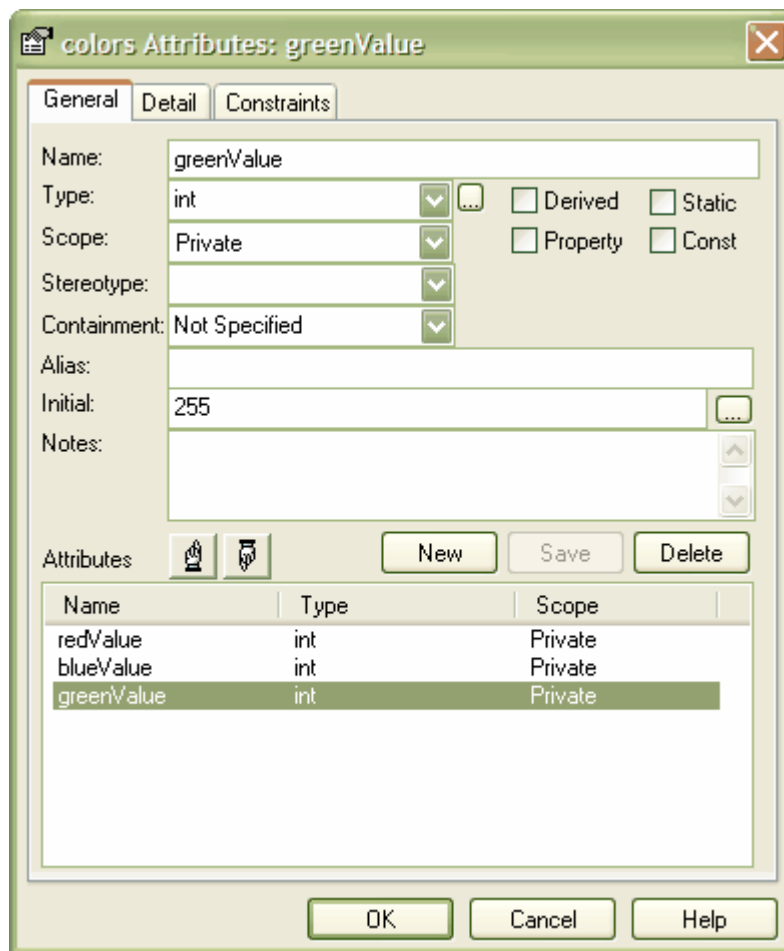
Enumerations may be used to restrict the values available to stereotype tags.

Note: Enumerations defined under a profile package do not appear as elements in the profile when imported.

Adding Enumerations Elements to UML Profiles

To add an enumeration element, follow the steps detailed below:

1. Drag an *Enumeration* item from the Profile section of the UML toolbox onto the diagram.
2. Set the name of the new Enumeration into the *Name* text field.
3. Open the *Detail* tab of the Enumeration and press the *Attributes* button.



4. Enter the name of the enumeration attribute in the *Name* field.
5. Set the *Type* and the initial value of the enumeration attribute.
6. Add any relevant notes into the notes text field.
7. Press the *Save* button.
8. Open the attribute properties for a stereotype.
9. Use the enumeration defined above in the attribute type field to restrict the corresponding tagged value set.

6.6.1.6 Export a UML Profile

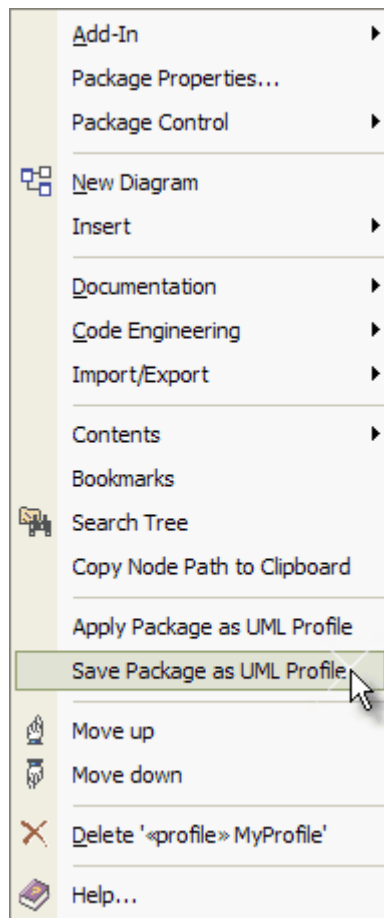
UML Profiles provide a convenient way to extend the base UML elements with stereotyped additional elements which can be used to extend and create new modeling languages. Once a profile has been created and the elements and metaclasses have been defined it is then possible to save the profile for future UML models.

Create a Profile

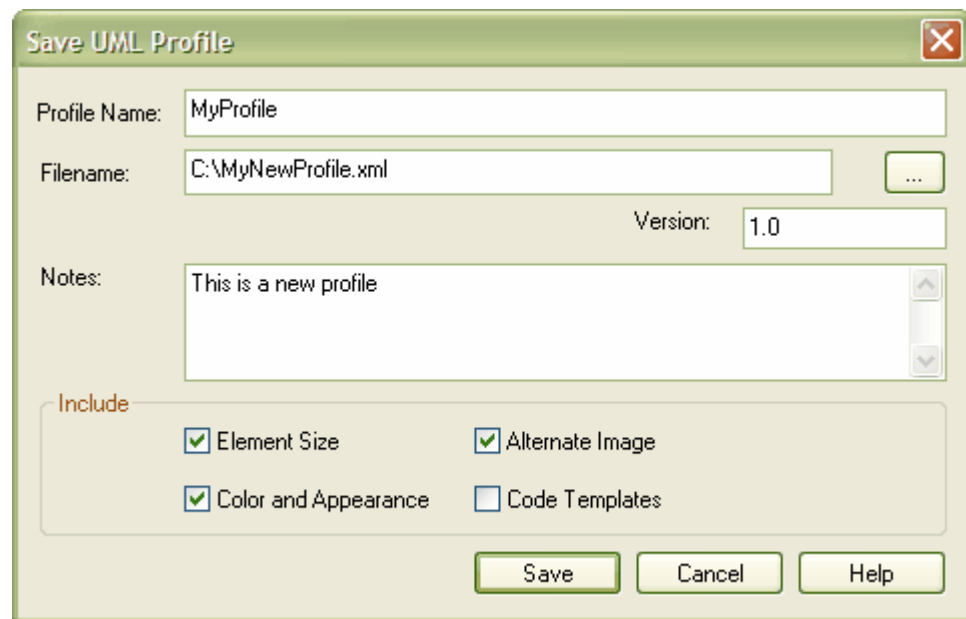
To save a profile, follow the steps shown below:

1. When you have completed defining your profile you can save your profile to disk by right-clicking anywhere in the profile diagram, or on the profile package in the *Project View* and select *Save*

Package as UML Profile.



2. The *Save UML Profile* Dialog will then be opened



3. Determine the destination for XML Profile file to exported using the *Browse [...]* button.
4. Set the required export options for all stereotypes defined in the profile - you can select to export
 - Element Size yes/no - check this to export the element size attributes.
 - Color and Appearance yes/no - check this to export the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image yes/no - check this to export the metafile image.
 - Code Templates yes/no - check this to export the code templates if they exist.
 - Version - enter information to determine the version of the profile.
5. Press *Save* to save the profile to disk.

Your profile, and its stereotypes, will now be available to use with other EA models by using the [Import Profile](#) option.

6.6.2 Using Profiles

This section describes the use of profiles for UML modeling. It describes tasks including how to import an available profile for use in a model, how to create elements and connectors using the stereotypes contained in the profile, applying and synchronizing values and constraints.

The following topics are discussed in this section:

- [Import a UML Profile](#)
- [Add Profiles to UML Toolbox](#)
- [Example Use of UML Profile](#)
- [Tagged Values in Profiles](#)
- [Add Profile Connector to Diagram](#)
- [Synchronize Tags and Constraints](#)

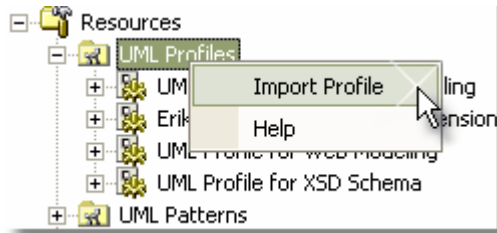
6.6.2.1 Import a UML Profile

To import a profile you will need a suitable Profile XML file - for example, like the profiles supplied on the Sparx Systems website at www.sparxsystems.com.au/uml_profiles.htm. If the Profile includes references to any Metafiles, they should be in the same directory as the XML profile.

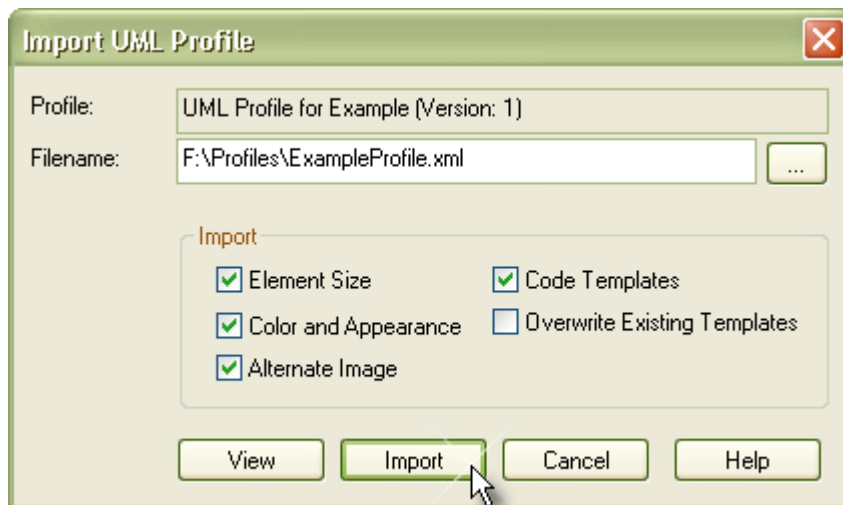
Import a Profile

To import a profile, follow the steps below:

1. Right click on the UML Profiles tree node and select ***Import Profile*** from the context menu - as in the example below.



2. The ***Import UML Profile*** dialog will open.



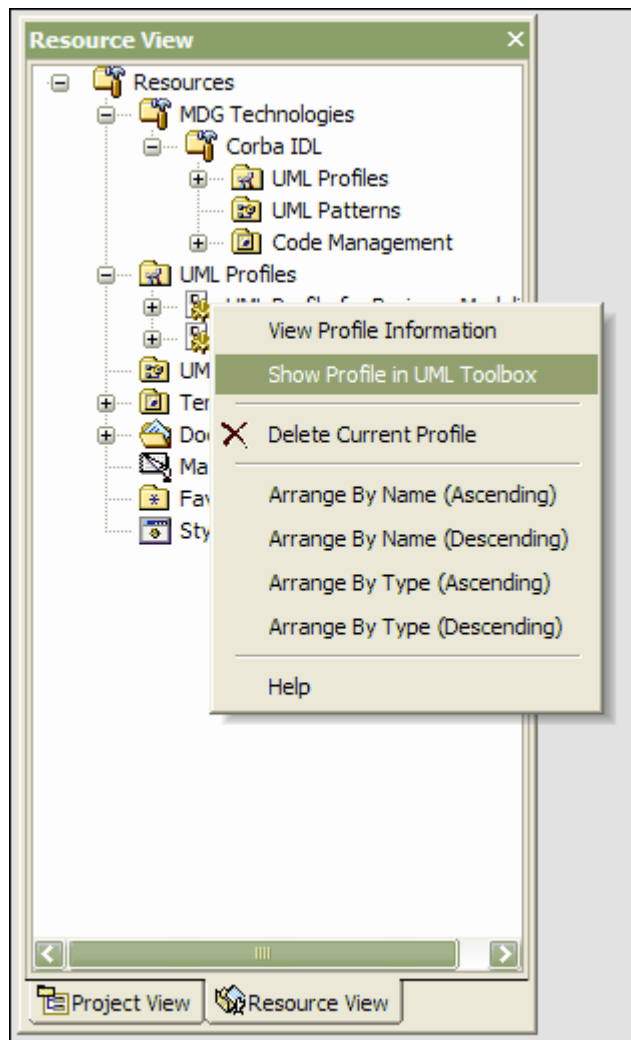
3. Locate the XML Profile file to import using the ***Browse [...]*** button.
4. Set the required import options for all stereotypes defined in the profile - you can select to import:
 - Element Size yes/no - check this to import the element size attributes.
 - Color and Appearance yes/no - check this to import the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image yes/no - check this to import the metafile image.
 - Code Templates yes/no - check this to import the code templates if they exist.
 - Overwrite Existing Templates yes/no - check this to overwrite any existing code templates defined in the current project.
5. Press ***Import***.

If the profile already exists, EA will offer to overwrite the existing version and import the new one - or cancel. Once the import is complete, the profile is ready to use.

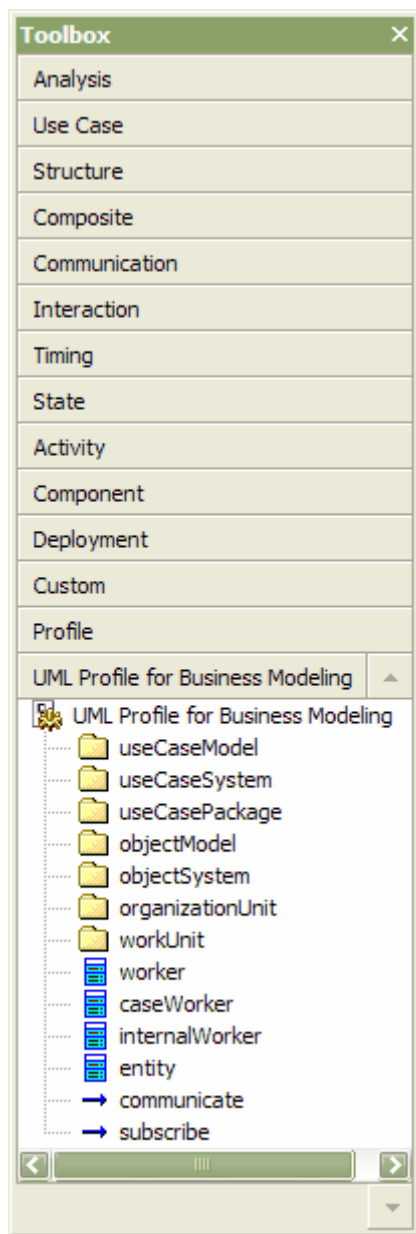
6.6.2.2 Add Profiles to UML Toolbox

MDG technologies and UML profiles can be added to the UML tool box to enable the quick use of the items contained in the profile or technology file. To add a profile to the UML toolbox use the following instructions:

1. Locate the Profile in the Resource View.
2. Right click on the Profile file in the resource window to bring up its context menu and select the *Show Profile in UML Toolbox* option.

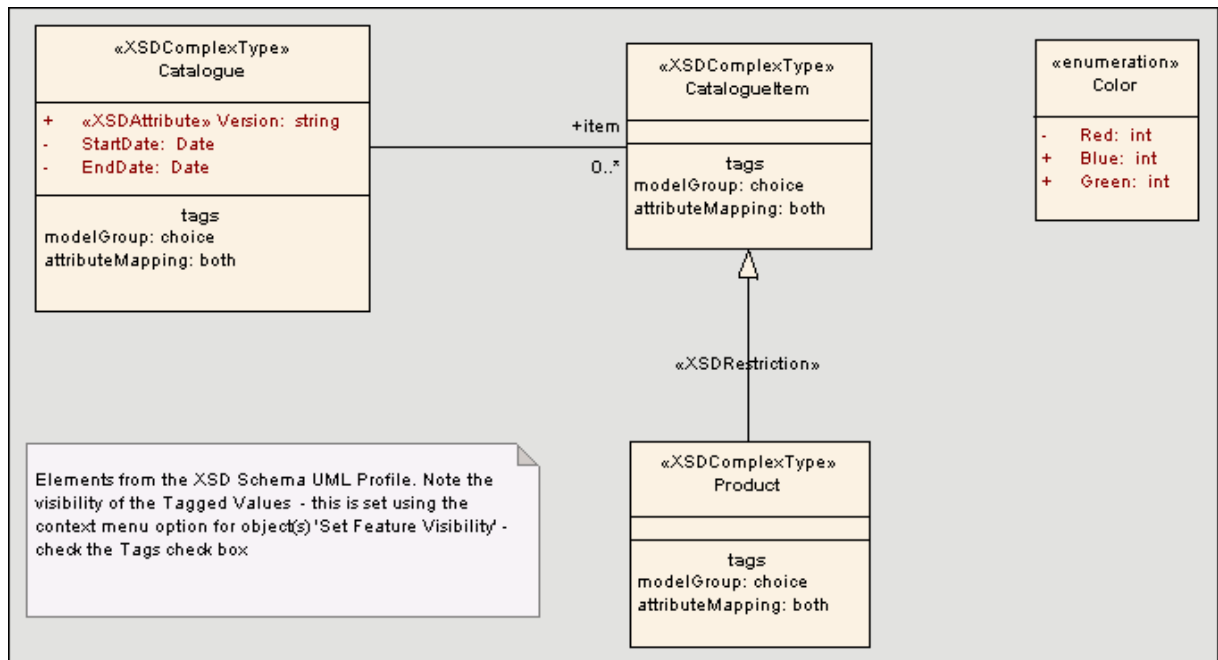


3. This will add the Profile to the UML Toolbox, from here the user can drag the items contained in the Profile directly into a diagram.



6.6.2.3 Example Use of UML Profile

The following example demonstrates some points about profile elements:



Note: Stereotypes are specified based on Profile stereotype. Attributes also may be dropped onto elements in a diagram and have their stereotype applied (eg. `<<XSDAttribute>>` in the diagram above). Links may be stereotyped and dropped from profile - eg. the `<<XSDRestriction>>`

Note: You can display the tagged values in the tags compartment of an element (not all elements support the tags compartment). The tags compartment can be shown using the context menu option - 'Set Feature Visibility'.

6.6.2.4 Tagged Values in Profiles

Stereotypes within a UML Profile may have one or more associated tagged values. When you create an element based on a UML Profile Stereotype by dragging from the Resource Tree to a diagram, any associated tagged values are added to the element as well. Tagged Values and Profiles are an excellent way to extend the usage of EA and the power of UML modeling.

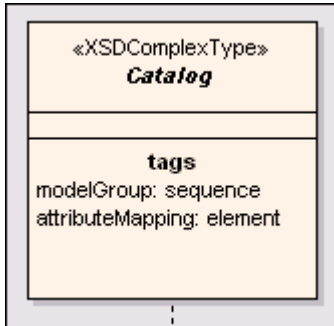
For example, in the UML Profile for XSD, there is an `XSDComplexType` stereotype, which has the following tagged value declaration

```
<TaggedValues>
  <Tag name="mixed" description="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>
```

Note: When you create an element from the `XSDComplexType` stereotype (by dragging from the UML Profile

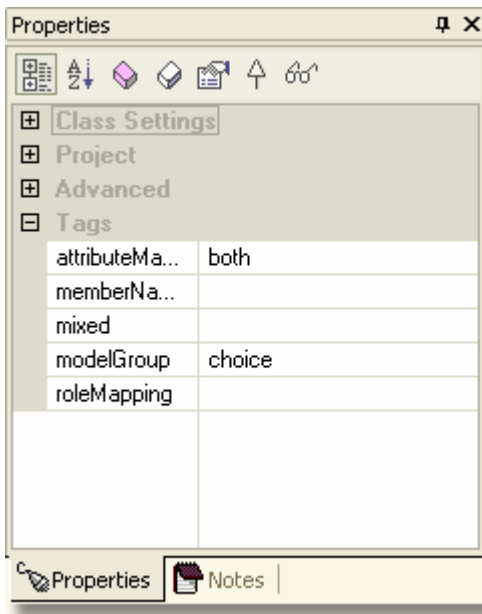
section onto a diagram), the tagged values are added automatically.

Tagged values which have default values are automatically set and displayed in the element tags section if applicable.



When you select the element, the Property Browser adds a Tagged Values section and displays all the associated tags - including ones that have no value set.

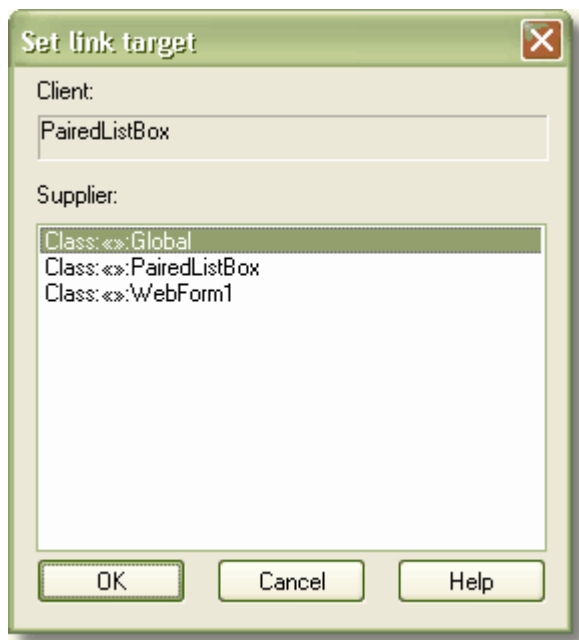
Also note that tagged values in the Profile that have a 'Values' section (eg. values="element | attribute | both" default="both") will display in the Property Browser with a drop list of allowable values (as in the example below). Where no Value list exists, the tag accepts free text.



6.6.2.5 Add Profile Connector to Diagram

To add a profile based connection to the current diagram, left click on the connection in the resource tree, then left click the source element in the diagram and drag it to the target.

You can also select the source and use the list box below to select the target.



6.6.2.6 Synchronize Tags and Constraints

When you first create an element, attribute, operation or link from a UML Profile item, tagged values and constraints may be created as well. Over time it may be that you modify the tagged values and constraints associated with a particular element - so the items already created may be missing additional tagged values or constraints.

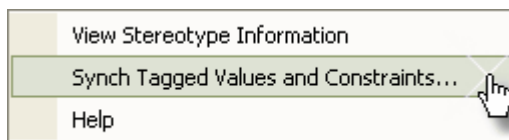
Likewise, you may have manually set the stereotype on a set of elements - and now want them to receive the tagged values and constraints normally associated with that stereotype.

To make sure you have all the related tagged values and stereotypes, use the *Synch Tagged Values and Constraints* function.

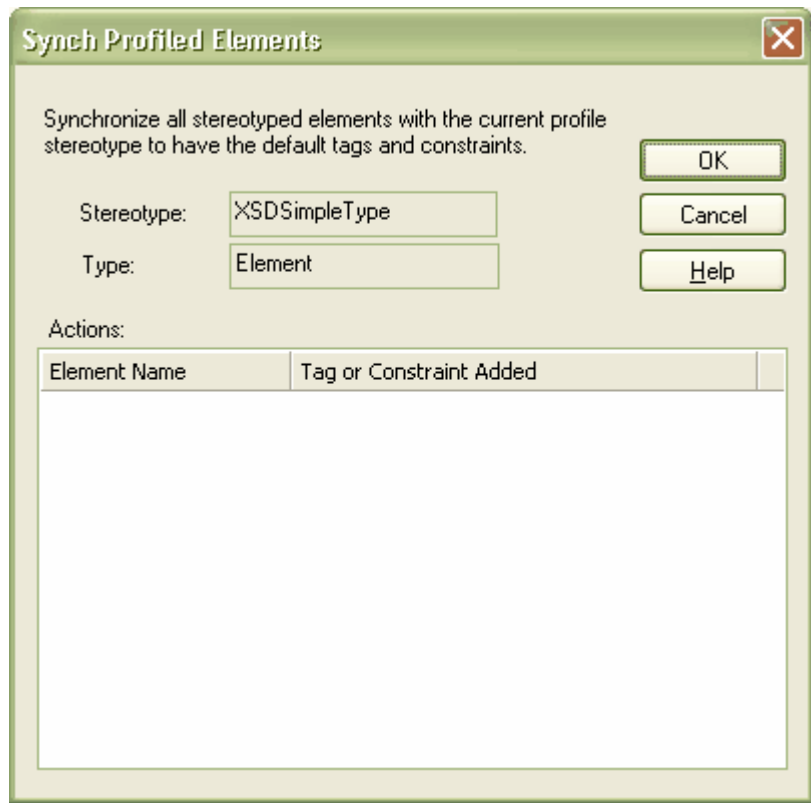
Synchronize Elements

To synchronize elements, follow the steps below:

1. Locate the required UML Profile in the Resource tree.
2. Locate the stereotyped profile element.
3. Right click and select the *Synch Tagged Values and Constraints* option.



4. When the *Synchronize tagged values and constraints dialog* appears - press **OK** to proceed - the list will be populated with the items that have been modified and the change that was made.



6.6.3 Deprecated Profile Tasks

The **deprecated** profile tasks do not fully support the new UML 2.0 method of dealing with profiles, contained within this section are the deprecated methods for applying profiles to a current model and saving diagrams as UML profiles.

6.6.3.1 Save a Diagram as a UML Profile (Deprecated)

In Enterprise Architect you can load Profiles from specially constructed XML source files, and use the new modeling elements as you would any of the standard UML modeling elements.

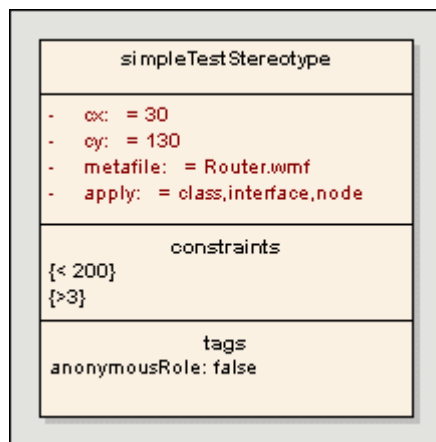
Note: that this method is now **deprecated**.

It is possible to write a new profile directly in XML - but it is more convenient and simpler to use the inbuilt **Save as Profile** function in the context menu of any diagram.

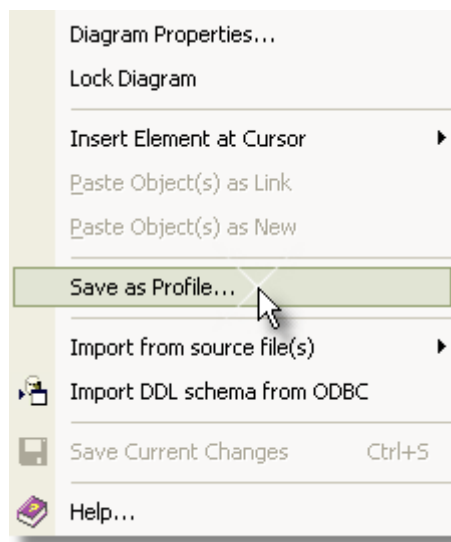
Creating a Profile

The basic rules for defining UML Profile stereotypes for your profile are below:

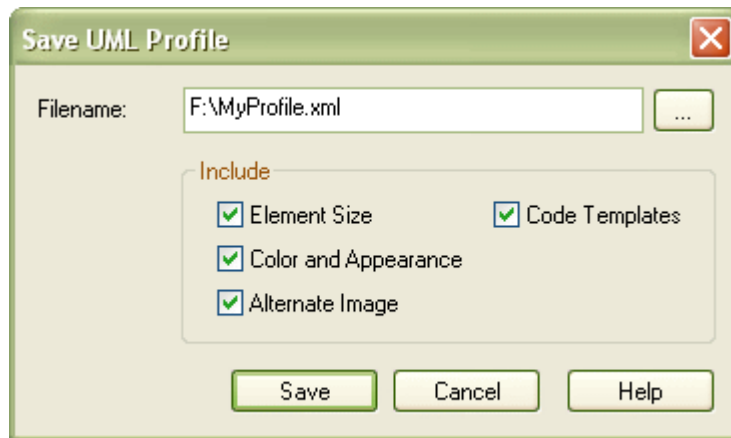
1. Create a class diagram to use to create your profile.
Note: It must be a **class** diagram.
2. Supply the diagram with a name, version number and notes explaining the profile's purpose.
3. Add a Class element for each stereotype you wish to create.



- As shown in the example above, for each class:
 - Provide the Name (which will become the UML Profile stereotype).
 - Provide 0 or more Constraints to apply to that stereotype.
 - Provide 0 or more Tagged Values to apply to the stereotyped element.
- As shown in the example above, for each class, provide the following Attributes (using the *Attribute* dialog):
 - Optionally, `cx` and `cy` attributes with numeric values describing the default size of the profile element.
 - Optionally, `metafile` attribute with the name of a suitable `.wmf` or `.emf` metafile which will be the visual representation of the stereotype (must exist in the same directory as the profile at load time).
- For each class you *must* provide an attribute (using the *Attribute* dialog) named "apply" with the default value set to the comma separated list of elements which this stereotype applies to (eg. class, interface, node).
- Once you have set up these parameters and all of your classes, right click on the diagram and choose *Save as Profile* from the context menu.



- The *Save UML Profile* dialog will appear.



9. Select the filename to save the XML Profile to using the *Browse [...]* button.
10. Set the required save options for all stereotypes defined in the profile - you can select to include:
 - Element Size yes/no - check this to save the element size attributes.
 - Color and Appearance yes/no - check this to save the color (background, border and font) and appearance (border thickness) attributes.
 - Alternate Image yes/no - check this to save the metafile image.
 - Code Templates yes/no - check this to save the code templates if they exist.
11. Press *Save*.

Loading Your Profile into EA

To [Import a Profile](#), refer to the [Import a Profile](#) section.

6.6.3.2 Applying Profiles to Current Model (Deprecated)

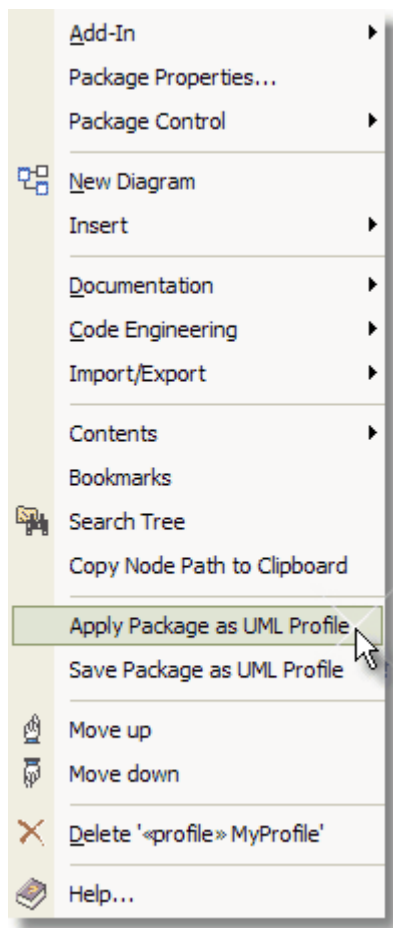
Once a profile has been created and the elements and metaclasses have been defined it is then possible to apply the profile to the current UML model.

Note: that this method is now *deprecated*.

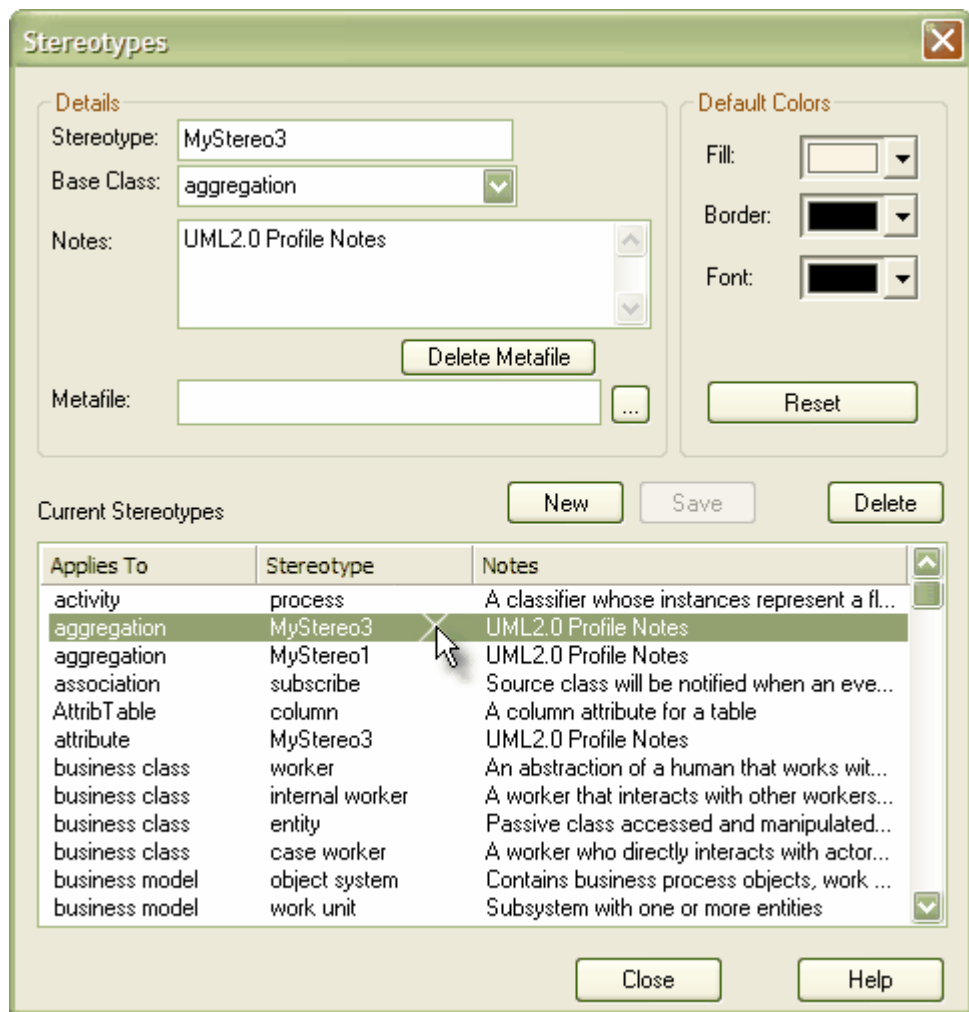
Create a Profile

To apply a profile to the current model, follow the steps shown below:

1. When you have completed defining your profile you can apply your profile to the rest of your model by right-clicking anywhere in the profile diagram, or on the profile package in the *Project View* and select *Apply Package as UML Profile*.



The newly created profile, and its stereotypes, will now be available in your model to apply to other elements in the current model.



6.6.4 Profile References

This section gives the user a reference of the supported tags that are available for users, the structure of profiles, supported attributes and an example of the XML file that constitutes a profile.

UML Profile Reference Subjects:

- [Supported Types](#)
- [Profile Structure](#)
- [Supported Attributes](#)
- [Example Profile](#)

6.6.4.1 Supported Types

UML Profile XML File Format Information

Enterprise Architect provides a facility to import pre-defined elements, operations, attributes and connectors as a source of re-useable components that meet common modeling needs (eg. profiles for XML schema, for business process modeling etc.). This document provides a quick list of the types of things that can be pre-defined and the characteristics of each. There is an example file at the end which indicates how each type is

specified.

A UML profile is made up of one or more "stereotypes" that may have "tagged values" and "constraints". The table below and the [Supported Attributes](#) table defines what can be stereotyped and what information needs to be supplied.

List of All Supported Types in Applies To/Apply Node

AppliesTo/ Apply	Type	Tags	Tags in Props Wind	Constraint	Metafile
"actor"	Element	Yes	Yes	Yes	Yes
"package"	Package	Yes	Yes	Yes	Yes
"usecase"	Element	Yes	Yes	Yes	Yes
"collaboration"	Element	Yes	Yes	Yes	Yes
"class"	Element	Yes	Yes	Yes	Yes
"table"	Element	Yes	Yes	Yes	Yes
"component"	Element	Yes	Yes	Yes	Yes
"node"	Element	Yes	Yes	Yes	Yes
"object"	Element	Yes	Yes	Yes	Yes
"sequence"	Element	Yes	Yes	Yes	Yes
"entity"	Element	Yes	Yes	Yes	Yes
"screen"	Element	Yes	Yes	Yes	Yes
"GUIElement"	Element	Yes	Yes	Yes	Yes
"requirement"	Element	Yes	Yes	Yes	
"state"	Element	Yes	Yes	Yes	
"activity"	Element	Yes	Yes	Yes	Yes
"interface"	Element	Yes	Yes	Yes	Yes
"event"	Element	Yes	Yes	Yes	
"issue"	Element	Yes	Yes	Yes	
"change"	Element	Yes	Yes	Yes	
"hyperlink"	Element		Yes	Yes	
"attribute"	Attribute	Yes		Yes	
"operation"	Operation	Yes		Yes	
"association"	Connector	Yes		Yes	
"associationEnd"	AssociationEnd				
"generalization"	Connector	Yes		Yes	
"dependency"	Connector	Yes		Yes	
"stateflow"	Connector	Yes		Yes	
"objectflow"	Connector	Yes		Yes	
"startnode"	Element	Yes	Yes	Yes	
"stopnode"	Element	Yes	Yes	Yes	
"note"	Element	Yes	Yes	Yes	
"decision"	Element	Yes	Yes	Yes	
"aggregation"	Connector	Yes		Yes	

6.6.4.2 Profile Structure

UML Profiles for Enterprise Architect are distributed in XML format. The file follows the following format:

General Header Details

The following is the general header details:

```
<?xml version="1.0"?>
  <UMLProfile>
    <!--Profile name, version number and general notes -->
    <Documentation id="XSDSchema" name="UML Profile for XSD Schema" version="1"
      notes="Defines a set of stereotypes and tagged values for XSD Schemas"/>
    <!-- The profile content -->
    <Content>
      <!-- List of stereotypes used in this profile. May also include tagged
        values, constraints, metafile and descriptive comments-->
      <Stereotypes>
```

Stereotype Definitions

This is followed by one or more Stereotype definitions - for example:

```
<!-- <<XSDComplexType>> -->
<Stereotype name="XSDComplexType" notes="ComplexType definition generated in XML Schema">
  <AppliesTo>
    <Apply type="class"/>
  </AppliesTo>
  <TaggedValues>
    <Tag name="mixed" description="URI to unique target namespace"/>
    <Tag name="modelGroup" description="Default model group used when generating
      complexType definitions for this Schema" values="all | sequence | choice"
      default="choice"/>
    <Tag name="attributeMapping" description="Default for generating UML attributes as
      elements, attributes or both within complexTypes" values="element | attribute| both"
      default="both"/>
    <Tag name="roleMapping" description="Prefix associated with namespace"/>
    <Tag name="memberNames" description="Schema version"/>
  </TaggedValues>

  <Constraints>
    <Constraint name="" type="" notes=""/>
  </Constraints>
</Stereotype>
```

Note the specification of Stereotype name and notes. Also note the use of tagged values to set properties for the profile element. The tagged values may have a default value, may be empty and may specify allowable values. Tagged values are edited in the properties window of an element, method, attribute or link.

You can also specify default size, default comment and Metafile for drawing element - see the fragment below:

```
<Stereotype name="Router" cx="130" cy="100" notes="" metafile="router.emf">
```

In the above example, the metafile shape for this element is specified as 'router.emf' - when you load this profile, the .emf file MUST be in the same directory as the Profile - or the load will fail.

Also note how to specify a default comment for an element. All white space between lines will be ignored. To force a line break, use the '\n' character. To force tabs, use '\t'.

```
<Comment>
  Some text here about how this will work\n\t
  with comments being imported from the XML description
  in one long row.
</Comment>
```

The example above would import like this:

```
Some text here about how this will work
  with comments being imported from the XML description in one long row.
```

6.6.4.3 Supported Attributes

Details of Attributes Supported by Main XML Element Nodes

The table below lists the three main types of element you can define in an XML Profile document. These are the "stereotype" (each stereotype will create a visible entry in the Resource Tab/UML Profile window in EA). The "tagged values" are additional properties that an element or link support and "constraints" that apply to the model element.

Type	Attribute	Optional	Notes	
stereotype	name	No	Stereotype name	
	notes	Yes	Notes visible in browser	
	metafile	Yes	Filename of associated metafile. MUST be in same directory as Profile XML	
	cx	Yes	Initial x coordinate of element (deprecated)	
	cy	Yes	Initial y coordinate of element (deprecated)	
	_sizeX	Yes	Initial x coordinate of element	
	_sizeY	Yes	Initial y coordinate of element	
	_imageFile	Yes	Location of image file (wmf)	
tag	name	No	Tag name	
	description	Yes	A description of the tag - appears in tag tab and for elements in property window setting notes	
	values	Yes	List of possible values. Values separated by " " (<space> <space>). eg. "true false". For elements will populate drop combo in tag section of docked properties window	
	default	Yes	A default value. eg. "true"	
constraint	name	Yes	constraint name	
	type	Yes	constraint type (eg. pre for precondition, post for postcondition)	
	notes	No	Additional explanatory notes	

6.6.4.4 Example Profile

Below is an example UML Profile showing the structure of file and usage:

```
<?xml version="1.0" encoding="UTF-8"?>
<UMLProfile>
  <Documentation id="EAExample" name="UML Profile for Example" version="1" notes="An
example set of stereotypes and tagged values"/>
  <!-- The profile content -->
  <Content>
    <!-- List of stereotypes used in this profile-->
    <Stereotypes>
      <!--A profile is a list of stereotypes, that will apply to elements, links and
features in a UML model. Stereotypes may have set tagged values, constraints,
```



```

Valid targets, default dimensions etc. The examples below are a good starting
point -->
<Stereotype name="SimpleStereotype" notes="Place notes about stereotype here"
metafile="router.emf">
  <!-- Place a list of types that this ill apply to ...
  valid types are any UML element (class, interface, component etc.,
  aggregation, generalization, association, stateflow etc., operation and
  attribute. Make sure you use lowercase names, XML is case sensitive-->
  <AppliesTo>
    <Apply type="class"/>
    <Apply type="interface"/>
    <Apply type="node"/>
  </AppliesTo>
  <!--Add one or more tagged values for this stereotype. These will be
  automatically added to the target element when created
  Note that you can specify a default value using "default=" and a pick list of
  values eg. " true | false" note the use
  of a " | " to separate values -->
  <TaggedValues>
    <Tag name="hasNamespace" description="Indicates element is bound to
  Namespace" default="true" values="true | false"/>
    <Tag name="targetNamespacePrefix" description="Prefix associated with
  namespace"/>
  </TaggedValues>
  <!-- Zero or more constraints to apply to element - specify name, type and
  notes -->
  <Constraints>
    <Constraint name="constraint1" type="pre" notes="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"/>
  </TaggedValues>
  <Constraints>
    <Constraint name="constraint1" type="pre" notes="My Notes"/>
  </Constraints>
</Stereotype>
<!-- <<Aggregation>> -->
<Stereotype name="aggregationLink" type="weak" notes="">
  <AppliesTo>
    <Apply type="aggregation"/>
  </AppliesTo>
</Stereotype>
<!-- <<Composition>> -->
<Stereotype name="compositionLink" type="strong" notes="">
  <AppliesTo>
    <Apply type="aggregation"/>
  </AppliesTo>
</Stereotype>
<!-- <<IndexKey>> -->
<Stereotype name="UniqueID" notes="">
  <AppliesTo>
    <Apply type="operation"/>
  </AppliesTo>
  <TaggedValues>
    <Tag name="indexed" description="indicates if indexed or not" values="true
  | false" default="true"/>
  </TaggedValues>
  <Constraints>
    <Constraint name="constraint1" type="pre" opType="pre" notes="My Notes"/>
    <Constraint name="constraint2" type="pre" opType="post" notes="My Notes"/>
  </Constraints>
</Stereotype>
<!-- <<Attribute>> -->

```

```

<Stereotype name="attname" notes="">
  <AppliesTo>
    <Apply type="attribute"/>
  </AppliesTo>
  <Constraints>
    <Constraint name="constraint1" type="pre" notes="My Notes"/>
  </Constraints>
</Stereotype>
<!-- <<Association>> -->
<Stereotype name="assocname" notes="">
  <AppliesTo>
    <Apply type="association"/>
  </AppliesTo>
  <Constraints>
    <Constraint name="constraint1" type="pre" notes="My Notes"/>
  </Constraints>
</Stereotype>
</Stereotypes>
</Content>
</UMLProfile>

```

6.7 UML Patterns

What is a Pattern?

Patterns are parameterized collaborations - that is they are a group of collaborating objects/classes that can be abstracted from a general set of modeling scenarios. Patterns are an excellent means of achieving re-use and building in robustness. As patterns are discovered in any new project, the basic pattern template from previous engagements can be re-used with the appropriate variable names modified for the current project.

Patterns generally describe how to solve an abstract problem, and it is the task of the pattern user to modify the pattern elements to meet the demands of the current engagement.

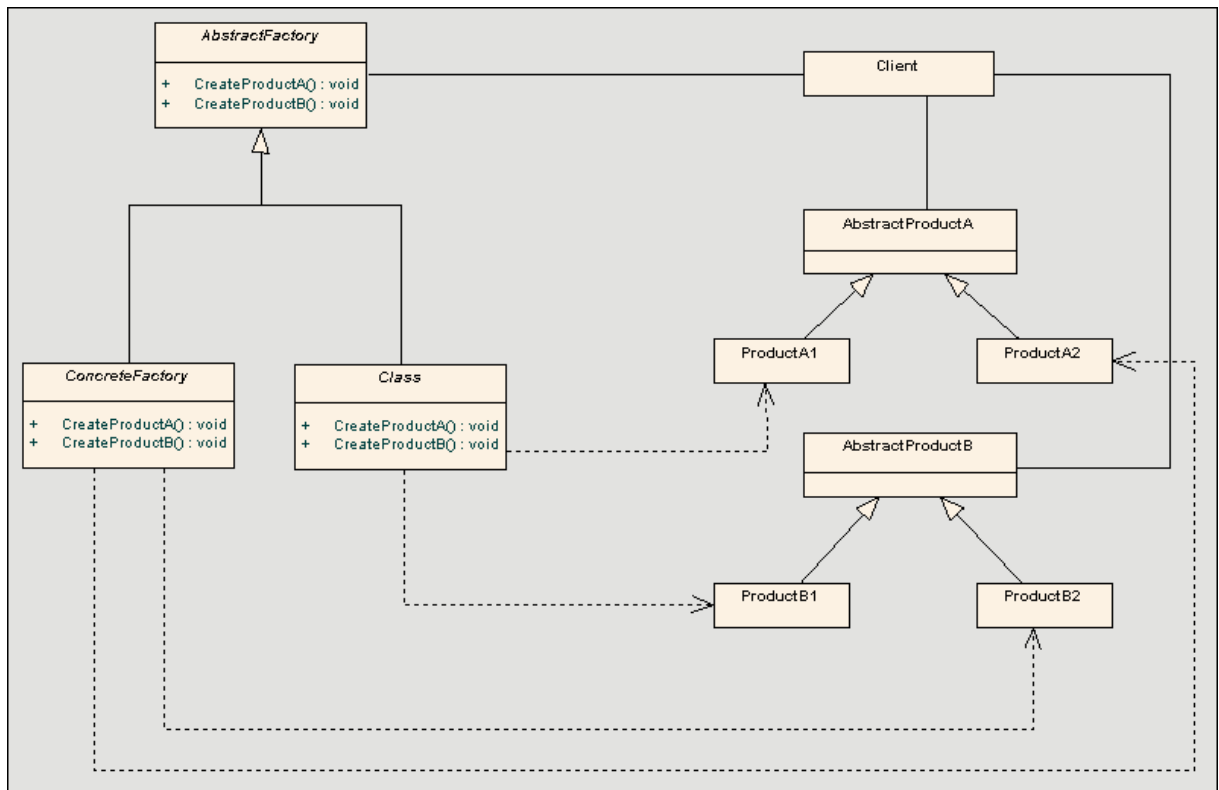
Before using a pattern it must first be [created](#) as a standard UML diagram and then saved as a XML pattern file. This XML file can then be [imported](#) as a UML Resource that may be [used](#) in any model.

Sparx created Gof Patterns

To get you started with Design Patterns in Enterprise Architect, the following zip file contains the Patterns described in the book "Design Patterns - Elements of Reusable Object-Oriented Software" by Gamma et al., referred to as the 'Gang of Four' or GoF for short. To download an example of the Gang of Four patterns for Enterprise Architect from http://www.sparxsystems.com.au/uml_patterns.html.

6.7.1 Create a Pattern

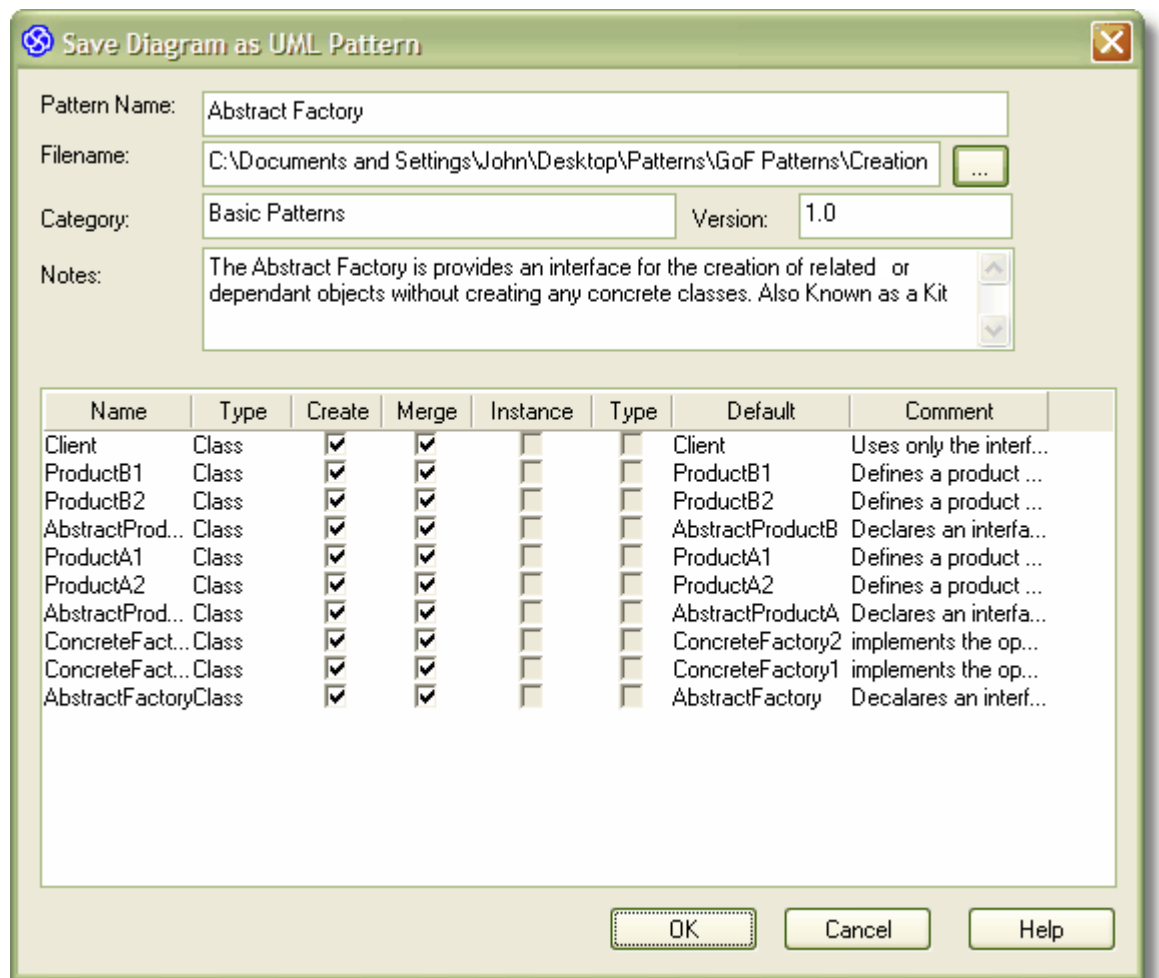
To create a pattern you will first need to model the pattern as standard UML Diagram from within EA. The following diagram was created from an example as set out in the GOF Design Patterns book.



Save a Diagram as a Pattern

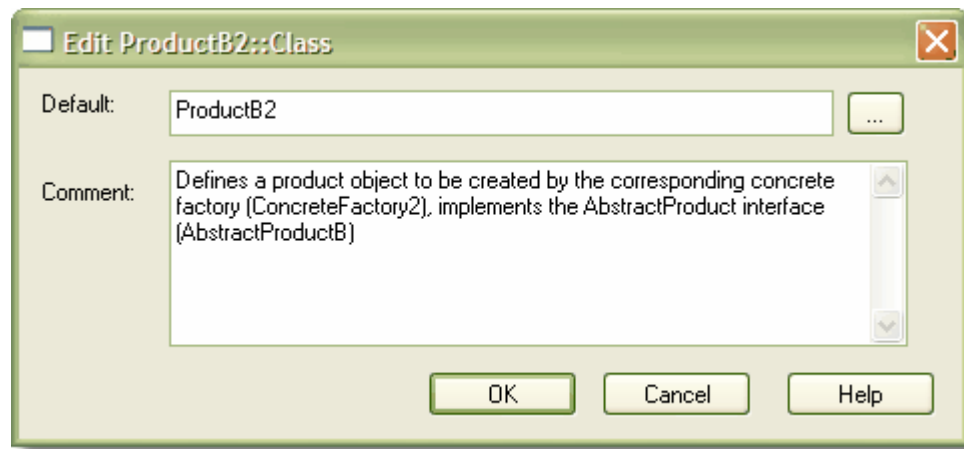
To save a diagram as a pattern, follow the steps below:

1. From the **Diagram** menu, select **Save as UML Pattern** to open the **Save Diagram as UML Pattern** dialog.



2. Set the *Pattern Name*.
3. Enter a *Filename* (.XML) to save the pattern as.
4. Select or enter a *Category* for the pattern to appear in under UML Patterns (required).
5. Enter the *Version* number and any additional *Notes*.
6. Select the appropriate actions for the elements that are contained in the pattern by checking the check boxes as appropriate. These actions are performed when the pattern is used (for more detail refer to the [Using Patterns](#) topic). The available actions are:
 - Create: Creates the pattern element directly without modification.
 - Merge: Merges the pattern element with an existing element, allowing the existing element to take on the role of the selected pattern element.
 - Instance: Creates the pattern element as an instance of an existing element. (this option is available if the pattern element supports this action).
 - Type: Creates the pattern element types as an existing element (this option is available if the pattern element supports this action).
7. To change the name of one of the elements, highlight the element and then click on the element to

bring up the *Edit* dialog, from this dialog the user can change the name of the element as well as adding comments detailing the elements purpose.

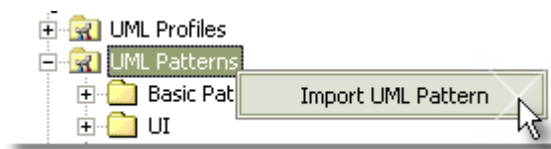


8. Press *OK* to save the pattern. Once saved you can load it into EA as a Pattern in the Resources tab.

6.7.2 Import a Pattern

Before using a previously [created pattern](#) file in a UML model it must first be imported into the current UML model which will then be available from the [Resource View](#) and optionally from the UML toolbox. To import a UML Pattern you have previously saved, follow the steps outlined below:

1. Select the *Resource View* tab on the Project Browser.
2. Right click on the *UML Pattern* node.
3. Select *Import UML Pattern* from the context menu
4. In the *File Find* dialog, locate the XML file to import.
5. Press *OK* to import the pattern.



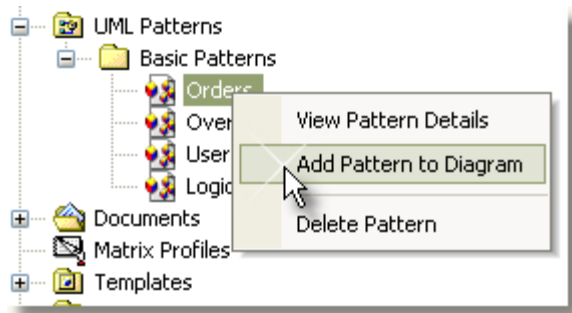
The imported pattern will be placed in the appropriate category as defined in the XML file. If the Category does not already exist under UML Patterns, a new one will be created. To download an example of the Gang of Four pattern for Enterprise Architect click [here](#).

6.7.3 Use a Pattern

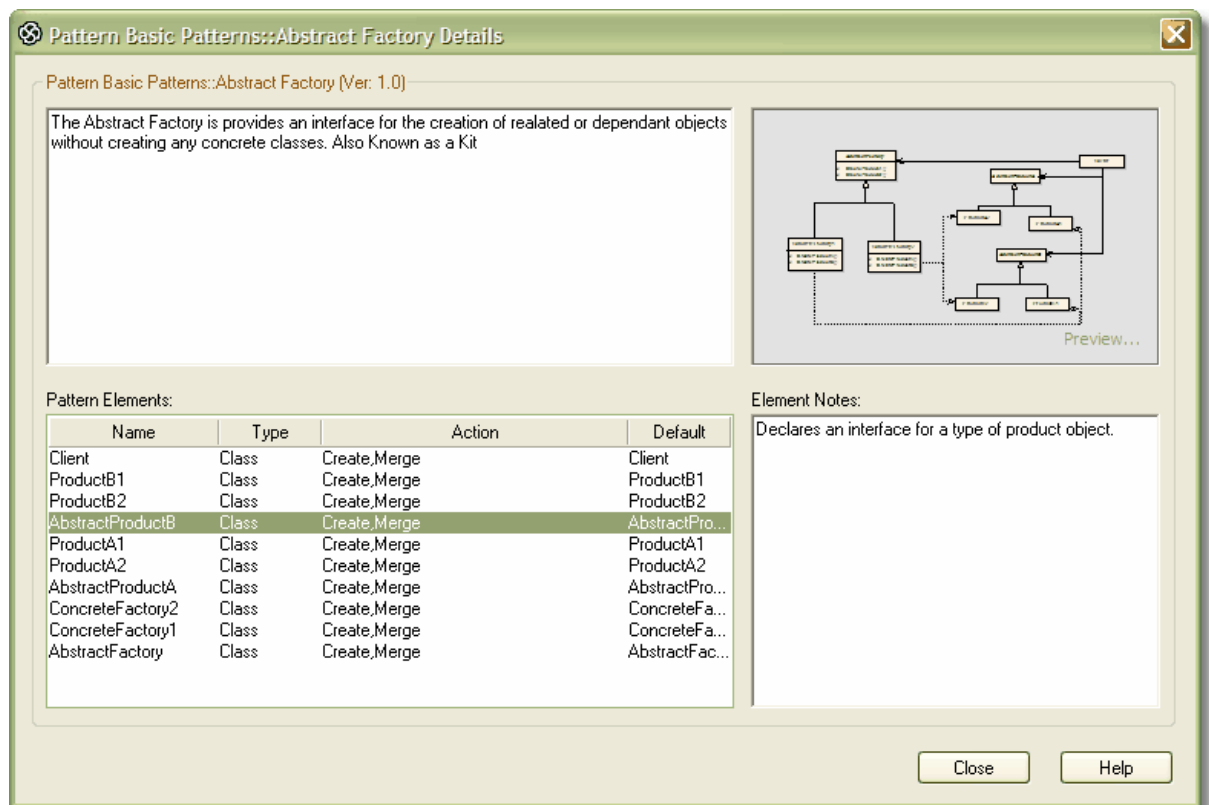
Using a pattern allows items that defined in the pattern to be used with the UML model. Using patterns allows for the rapid creation of template solutions for code structures have performed the same type of task in other situations.

To use a Pattern that you have [previously imported](#) into the model, follow the steps outlined below:

1. Open the diagram into which you want to add the UML pattern.
2. Select the *Resource View* tab on the Project Browser.
3. Open the *UML Pattern* folder and find the pattern you wish to add.
4. Right click on the pattern and select the *Add Pattern to Diagram* menu item or drag and drop the pattern from the Resource View onto the diagram or view the Pattern details by selecting the *View Pattern Details* menu item to view the pattern details in read only mode.



5. This will open the *Add Pattern* dialog (below).



6. The **Add Pattern** dialog allows users to perform several operations, these are detailed in the table below:

Control	Description
Preview	This pane allows the preview of the pattern, click on the Preview link to open a larger preview of the pattern.
Element Note	This pane is used to display the comments that describe the element in the pattern, highlight the element in the Pattern Elements pane to view the notes.
Pattern Elements	This pane provides access to the individual elements contained in the pattern, from here you can select the action for the individual element (Create , Merge, Instance or Type - as applicable for each element), Modify the name of the pattern element and choose the namespace for the merged element.
OK	Add the pattern to the diagram.
Cancel	Cancel the addition of the pattern to the diagram
Help	Opens the help file to display contextual help (this topic).

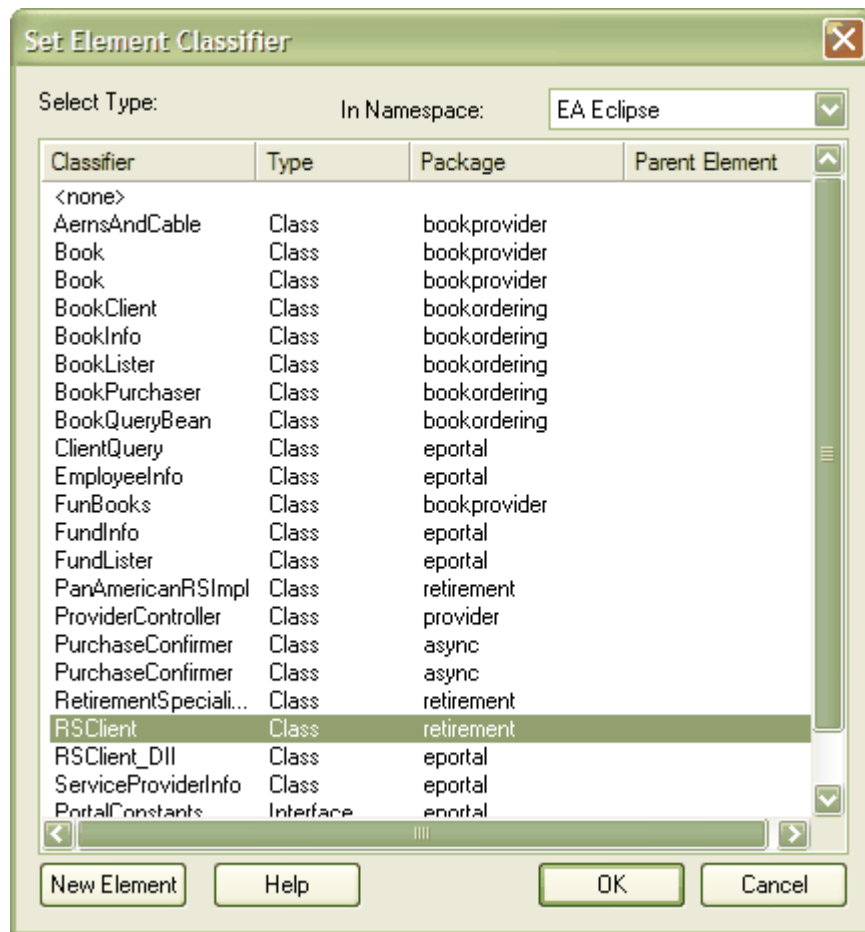
7. Once the appropriate selections have been made press the **OK** button to import the pattern into the model recreating the original diagram with new GUIDs.

Changing Pattern Element Name

The name of the pattern element may be changed by using the following instructions:

1. From the **Add Pattern** dialog select the individual element in the Pattern Element pane.

2. Click on the ... button to bring up the *Edit* dialog, the specific method for changing the element name will be dependant upon the entry in the *Action* section of the *Pattern Elements* pane.
3. If the Action selection is set to *Create* then in the *Edit* dialog the process used to rename the element is to delete the existing default name and replacing with a user defined name. The element name will now be changed in the *Add Pattern* dialog.
4. If the Action for the element is set as *Merge*, in the *Edit* dialog press the ... button to browse to an existing element classifier. This will bring up the *Set Element Classifier* dialog.
6. From the *Set Element Classifier* dialog select an existing element classifier from the list of available classifiers or restrict the number of available choices by selecting the elements from a specific namespace by selecting a namespace from the *In Namespace* drop down menu. For more information regarding setting Element Classifiers refer to the [Using Classifiers](#) topic.



6.8 Requirements Management

Requirements Management

Gathering requirements is typically the first step in developing a solution, be it for the development of a

software application or the detailing of a business process. Requirements are essentially "what the system needs to do". The requirements management built into EA can be used to define requirement elements, link requirements to model elements that implement that requirement, link requirements together into a hierarchy, report on requirements and move requirements into and out of model element requirements.

Typical Tasks

Typical tasks you might want to perform when using requirements with EA include:

- [Creating Requirements](#)
- [External Requirements](#)
- [Color Coding External Requirements](#)
- [Internal Requirements](#)
- [Move Internal Requirements to External Requirements](#)
- [Requirement Properties](#)
- [Composition](#)
- [Implementation](#)
- [Requirements's Hierarchy](#)
- [The Matrix](#)
- [Dependency Report](#)

6.8.1 Creating Requirements

Create Requirements at Diagram Level

To create requirements at the diagram level, follow the steps below:

1. Open the *Custom* group on the UML Toolbox.
2. Drag the requirement element onto the current diagram.
3. Enter *Name* and other details for the requirement.

EA creates a requirement in the current diagram and in the current package.

Create Requirements at Package Level

To create a requirement at the package level, follow the steps below:

1. Right click on a package to open the context menu.
2. Select *New Element* from the *Insert* menu. Alternatively, press *Ctrl+M*
3. In the *New Element dialog* select *Requirement* type.
4. Enter *Name* (or select *Auto*) and press *OK*.

EA creates a requirement in the current package.

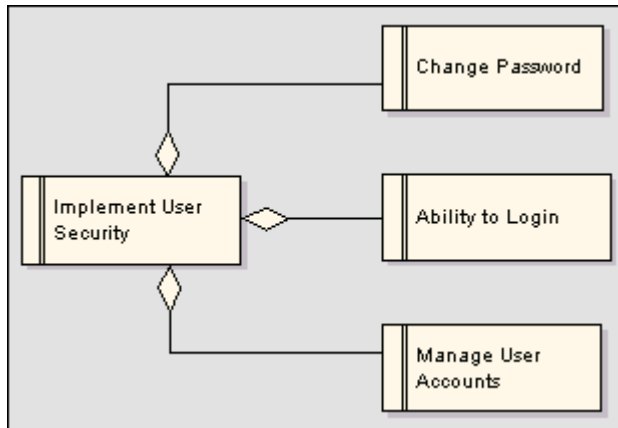
Tip: You may also move internal responsibilities of an element to external requirements - see the section on Moving Internal requirements

6.8.2 External Requirements

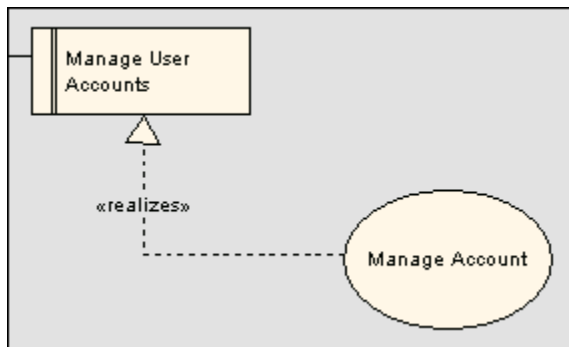
Separate Requirement elements can be manipulated at the diagram level. These correspond to the 'system level requirements' and may be linked using Realization type connections to other model elements which take on the responsibility of implementing the requirement. Requirements at this level have their own properties and are reported on separately in the RTF documentation.

In this context, requirements may also form a hierarchy, as in the example below.

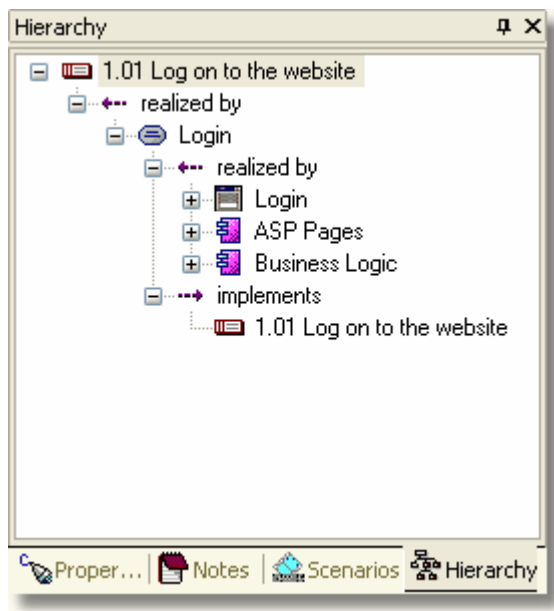
Tip: Using Aggregation, requirements can be linked to show construction of a complete requirements 'tree'.



Implementation is managed using Realization links as in the example below:



Once the links are established, the [Hierarchy window](#) will display the complete requirement implementation / composition details (see example below).

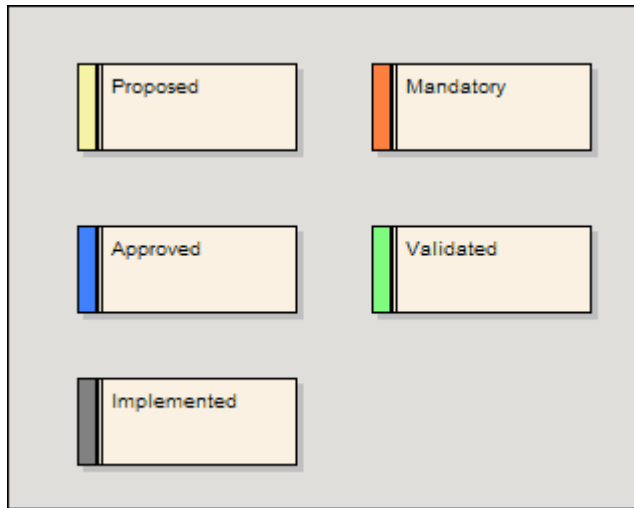


Tip: Use the Relationship Matrix to create and manage the relationships between the requirements - this is a convenient way of quickly building up complex relationships and hierarchies.

6.8.3 Color Coded External Requirements

External requirements may be color coded to enable quick visual cues indicating the status of an requirement. To enable color coded external requirements use the following instructions:

1. From the **Tools** menu select **Options** to bring up the **Local Options** dialog.
2. From the hierarchical tree select **Objects**. Ensure that the **Show color-coded status for requirement elements** checkbox is marked.
3. This will enable the status of external requirements to be represented by color coding.
4. The color code requirements will use the following conventions, yellow for Proposed, blue for Approved, Green for validated, Orange for Mandatory and Black for Implemented.

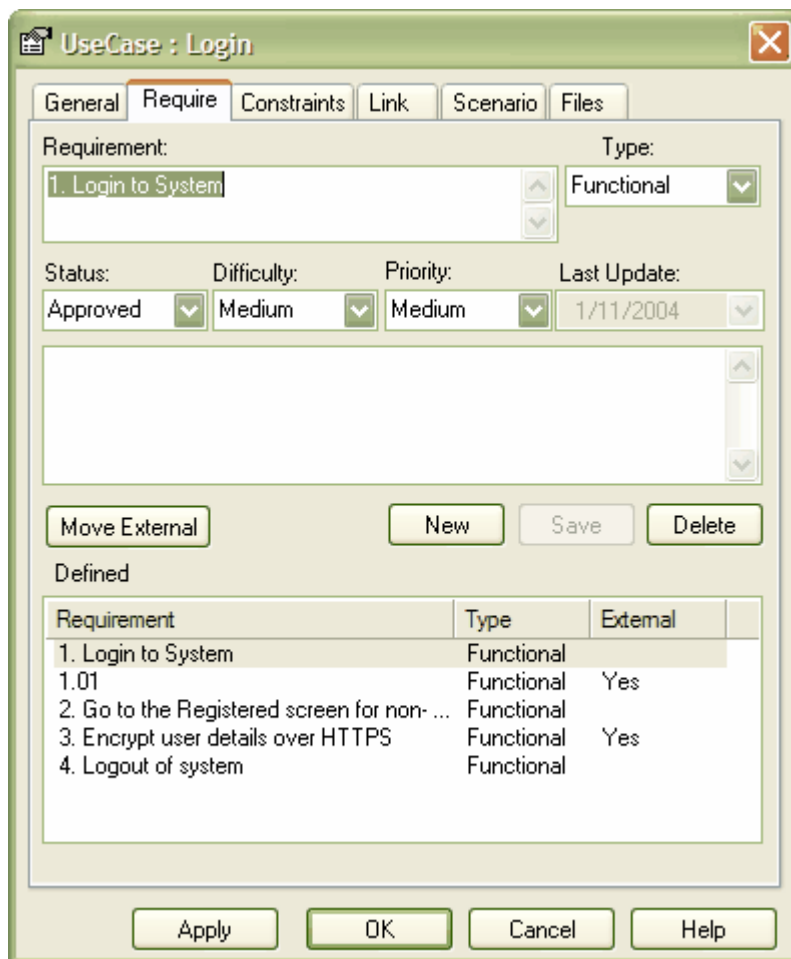


6.8.4 Internal Requirements

Internal requirements in EA are class (or any element) Responsibilities.

In the example below an Internal responsibility to allow the user to login to the system has been defined for the Login use case. This is a *Responsibility* of the Use Case - some action it is responsible for carrying out - and it applies only to this Use Case.

The use case also has connections to external requirements - which are system functions that the Use Case will implement either in full or in part.



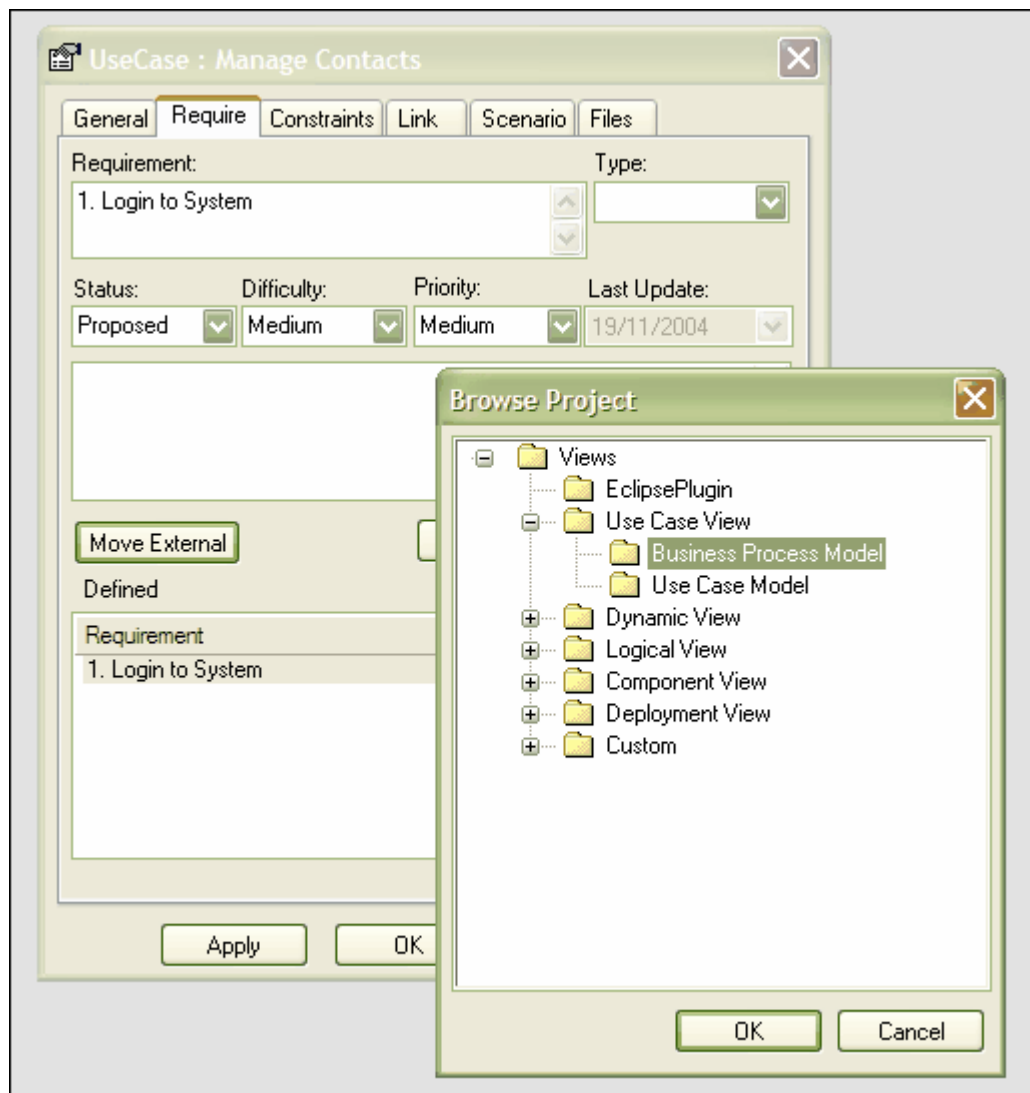
6.8.5 Move Internal Requirements to External Requirement

Elements in EA have internal requirements (what they must do or accomplish). These often overlap or duplicate more formal requirements that the system in general must meet. You can move internal requirements to external requirements in one go using the 'Move External' function.

Move an Internal Requirement to External Requirement

If you have defined an internal requirement for an element and want to move it out (where it will perhaps be implemented by multiple elements) then follow these steps:

1. Open the element properties by double clicking the element in a Diagram, or in the Project Browser.
2. Go to the *Require* tab.
3. Locate and highlight the requirement.
4. Press *Move External*.
5. Select the Package to place the new requirement in.
6. Press *OK*.

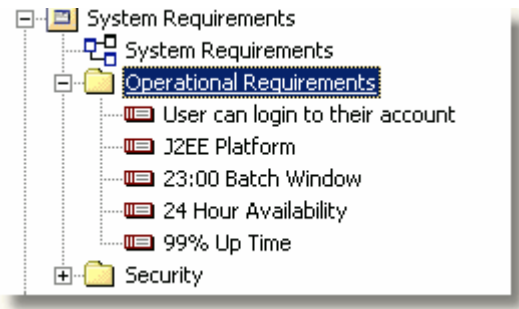


EA will create a new Requirement element in the target package and automatically generate a Realization link from the element to the requirement

Requirement	Type	External
1. Login to system	Functional	
1.01 Log on to the website	Functional	Yes
2. Go to register screen for non-registered users	Functional	
3. Encrypt user details over HTTPS	Functional	Yes
4. Logout of system	Functional	

Notice the requirement is now marked external and the form grayed out. You can double click on the requirement to edit its details.

Also notice that a requirement has been created in the target package.



6.8.6 Requirement Properties

If you double click on a requirement in a diagram or right-click on a requirement in the Project Browser and select *Properties*, you will be able to edit the main properties for the relevant element. Requirement properties are a little different to normal properties - they are mainly focused on the status of the requirement, who owns it and when it was created and/or updated.

Enter a short description to describe the requirement in a sentence - then enter the full requirement details in the notes section at the bottom of the dialog. Set the status, difficulty, priority and other parameters as desired.

When you have completed filling in details for the requirement - press the *OK* button.

Requirement

Properties Files

Short Description: Product Records Must be stored in relational database

Status: Proposed Type: Functional

Difficulty: Medium Phase: 1.0

Priority: Medium Last Update: 23/09/2004

Author: John Redfern Created: 23/09/2004

Version: 1.0

Details:

OK Cancel Help

6.8.7 Composition

Requirements that are linked by aggregation relationships form a composition hierarchy. High level requirements may be composed of lower level requirements, which in turn are made up of finer and more specialized requirements. This hierarchical structure helps manage the complexity of large systems with thousands of requirements and many elements being employed to implement the requirements.

6.8.8 Implementation

Requirements are implemented by model elements - such as Use Cases, Classes, Interfaces, Components, etc. You may specify this relationship in EA using the Realization link. A model element is marked as 'Realizing' a requirement. Once this link exists, EA will display the requirement in the element Responsibilities tab, in the requirement hierarchy tab and in the dependency and implementation reports, as well as the standard RTF output.

A quick method of generating a Realization link is to drag a Requirement element from the Project Browser over an element in a diagram which is to be the implementing element. EA will interpret this as a request to create the realization link and do so automatically. To confirm this, perform the action, and then go to the

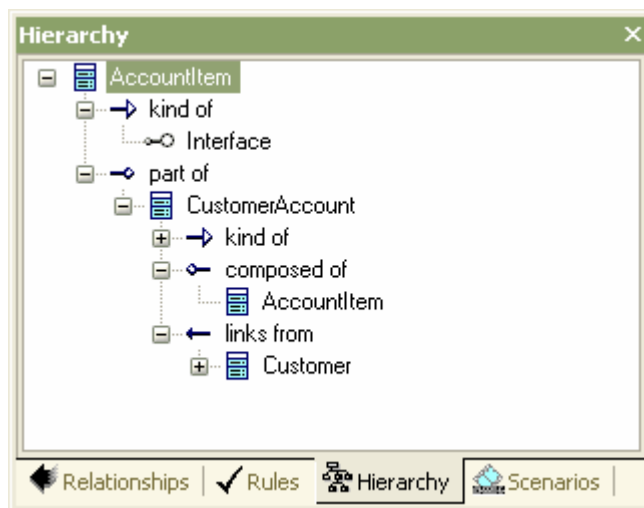
Responsibilities tab page of the target element - there should now be an external relationship to the requirement that was dragged over the target.

6.8.9 Requirements Hierarchy

Requirements may be linked to other elements or other requirements to create a relationship hierarchy.

In general, the aggregation relationship can be used to good effect to show how high level relationships are composed of smaller requirements. This hierarchy is useful in managing the composition of your model and the dependencies between elements and requirements.

The example below shows the hierarchy structure for AccountItem. See the [Hierarchy](#) topic for further information.



6.8.10 The Matrix

The relationship matrix is a spreadsheet like display of relationships between model elements. You select a source package and a target package, the relationship type and direction, and Enterprise Architect will highlight all the relationships between source and target elements by highlighting a grid square.

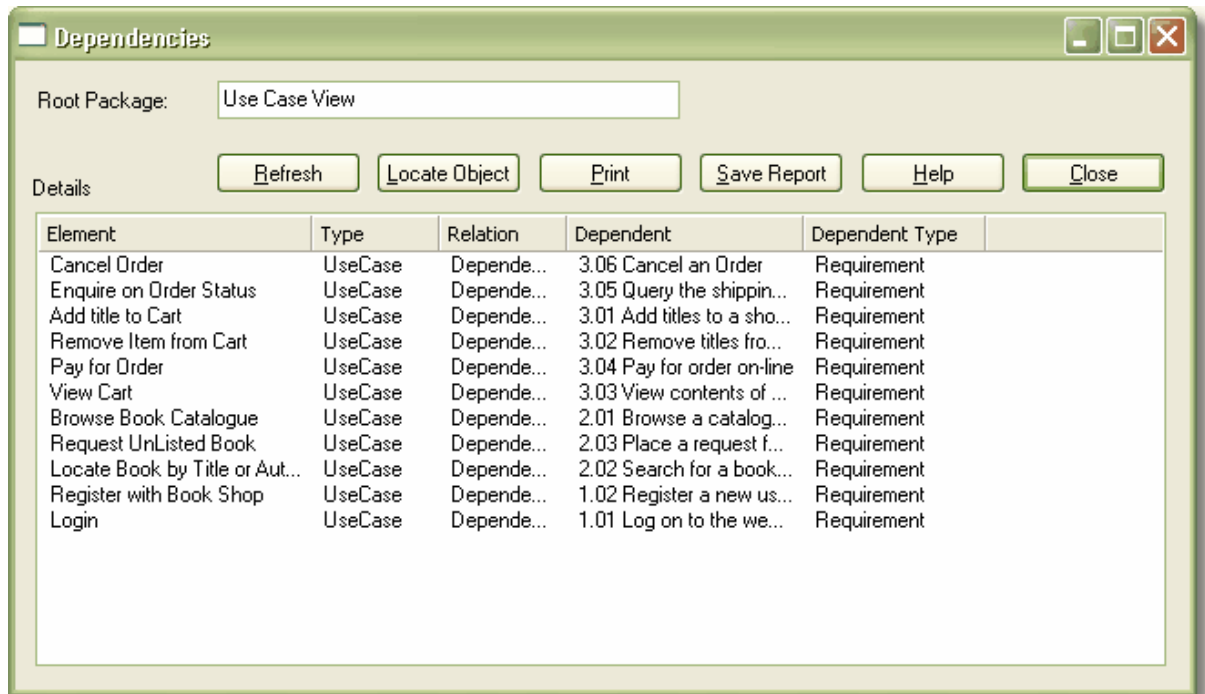
The matrix is a convenient method of visualizing relationships quickly and definitively. It is also possible to delete and create relationships in the Matrix view - another quick way to set up complex sets of element relationships with a minimum of effort.

6.8.11 Dependency Report

To run a dependency report follow these steps:

1. In the Project Browser, select the package you wish to report on (the report includes all sub-folders as well)
2. From the *Project | Documentation* submenu, select *Dependency Details...*

The *Dependencies* dialog displays a list of all elements that implement other elements in the provided list, together with the elements that are dependent. You may print out the results if required.



Control	Description
Root Package	The root package. All elements and packages under this will appear in the report.
Locate Object	This button lets you locate the element selected in the report list in the Project Browser.
Refresh	Run the report again.
Dependency Details	List of dependency details - lists elements in the current hierarchy and elements that implement them.
Print	Print the list.

6.9 Element Relationship Matrix

The relationship matrix allows you to study the relationships between model elements within packages. It also allows you to create, modify and delete relationships between elements with a single mouse click.

Tip: The relationship matrix provides a quick overview of relationships between elements in packages.

Source: Business Model Type: <All> Link Type: Aggregation Profile:
 Target: Business Model Type: <All> Direction: Source -> Target Refresh

	Accounts::Account	Accounts::Message Folder	Accounts::Staff Member	Address Book::Address Book C	Address Book::Address Book	Address Book::Contact	Business Model::Business Proc	Business Model::Domain Model	Business Process Model::Class	Domain Model::Accounts	Domain Model::Address Book	Domain Model::Messages	Messages::Attachment	Messages::Email	Messages::Fax	Messages::Message	Messages::Postal Mail	Messages::SMS
Accounts::Account																		
Accounts::Message Folder	X																	
Accounts::Staff Member																		
Address Book::Address Book...				X	X													
Address Book::Address Book	X																	
Address Book::Contact				X	X													
Business Model::Business Pr...																		
Business Model::Domain Model																		
Business Process Model::Class1																		
Domain Model::Accounts																		
Domain Model::Address Book																		
Domain Model::Messages																		
Messages::Attachment													X					
Messages::Email																		
Messages::Fax																		
Messages::Message		X																
Messages::Postal Mail																		
Messages::SMS																		

6.9.1 Open the Relationship Matrix

To open the Relationship Matrix you can:

- Select *Relationship Matrix* from the *View* menu, or
- Right click on any package in the Project Browser, select *Documentation* | *Open in Relationship Matrix*, and select *As Source* or *As Target*.

Once the Relationship Matrix opens you can:

- [Set the source and target packages](#)
- [Select which element type to show](#)
- [Select link type and direction to show](#)

The matrix will refresh itself after every change you make to the input parameters.

Tip: The matrix will include -ALL- child elements in a hierarchy - sometimes in a large model this can be a lot of elements- possibly too many to be useful. Take care in selecting the source and target package.

6.9.2 Setting Source and Target Package

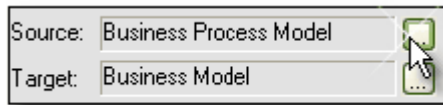
You need to set both the Source and Target packages for the Matrix before relationships can be displayed.

Tip: Set the source and target packages *AFTER* setting the link and element types/details - as EA refreshes the content after each change, this is usually faster.

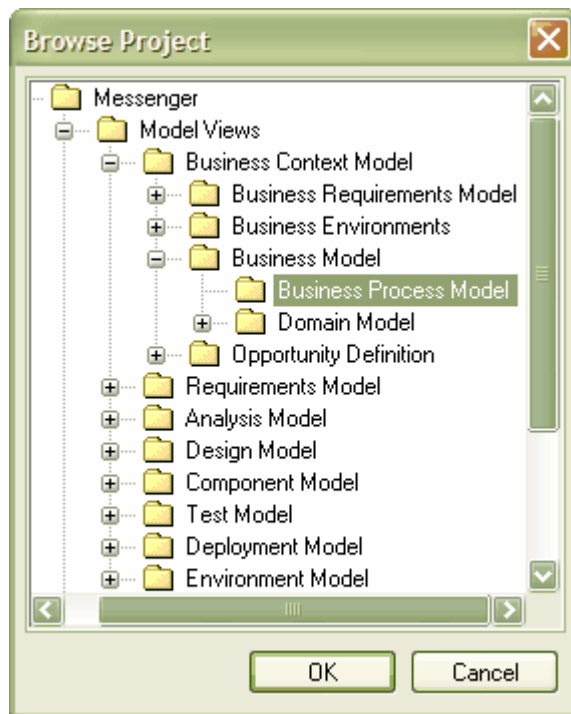
Set the Source or Target Package

To set the source or target package, follow the steps below:

1. Press the **Browse [...]** button at the end of the source or target item.



2. A Project Browser window will pop up - select the required package and press **OK**.



6.9.3 Setting Element Type

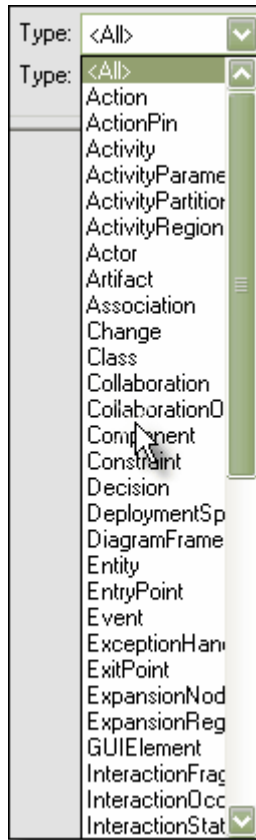
The Relationship Matrix can show *all* element types, or you can specify which type to show.

Set Element Type

To set element type, follow the steps below:

1. Left click on the drop down list of link types for the Source or Target package.
2. Find the required link and select.

EA will refresh the matrix content.



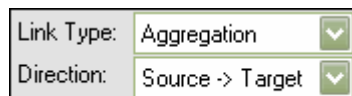
6.9.4 Setting the Link Type and Direction

The matrix requires that you set the link type to report on -and- the link direction.

Set Link Type

To set link type, follow the steps below:

1. Left click on the drop down list of link types.
2. Find the required link and select the appropriate link type, EA will then refresh the matrix content.



Set Link Direction

To set link direction, follow the steps below:

1. Left click on the drop down list of link directions.

2. Select the required type.

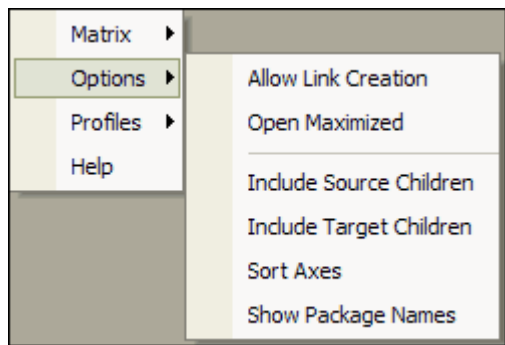
EA will refresh the matrix content.

6.9.5 Matrix Options

You can set some local settings for how the Matrix works from the *Matrix Options* menu:

- *Allow Link Creation* - select to allow access to the right-click link creation behavior
- *Open Maximized* - select to ensure Matrix opens in full screen mode
- *Include Source Children* - select to recursively include child packages and contents under the Source
- *Include Target Children* - select to recursively include child packages and contents under the Target
- *Sort Axes* - when selected, package elements appear in alphabetical order
- *Show Package Names* - hide or show package names in the Matrix, this is useful for shortening the displayed texts, especially in circumstances where packages have long names.

To access the Matrix Options menu right click on the Matrix Menu background to access the Matrix Menu then mouse over the *Options* menu item.



6.9.6 Modifying Relationships in Matrix

Relationships may be modified or deleted, or new relationships may be created, directly from the Relationship Matrix.

To Modify or Delete Relationships

1. Right click on a highlighted relationship to open the context menu (shown below).
2. Select from the following options:
 - *View relationship...* - opens the Property dialog for the selected relationship
 - *Source element properties...* - opens the Property dialog for the source element
 - *Target element properties...* - opens the Property dialog for the target element
 - *Delete relationship*
3. If you have selected *Delete relationship*, you will be prompted to confirm this action

-OR-

If you have selected one of the other options, modify any properties you need to, and Press **OK** to save changes

	Accounts::Account	Accounts::Message Folder	Accounts::Staff Member	Address Book::Address Book G	Address Book::Address Book	Address Book::Contact	Business Context Model::Busin	Business Context Model::Busin	Business Context Model::Busin	Business Context Model::Oppor	Business Environments::Comm	Business Environments::Office
Accounts::Account												
Accounts::Message Folder	X											
Accounts::Staff Member												
Address Book::Address Book...				X								
Address Book::Address Book	X											
Address Book::Contact				X								
Business Context Model::Bus...												
Business Context Model::Bus...												
Business Context Model::Bus...												

View relationship...
 Source element properties...
 Target element properties...
 Delete relationship

To Create a New Relationship

1. Right click on an empty space to open the context menu (shown below).
2. Select *Create new relationship...* to create a new connection between two elements

	Accounts::Account	Accounts::Message Folder	Accounts::Staff Member	Address Book::Address Book C	Address Book::Address Book	Address Book::Contact	Business Context Model::Busin	Business Context Model::Busin	Business Context Model::Busin	Business Context Model::Oppor	Business Environments::Comm	Business Environments::Office
Accounts::Account												
Accounts::Message Folder	X											
Accounts::Staff Member												
Address Book::Address Book...												
Address Book::Address Book	X											
Address Book::Contact				X	X							
Business Context Model::Bus...												
Business Context Model::Bus...												
Business Context Model::Bus...												

Tip: Use the Matrix relationship management features to quickly create and manage relationships like Realization and Aggregation between Requirements and implementation elements (such as Use Cases)

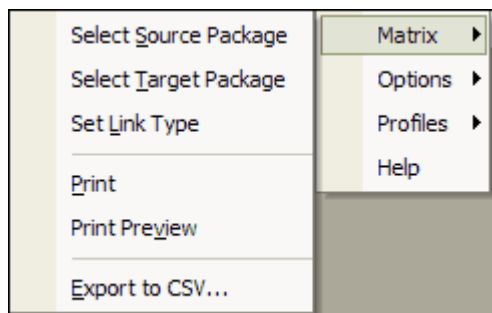
6.9.7 Exporting to CSV

The contents of the relationship matrix may be exported to a CSV file. This provides a convenient mechanism for moving the matrix data to a spreadsheet environment such as Microsoft Excel.

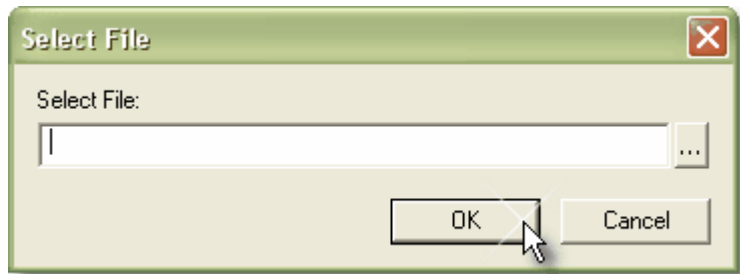
Export to CSV

To export to CSV, follow the steps below:

1. To access the Matrix sub-menu right click on the Matrix Menu background to access the Matrix Menu then select mouse over the Matrix menu item.



2. Select the menu option Export to CSV.
3. Enter a Filename to export to.



4. Press OK to export the data.

6.9.8 Printing the Matrix

You can print a WYSIWYG representation of the matrix using the current printer settings. To access the Matrix sub-menu right click on the Matrix Menu background to access the Matrix Menu then mouse over the Matrix menu item then select the *Print* or *Print Preview* Options.

- To print the matrix select *Matrix | Print*
- To preview prior to printing, select *Matrix | Print Preview*

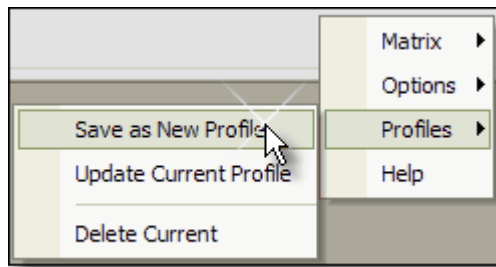
Source: Business Model Type: <All> Link Type: Aggregation Profile:
 Target: Business Model Type: <All> Direction: Source -> Target Refresh

	Accounts::Account	Accounts::Message Folder	Accounts::Staff Member	Address Book::Address Book C	Address Book::Address Book	Address Book::Contact	Business Model::Business Proc	Business Model::Domain Model	Business Process Model::Class	Domain Model::Accounts	Domain Model	Domain Model	Messages::Att	M	M	M	M	M
Accounts::Account																		
Accounts::Message Folder	X																	
Accounts::Staff Member																		
Address Book::Address Book...				X	X													
Address Book::Address Book	X																	
Address Book::Contact				X	X													
Business Model::Business Pr...																		
Business Model::Domain Model																		
Business Process Model::Class1																		
Domain Model::Accounts																		
Domain Model::Address Book																		
Domain Model::Messages																		
Messages::Attachment																X		
Messages::Email																		
Messages::Fax																		
Messages::Message		X																
Messages::Postal Mail																		
Messages::SMS																		

6.9.9 Profiles

You can save a certain Matrix configuration as a named profile for later recall. To do this, follow the steps below:

1. Set up the Matrix the way you want - with source and target, element types and relationship types.
2. To access the *Profiles* sub-menu right click on the Matrix Menu background to access the Matrix Menu, then mouse over the *Profiles* menu item then select the *Save as New Profile* option.
3. Enter a *Name* (a limit of 12 characters applies) and press *OK*.



4. Once you have created one or more Profiles, you can select them from the drop list on the matrix screen.

6.10 Business Modeling

Modeling the Business Process

Modeling the business process is an essential part of any software development process. It allows the analyst to capture the broad outline and procedures that govern what it is a business does. This model provides an overview of where the proposed software system being considered will fit into the organisational structure and daily activities. It may also provide the justification for building the system by capturing the current manual and automated procedures that will be rolled up into a new system, and the associated cost benefit.

As an early model of business activity, it allows the analyst to capture the significant events, inputs, resources and outputs associated with business process. By connecting later design elements (such as Use Cases) back to the business process model through implementation links, it is possible to build up a fully traceable model from the broad process outlines to the functional requirements and eventually to the software artefacts actually being constructed.

As the Business Process Model typically has a broader and more inclusive range than just the software system being considered, it also allows the analyst to clearly map what is in the scope of the proposed system and what will be implemented in other ways (eg. a manual process).

Process Modeling Notation

A business process model typically defines the following elements:

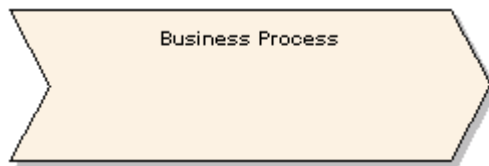
- The Goal or reason for the process;
- Specific inputs;
- Specific outputs;
- Resources consumed;
- Activities that are performed in some order; and
- Events that drive the process.

The business process:

- May affect more than one organisational unit.
- Have a horizontal organisational impact;
- Creates value of some kind for the customer. Customers may be internal or external

The Business Process

A business process is a collection of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how the work is done within an organisation, in contrast to a product's focus on what. A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly defined inputs and outputs: a structure for action. The notation used to depict a business process is illustrated below.

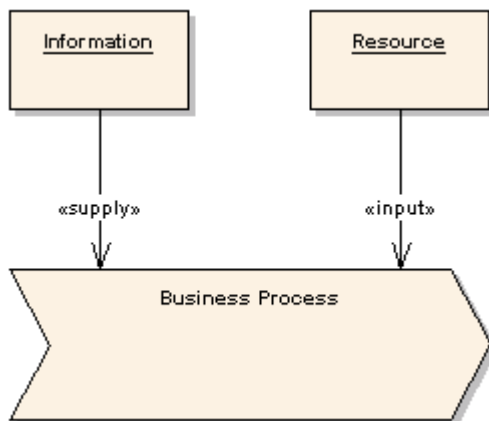


The process notation implies a flow of activities from left to right. Typically an event element is placed to the left of the process and the output to the right. To specifically notate the internal activities, UML activity elements may be placed inside the process element.

Inputs, Resources and Information

Business processes use information to tailor or complete their activities. Information, unlike resources, is not consumed in the process - rather it is used as part of the transformation process. Information may come from external sources, from customers, from internal organisational units and may even be the product of other processes. A resource is an input to a business process, and, unlike information, is typically consumed during the processing. For example, as each daily train service is run and actuals recorded, the service resource is 'used up' as far as the process of recording actual train times is concerned.

The notation to illustrate information and resources is shown below.

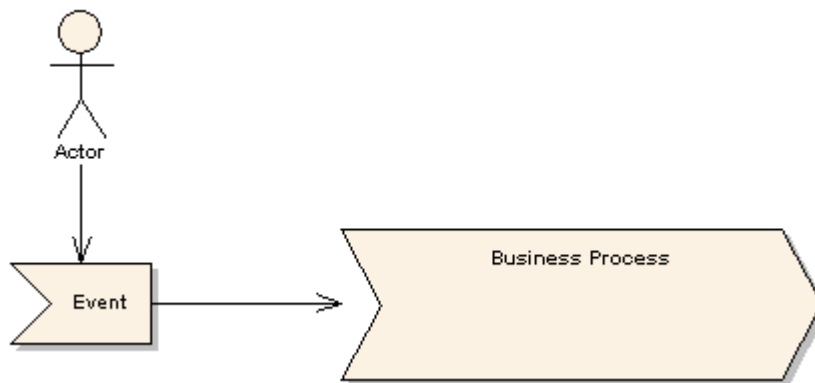


A supply link indicates that the information or object linked to the process is not used up in the processing phase. For example, order templates may be used over and over to provide new orders of a certain style - the templates are not altered or exhausted as part of this activity.

An input link indicates that the attached object or resource is consumed in the processing procedure. As an example, as customer orders are processed they are completed and signed off, and typically are used only once per unique resource (order).

Events

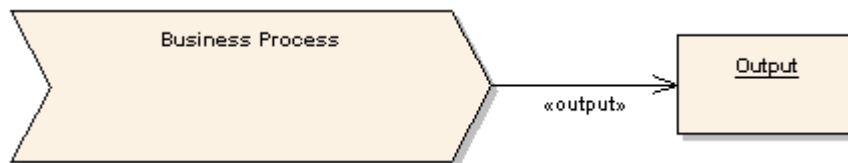
An event is the receipt of some object, a time or date reached, a notification or some other trigger that initiates the business process. The event may be consumed and transformed (for example a customer order) or simply act as a catalyst (e.g. nightly batch job).



Outputs

A business process will typically produce one or more outputs of value to the business, either for internal use or to satisfy external requirements. An output may be a physical object (such as a report or invoice), a transformation of raw resources into a new arrangement (a daily schedule or roster) or an overall business result such as completing a customer order.

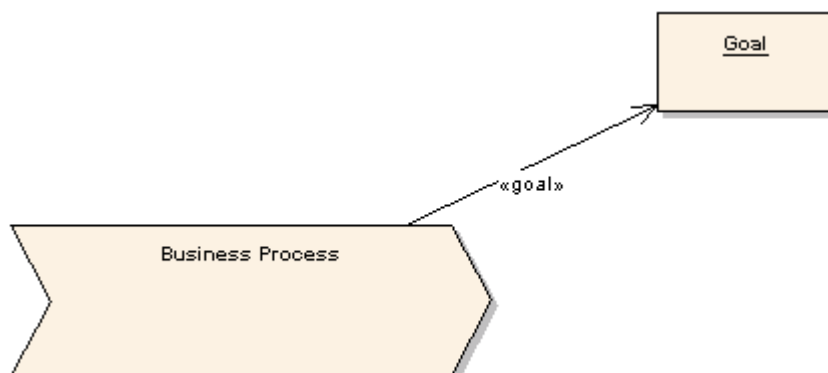
An output of one business process may feed into another process, either as a requested item or a trigger to initiate new activities.



An output link indicates that the business process produces some object (either physical or logical) that is of value to the organisation, either as an externally visible item or as an internal product (possibly feeding another process).

Goals

A business process has some well defined goal. This is the reason the organization does this work, and should be defined in terms of the benefits this process has for the organization as a whole and in satisfying the business needs.

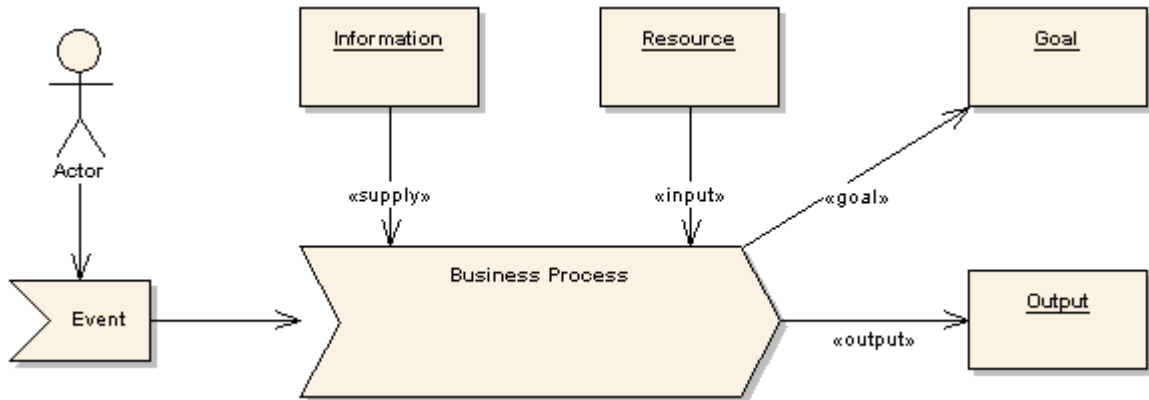


A goal link indicates the attached object to the business process describes the goal of the process. A goal is the business justification for performing the activity.

Putting it together

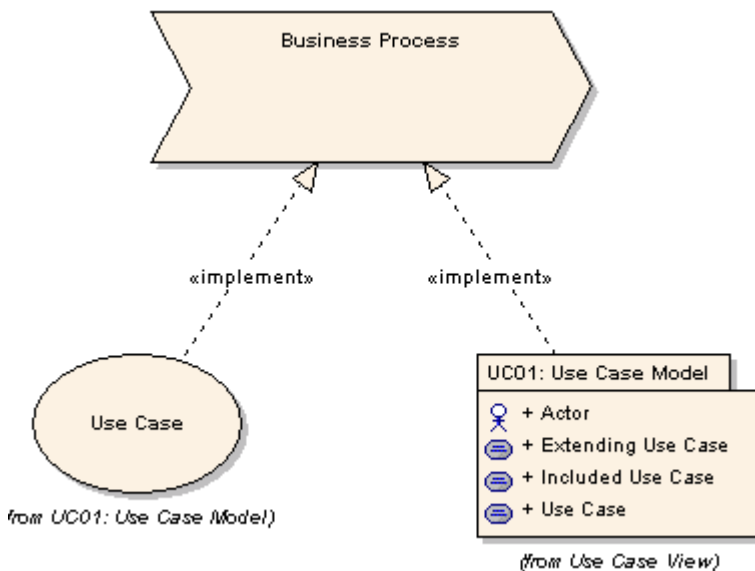
The diagram below illustrates how the various model elements may be grouped together to produce a coherent picture of a named business process. Included are the inputs, outputs, events, goals and other

resources which are of significance.



Traceability

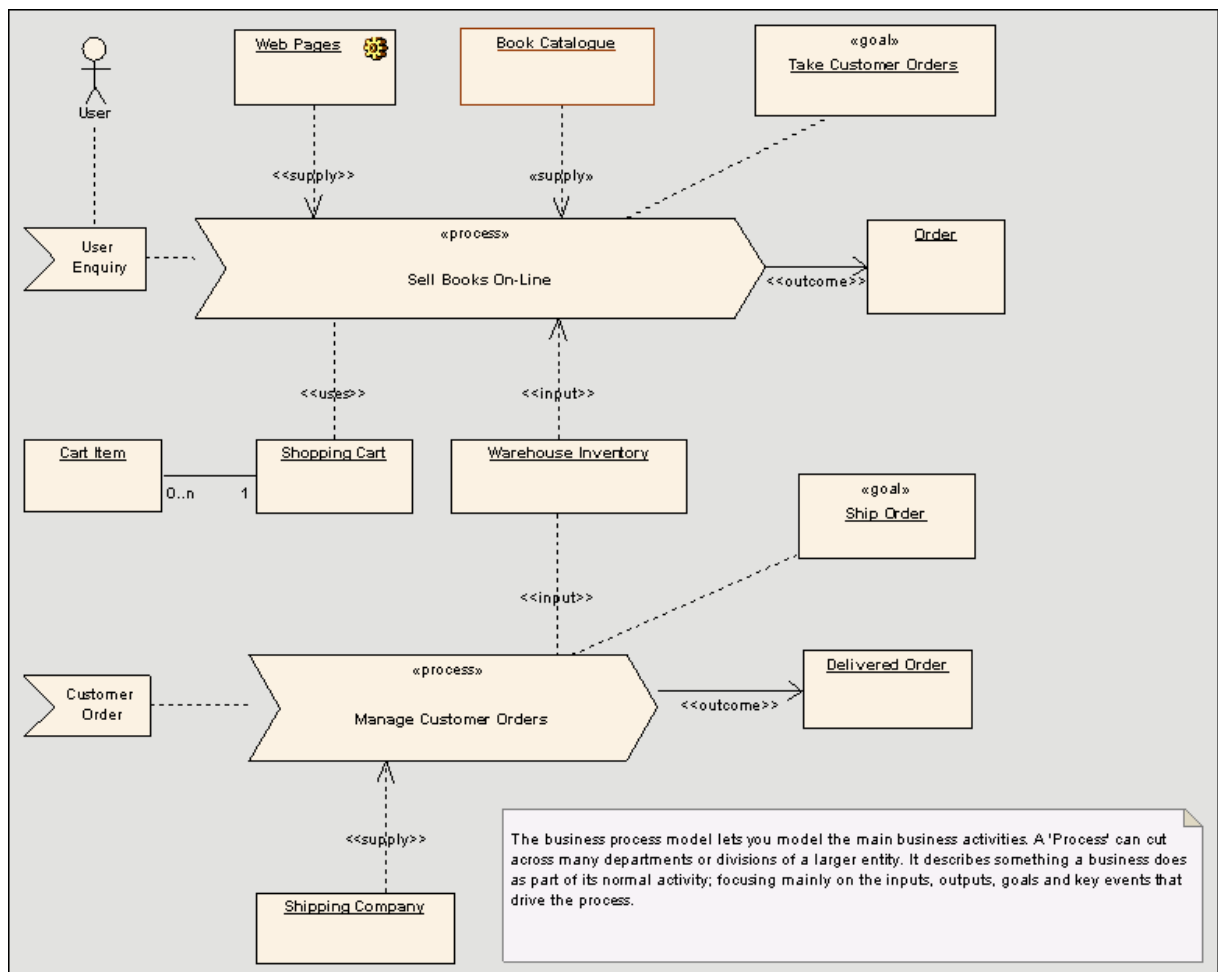
Traceability defines the way a given business process will be implemented in the proposed system. In an implementation diagram, use cases, packages and other model artefacts may be linked back to the business process with <<implementation>> links to signify the dependent relationship. The example below illustrates how a Business Process is implemented by a Use Case and a package. As the model develops and functional software components are built and linked to Use Cases, the business justification for each element can be derived from this model.



Note that this model also implies what is NOT being delivered. The Business Process will typically include a wide range of manual and automated procedures. This model illustrates exactly what functionality (Use Cases) will be built to service a particular business process: any missing functionality must come from other (manual or automated) systems and procedures.

An Example

The example below is an example of the kind of model that may be built up to represent a business process. In this model, the goal of the business process is to take customer orders and to ship those orders out. A user starts the process with an inquiry, which leads to the involvement of the Book Catalogue, Shopping Cart, On-line pages and warehouse inventory. The output of significance to the business is a customer order.



The second half of the process model is to respond to a customer order and ship the required items. The second process involves the warehouse inventory, shipping company and completes when an order is delivered to the customer.

Part



7 Model Management

The Model Management section covers the following topics:

Model Files

An Enterprise Architect model is stored in a *data repository*. EA allows you to work with [.EAP files](#) and [DBMS repositories](#). A [.EAP file](#) is a Microsoft JET database. You can also work with [SQL Server](#), [MySQL](#) and [Oracle9i](#) data repositories in EA Corporate edition. Information on how to get started with models can be found in the [Create/Open Model Files](#) section.

Replication

Note: *This functionality is available in the Professional and Corporate editions only. The Desktop edition is intended for single users, so does not support replication.*

In addition to sharing projects in real time over a network, EA also allows for projects to be shared using [replication](#). Replication is a powerful means of distributing a single .EAP model over a wide area, with occasional synchronization between Design Master and Replica.

Project Sharing

Note: *This functionality is available in the Professional and Corporate editions only. The Desktop edition is intended for single users, so does not support shared files.*

Enterprise Architect allows [project sharing](#) for efficient management of team development. You can create a replica of your project, make changes to it, then merge your changes back into the master project.

User Security

Note: *This feature is available in the Corporate edition only.*

[User security](#) provides a means of limiting access to update functions in a model. Elements may be locked on a per user or per group basis and a password required to login.

Data Transfer

Note: *This feature is available in the Corporate edition only.*

Enterprise Architect provides functionality to perform [data transfer](#) between data repositories, row by row, table by table.

See:

- [Create and Open Model Files](#)
- [Upgrading Models](#)
- [Data and Model Integrity](#)
- [Data Transfer](#)
- [Upsizing Models](#)
- [Model Maintenance](#)
- [Manage Views](#)
- [Import and Export](#)
- [Version Control Options](#)
- [Team Development](#)
- [Spell Checking](#)
- [Reference Data](#)

7.1 Create and Open Model Files

An Enterprise Architect model is stored in a *data repository*. In EA Desktop and Professional editions, you can work with a single file with a *.EAP* extension is used for model files used. A suitable *DBMS* database can be used for model files in EA Corporate edition.

Model Files

.EAP Files

A single file with a *.EAP* extension is used to store models in EA Desktop and Professional editions. An *.EAP* file is a Microsoft JET database, so can also be opened using MS Access or any other reporting tool that can work with JET databases.

DBMS Repositories

A suitable *DBMS* database can be used for model files in EA Corporate edition. *DBMS* model files have the same logical structure as *.EAP* model files, but must be connected to using ADO/ODBC. EA currently supports the following data repositories:

- MS Access (in all editions - *.EAP* files are stored in Microsoft JET databases as mentioned above)
- [SQL Server](#)
- [MySQL](#)
- [Oracle9i](#)
- [PostgreSQL](#)
- [MSDE](#)
- [Adaptive Server Anywhere](#)

Create a New Project File

To [create a new project](#) in Enterprise Architect, you need to use an existing project model. There is a blank model file supplied when you install Enterprise Architect - this is called [EABase.EAP](#). You can also use an existing project as a template to create an existing project. If you [plan your projects wisely](#), this can be a very efficient approach.

Open an Existing Project

There are various ways to [open existing projects](#) 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 allows you to connect to a data repository. EA currently supports the following data repositories:

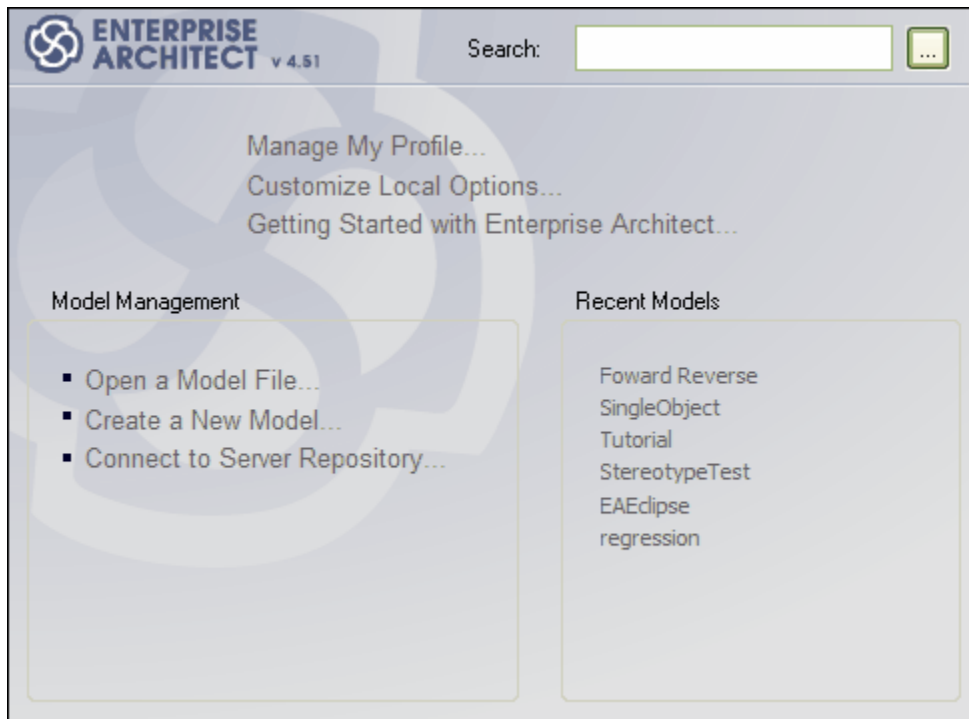
- MS Access (in all editions - *.EAP* files are stored in Microsoft JET databases)
- [SQL Server](#)
- [MySQL](#)
- [Oracle9i](#)
- [PostgreSQL](#)
- [MSDE](#)
- [Adaptive Server Anywhere](#)

To create a new data repository, you first need to create a new database with the *DBMS* management software, then run supplied scripts to create the logical structure. You should then use EA data transfer functions to move a model from an *.EAP* or *DBMS* model into the new model.

7.1.1 Open Project Dialog

Whenever you launch Enterprise Architect, the first thing you will see is the [Start Page](#) (shown below). From here, you can [create a new project](#), [open an existing project](#) and [connect to a data repository](#) (*Corporate*

Edition only).



Element	Description
Open a Model File	Displays the Open an Existing Project dialog
Recent Models	Contains a list of the most recently used EA projects. Click on the project you wish to open.
Create a New Model	Opens the Create New Enterprise Architect Project dialog.
Connect to Server Repository	Allows you to specify a Data Source name to connect to. MySQL , SQL Server , Oracle9i , PostgreSQL , MSDE and Adaptive Server Anywhere repositories are supported. Note: This feature is available in the Corporate edition only.

7.1.2 Open an Existing Project

To open an existing model

There are a number of ways in EA to open an existing model. From the [Start Page](#),

- Click on **Open a Model File**.
- The **Select Enterprise Architect Project to Open** dialog will appear.
- Use the file browser to navigate to the project (*.EAP) you wish to open. Select the project and click **Open**.

Recently Opened Projects

Enterprise Architect keeps a list of recently opened projects and displays them on the Start Page for easy selection. If the project you wish to open is in the Recently Opened Projects list, simply click once on the name of the project to open it.

Note: If you already have a project open, you will be prompted to save changes before loading.

EA Example Project File

New Enterprise Architect users in particular should start by exploring the EAExample file supplied with Enterprise Architect. The example model file is stored in your EA installation directory. The default installation directories, depending on which version you have installed, are:

- Registered version: C:\Program Files\Sparx Systems\EA
- Trial version: C:\Program Files\Sparx Systems\EA Trial
- Lite version: C:\Program Files\Sparx Systems\EA Lite

Connecting to a Data Repository

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

If you are using EA Corporate edition, you also have the option to connect to [SQL Server](#), [MySQL](#) and [Oracle9i](#) data repositories.

7.1.3 Create a Model

When you create a new Enterprise Architect project, you must do so from an *existing project model*. Select **Create new Model** from the Start Page to invoke the [Create New Enterprise Architect Project](#) dialog. The names of the new model and the base model can be entered in this dialog.

The EABase Model File

The default model file - EABase.EAP - is supplied when you install Enterprise Architect. By default, the example model file is stored in your EA installation directory. The default installation directories, depending on which version you have installed, are:

- Registered version: C:\Program Files\Sparx Systems\EA
- Trial version: C:\Program Files\Sparx Systems\EA Trial
- Lite version: C:\Program Files\Sparx Systems\EA Lite

Designing a Custom Template

You can customise any Enterprise Architect project and use it as the base for the new project. This allows you or your organization to build a default project with company standards, tutorials, frameworks or any other common piece of modeling already in-built. A model project is no different to an ordinary project - Enterprise Architect simply copies and renames it as a starter for your new project. With careful planning you can save yourself many hours of work at project start-up.

7.1.3.1 Create a New Model File

To create a new model file

From the [Start Page](#), select **Create a new Model File...**

The **Create a New Model** dialog will appear (below). To create a new Enterprise Architect project, you need to select a project template that will form the base model for the new project. When you install Enterprise Architect a default model is installed called EABase.EAP

To create a new project, first select a model project (EABase.eap by default), then enter the name and directory of the new project that will be created in the space provided.

New Project:

Model Project:

Reset New Project GUIDs

- Use the *New Project Browse...* option to select where to save your project. If this is to be a shared project, place the file on a shared network resource (eg. Network Server or Workgroup Server).
- Use the *Model Project Browse...* option to select the [base model for your project](#). EABase.EAP is the default - but you can select any existing model file you wish (see [Designing a Custom Template](#)).

When you have entered the filenames, press *Create Project* to create your project. The *Cancel* button closes the dialog without creating a new project.

Tip: You can copy any Enterprise Architect project using Windows Explorer, and open the copied project as a new project.

7.1.4 Setting Up a Database Repository

Introduction

The Corporate Edition of EA allows you to work with database repositories rather than use the standard .EAP files. Available repositories are **SQL Server**, **MySQL**, **Oracle9i**, **PostgreSQL**, **MSDE** and **Adaptive Server Anywhere**. Setting up a database repository is a two or three step process: Firstly, you set up an ODBC driver for your database; secondly, you create the repository tables using scripts downloaded from the Sparx Systems web site; and finally, you connect to the repository. Full instructions on all three steps are given below.

Setting Up an ODBC Driver

Setting up an ODBC driver is only necessary for **MySQL**, **PostgreSQL** and **Adaptive Server Anywhere**. The other supported databases connect using OLE DB, so this step can be skipped. To find out how to set up an ODBC driver, go to:

- [Setup a MySQL ODBC Driver.](#)
- [Setup a PostgreSQL ODBC Driver.](#)
- [Setup an Adaptive Server Anywhere ODBC Driver.](#)

Creating a Repository

To find out how to download the scripts and create the data repository tables, go to:

- [Connect to a MySQL Data Repository.](#)
- [Connect to a SQL Server Data Repository.](#)
- [Connect to a Oracle9i Data Repository.](#)
- [Connect to a PostgreSQL Data Repository.](#)
- [Connect to an Adaptive Server Anywhere Data Repository.](#)
- [Connect to a MSDE Server Data Repository.](#)

Connecting to a Repository

Once the repository is created, you can connect to it. To find out how, go to:

- [Connect to a MySQL Data Repository.](#)
- [Connect to a SQL Server Data Repository.](#)
- [Connect to an Oracle9i Data Repository.](#)
- [Connect to a PostgreSQL Data Repository.](#)
- [Connect to an Adaptive Server Anywhere Data Repository.](#)
- [Connect to a MSDE Server Data Repository.](#)

7.1.4.1 Setting Up an ODBC Driver

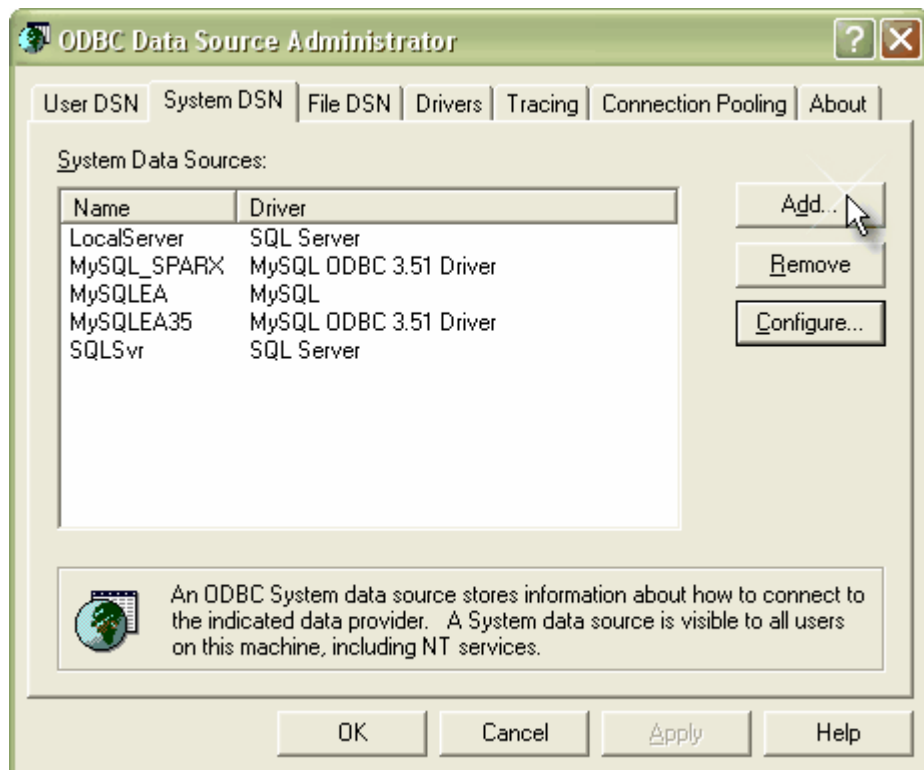
This section details how to set up the following ODBC drivers:

- [MySQL ODBC Driver](#)
- [PostgreSQL ODBC Driver](#)
- [Adaptive Server Anywhere ODBC Driver](#)

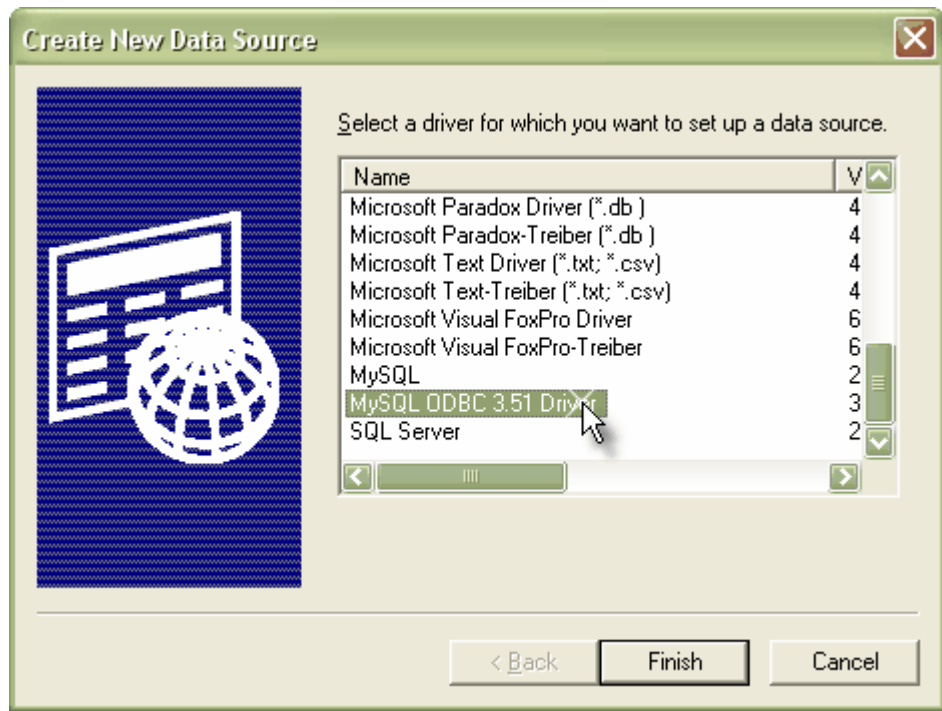
7.1.4.1.1 Setup a MySQL ODBC Driver

Before you can connect to a MySQL data repository, you must first set up a MySQL ODBC driver. To do this, you will require Microsoft MDAC components, MySQL DBMS system and MySQL ODBC driver (version 3.51 minimum). Follow the steps below to set up your MySQL ODBC Driver:

1. Open the *Control Panel | Administrative Tools | Data Sources (ODBC)* window (below).

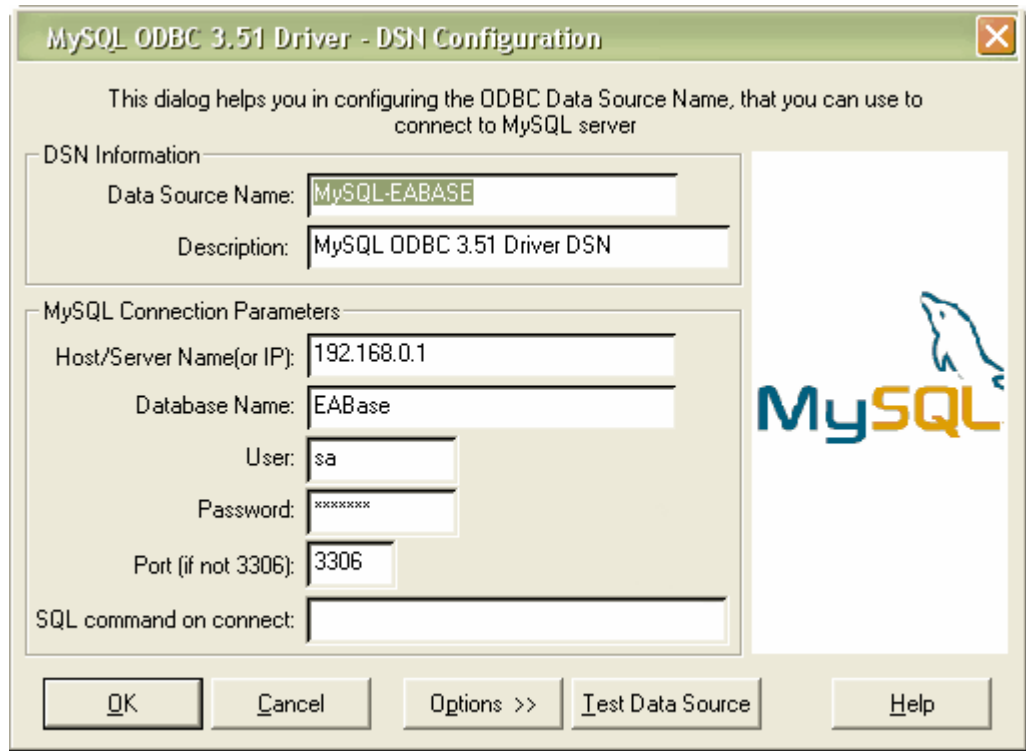


2. Press *Add* - this will bring up the dialog below, which allows you to add a new DSN.

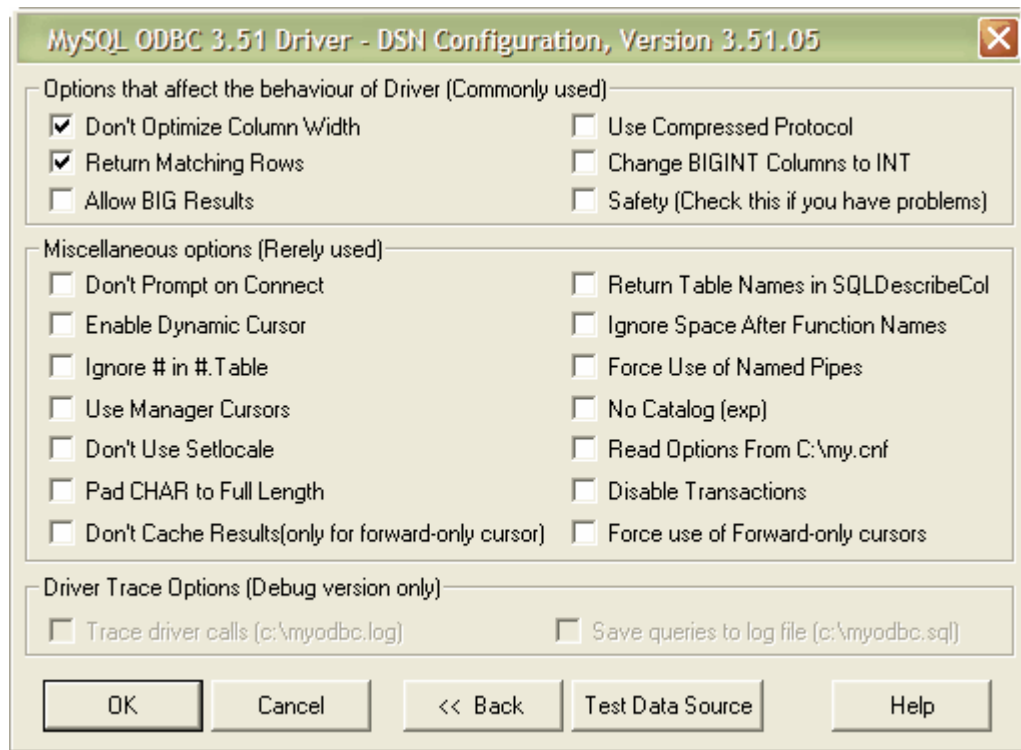


3. Select "MySQL ODBC 3.51 Driver" from the list.
4. Press **Finish**. This will open the DSN Configuration dialog.
5. Enter the following configuration details:
 - A name for the connection
 - Description (optional)
 - The host address of the DBMS server
 - The Database Name on the selected server
 - User name and Password

See the example below:



6. Press the *Options* button to set the advanced options.
7. Ensure that the *Don't Optimize Column Width* and *Return Matching Rows* items are checked, then press the *OK* button.



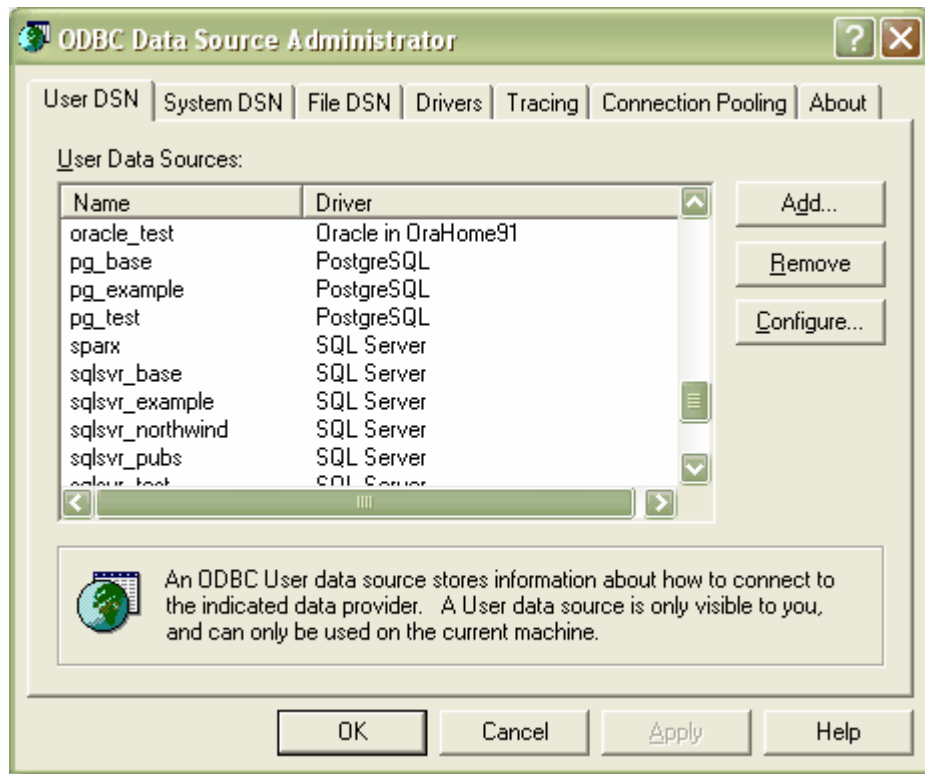
8. Press **Test Data Source** to confirm that the details are correct.
9. If the test succeeds, press **OK** to complete the configuration.
10. If the test does not succeed, review your settings.

Your MySQL connection is now available to use in Enterprise Architect.

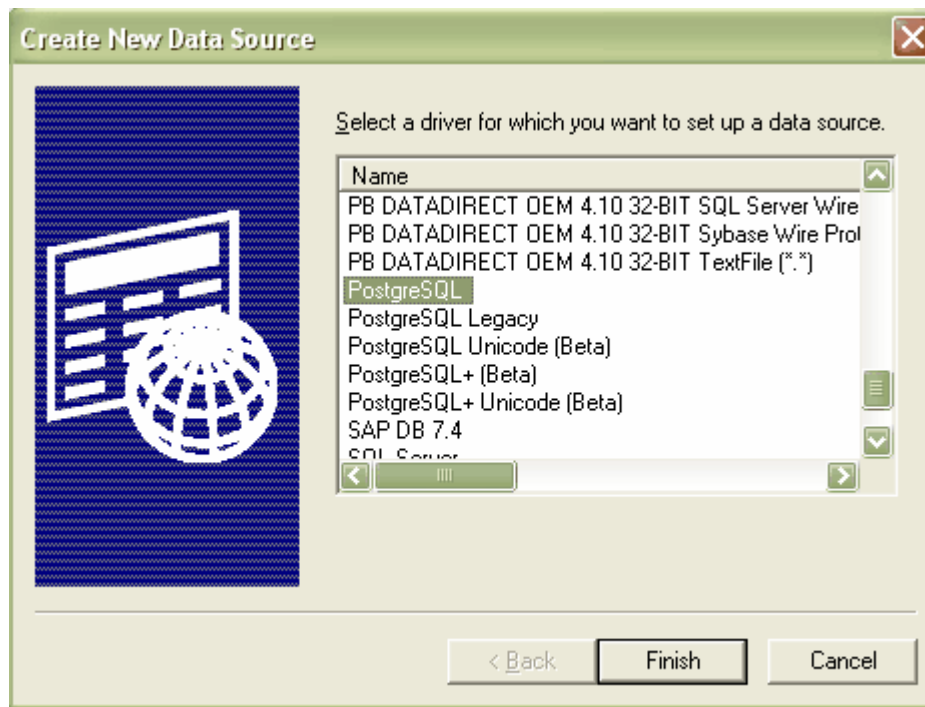
7.1.4.1.2 Setup a PostgreSQL ODBC Driver

Before you can connect to a PostgreSQL data repository, you must first set up a PostgreSQL ODBC driver. To do this, you will require Microsoft MDAC components, PostgreSQL DBMS system and PostgreSQL ODBC driver (version 7.03.01.00 minimum). Follow the steps below to set up your PostgreSQL ODBC Driver:

1. Open the **Control Panel | Administrative Tools | Data Sources (ODBC)** window (below).

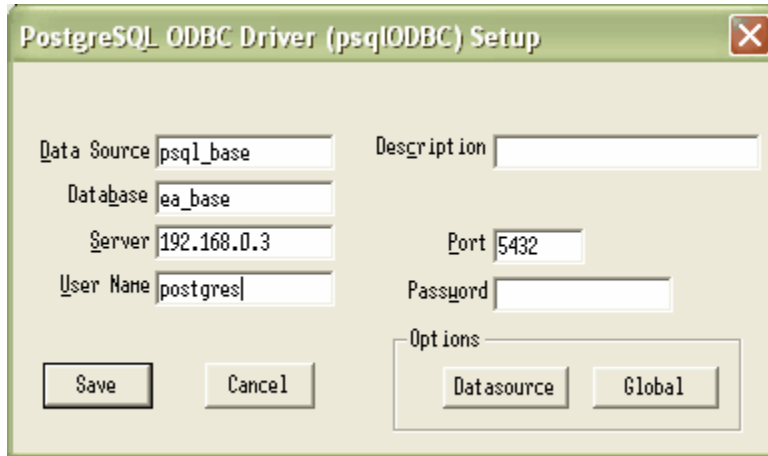


2. Press the **Add** button - this will bring up the dialog below, which allows you to add a new DSN.

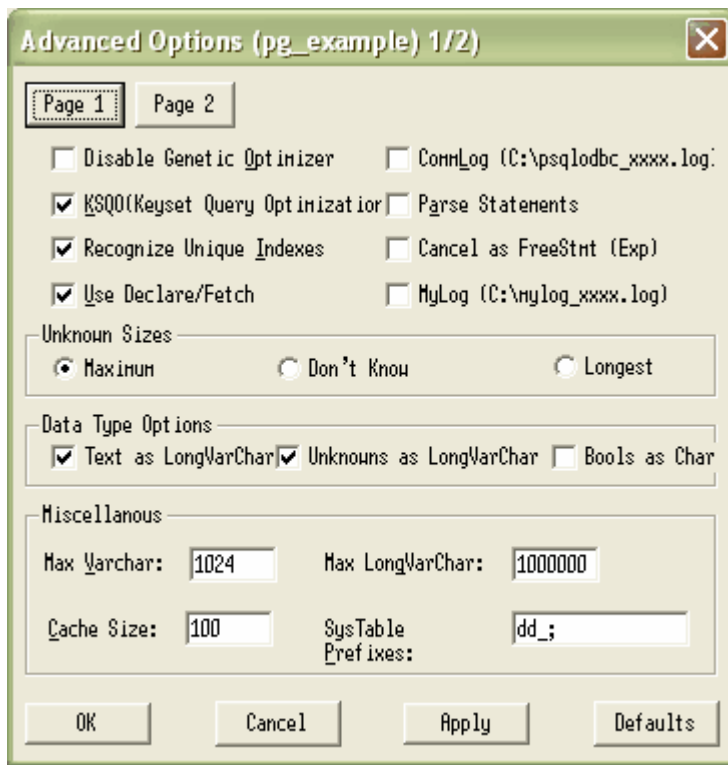


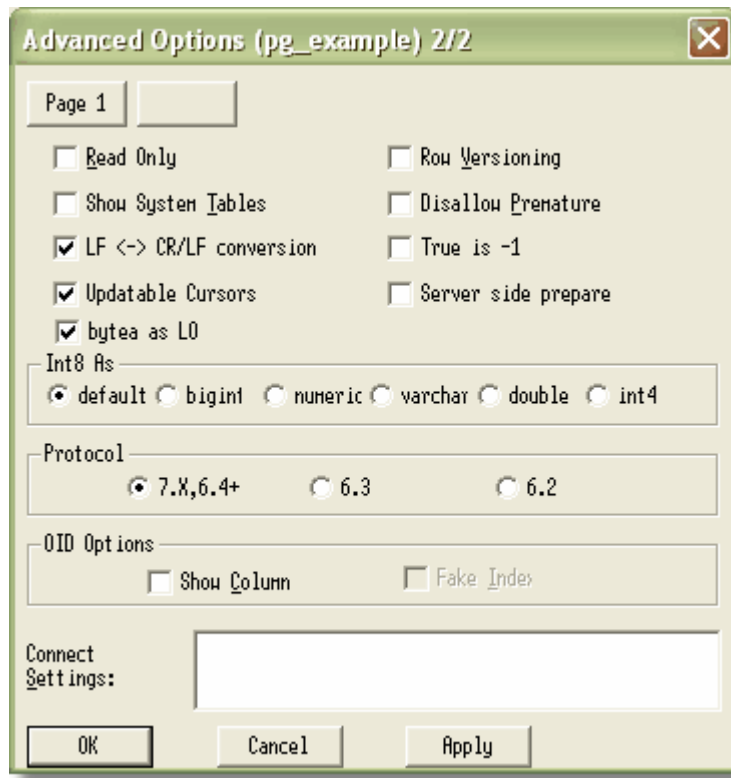
3. Select "PostgreSQL" from the list.

4. Click the *Finish* button.
5. Enter the following configuration details:
 - A name for the connection
 - The actual name of the database.
 - Description (optional)
 - The host address of the PostgreSQL server.
 - User name and password.



6. Click the *Data Source* button and set the options on Pages 1 and 2 as shown on the examples below:





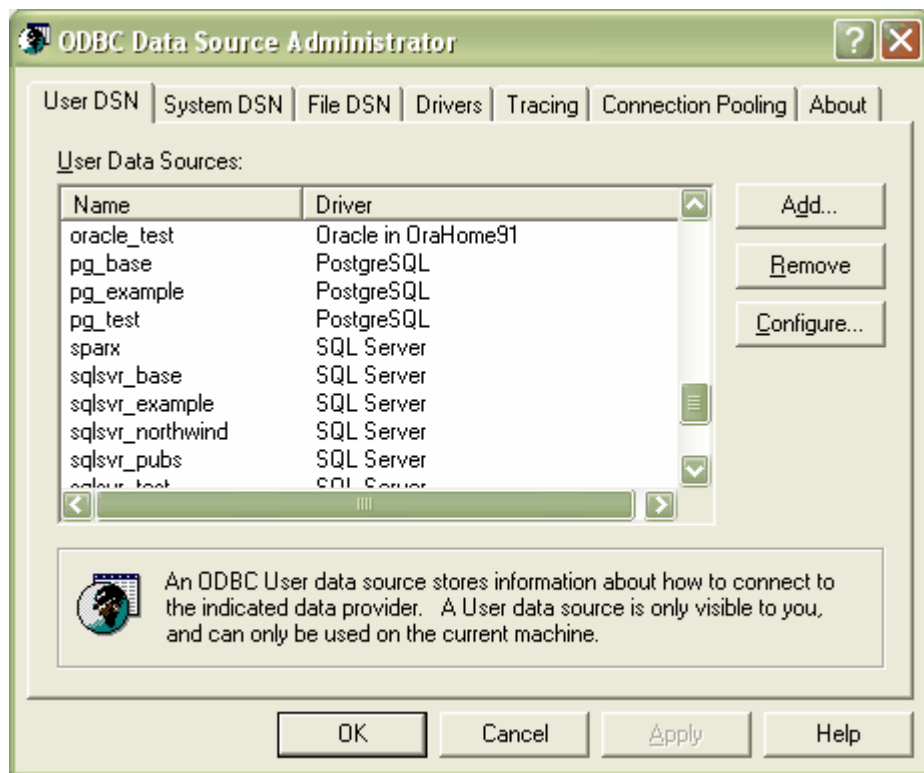
7. Press **OK** to complete the configuration.

Your PostgreSQL connection is now available to use in Enterprise Architect.

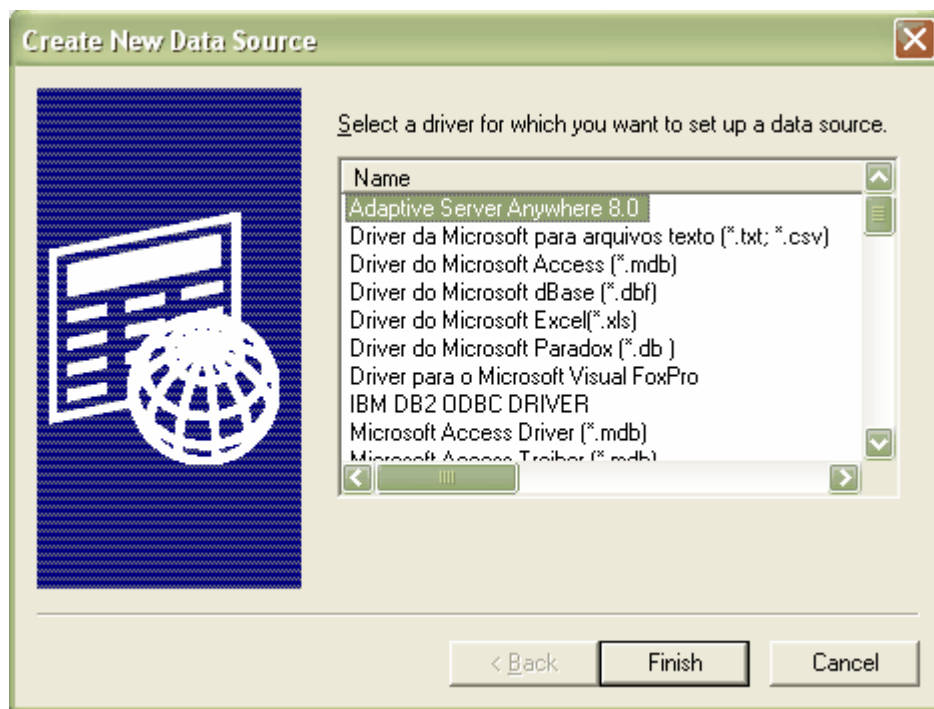
7.1.4.1.3 Setup an Adaptive Server Anywhere ODBC Driver

Before you can connect to a ASA data repository, you must first set up a ASA ODBC driver. To do this, you will require Microsoft MDAC components, the ASA DBMS system and the ASA ODBC driver (installed with the ASA DBMS). Follow the steps below to set up your ASA ODBC Driver:

1. Open the **Control Panel | Administrative Tools | Data Sources (ODBC)** window (below).



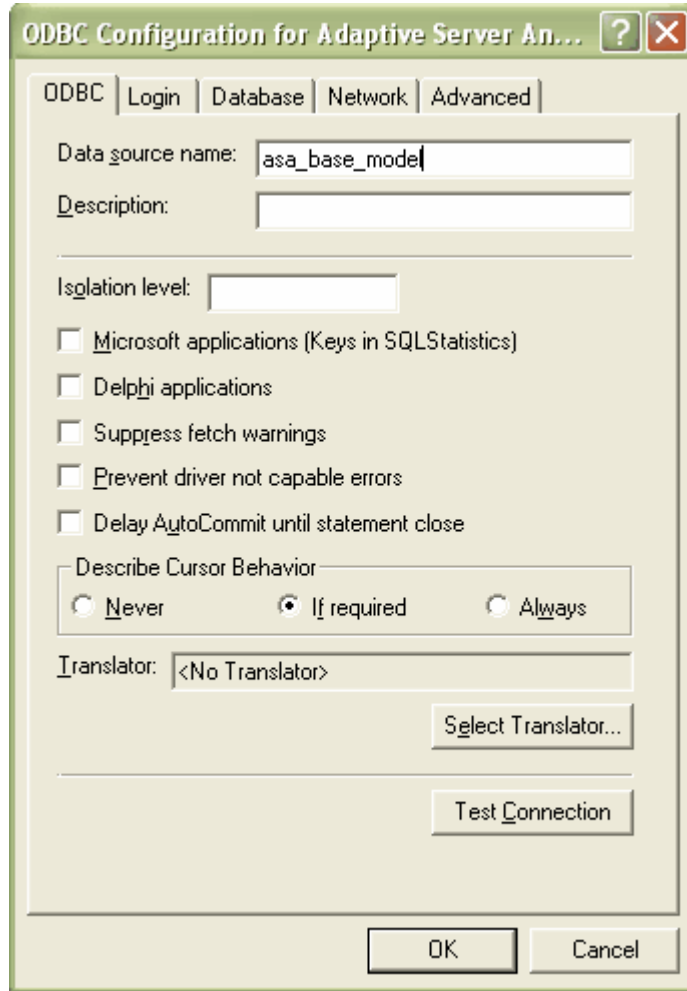
2. Press the **Add** button - this will bring up the dialog below, which allows you to add a new DSN.



3. Select "Adaptive Server Anywhere 8.0" from the list.

4. Click the *Finish* button.
5. Enter the following configuration details:

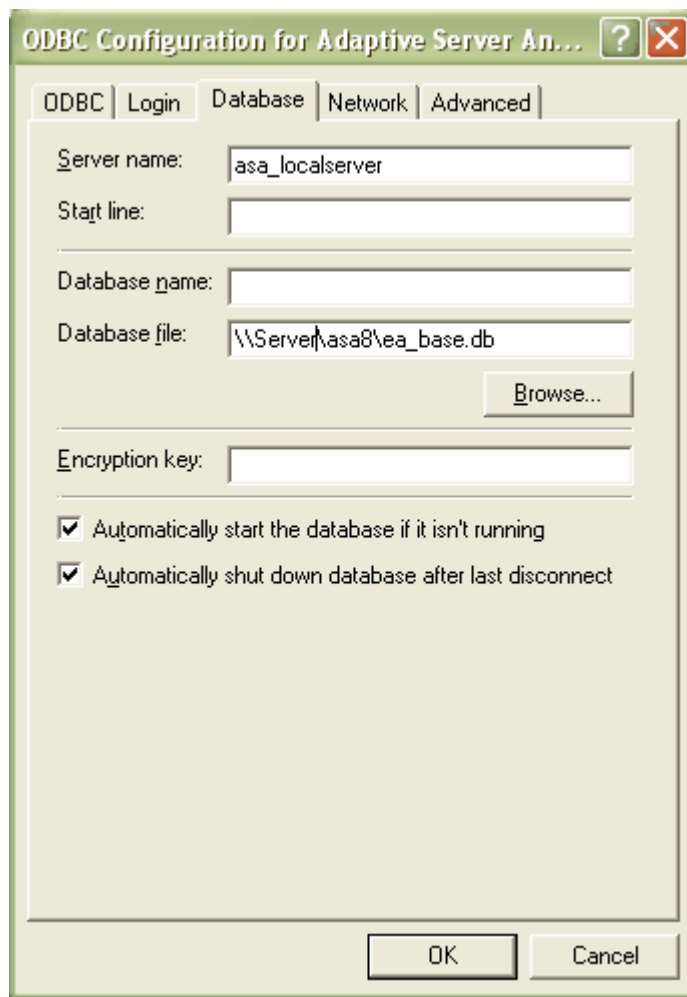
A name for the connection on the ODBC tab.



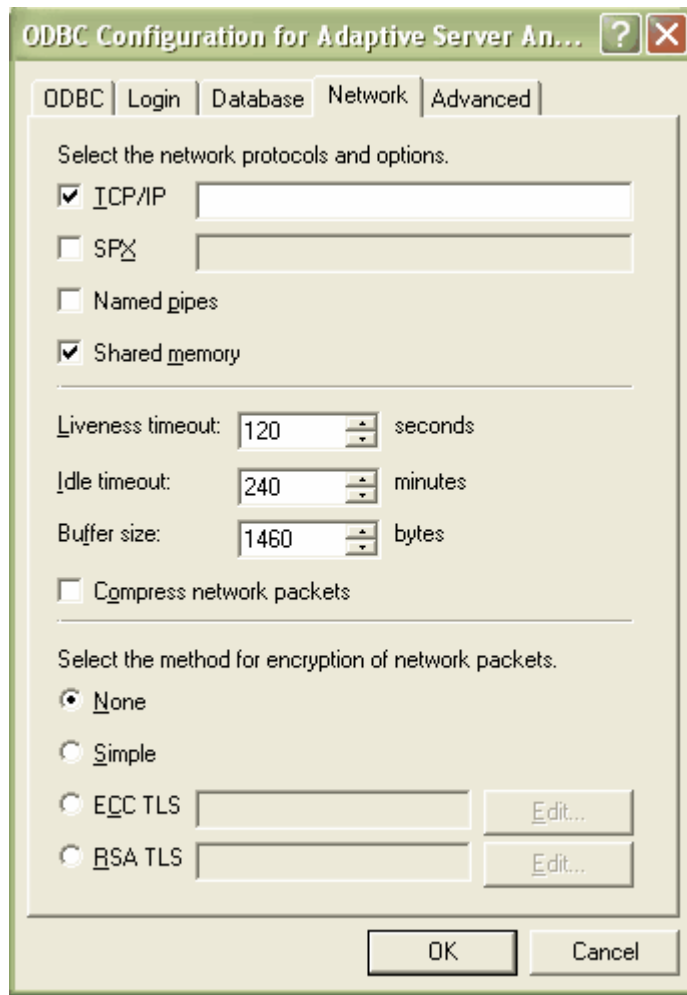
The username and password on the Login tab (dba, sql - are the defaults when ASA is installed).



The server name and the path to the database on the Database tab.



The network protocol on the Network tab (if the database is on a network location).



6. You can now return to the ODBC tab and test the connection.
7. Press **OK** to complete the configuration.

Your Adaptive Server Anywhere connection is now available to use in Enterprise Architect.

7.1.4.2 Connecting to a Data Repository

This section details how to connect to the following data repositories:

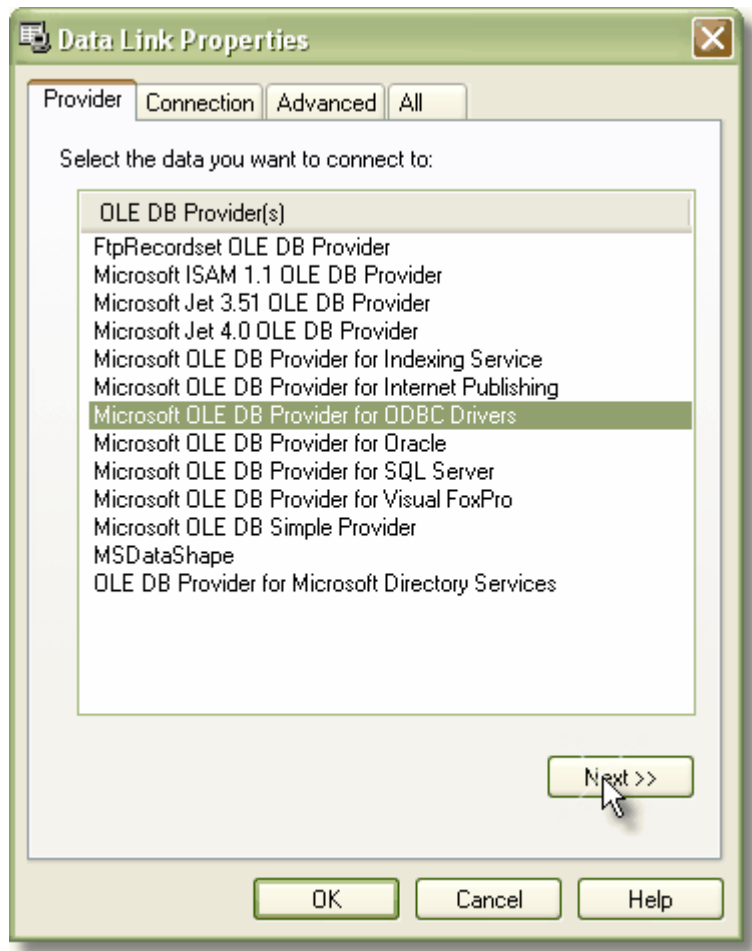
- [MySQL Data Repository](#)
- [SQL Server Data Repository](#)
- [Oracle9i Data Repository](#)
- [PostgreSQL Data Repository](#)
- [Adaptive Server Anywhere Data Repository](#)
- [MSDE Server Data Repository](#)

7.1.4.2.1 Connect to a MySQL Data Repository (Corporate Edition only)

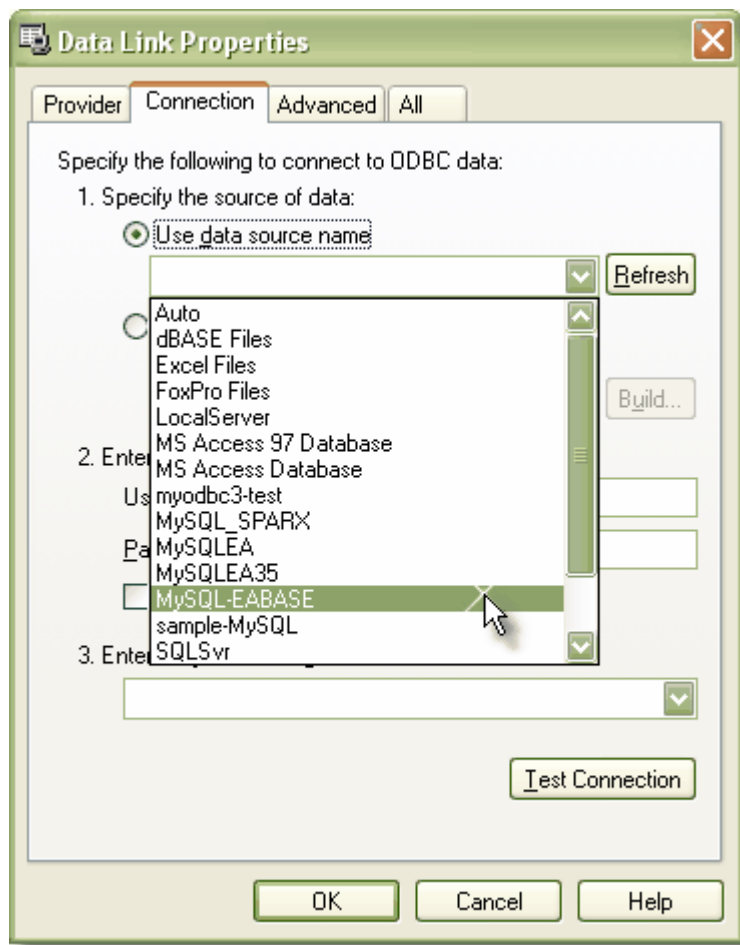
Note: This feature is available in the Corporate Edition only.

In order to use a MySQL data repository, you must connect to it in EA first. Before connecting to the repository, you will need to have [set up a MySQL ODBC driver](#). Be aware, there are some [limitations with using MySQL](#) with EA. Follow these steps to connect to a MySQL data repository in EA:

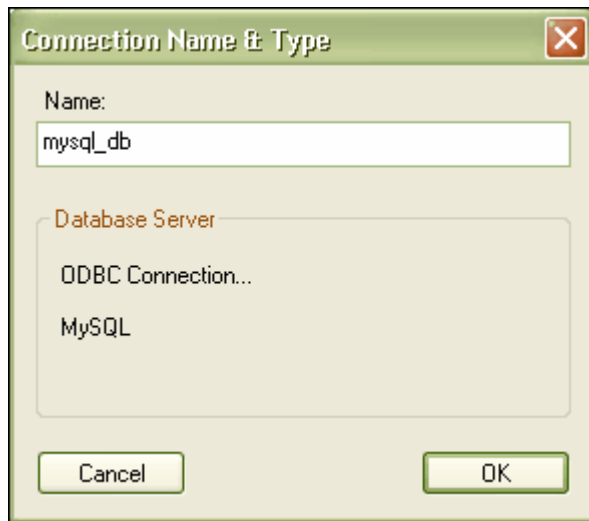
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press **Next**. This will take you to the **Connection** tab (below).



5. Select the ODBC driver you have set up to connect to your MySQL repository from the *Use data source name* drop down list. In the [setup example](#) the driver title was "MySQL-EABASE".
6. Enter the *User name* and *Password* (these are optional).
7. Enter the initial catalog (this is optional).
8. Press *Test Connection* to confirm that the details are correct.
9. If the test succeeds, press *OK*.
10. If the test does not succeed, revise your settings.
11. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [open project dialog](#).



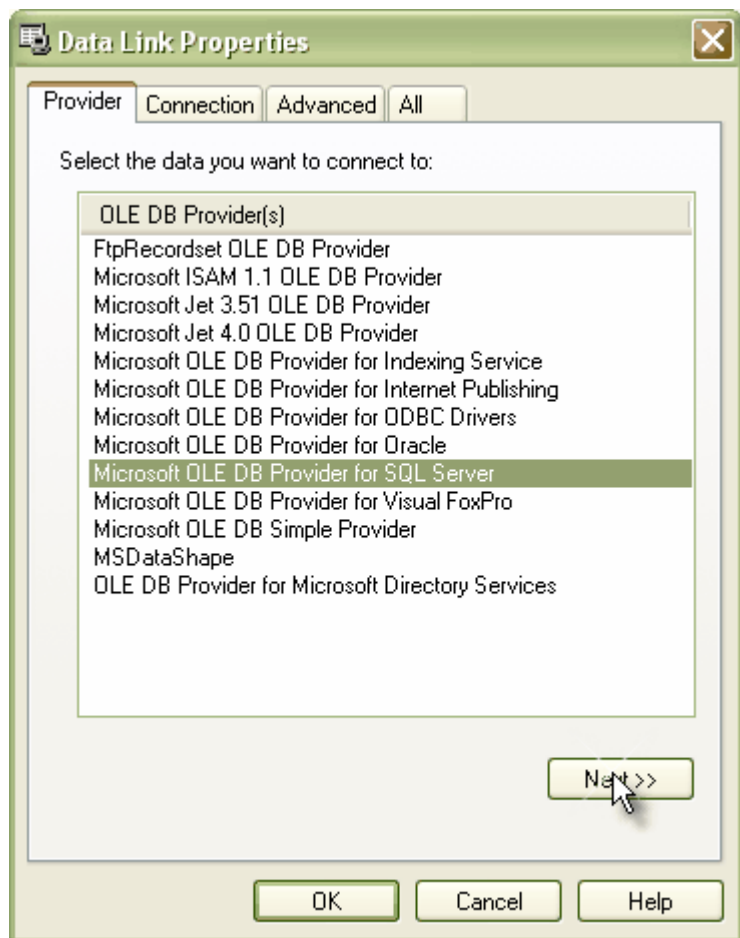
12. Press **OK** to complete the configuration.

7.1.4.2.2 Connect to a SQL Server Data Repository (Corporate Edition only)

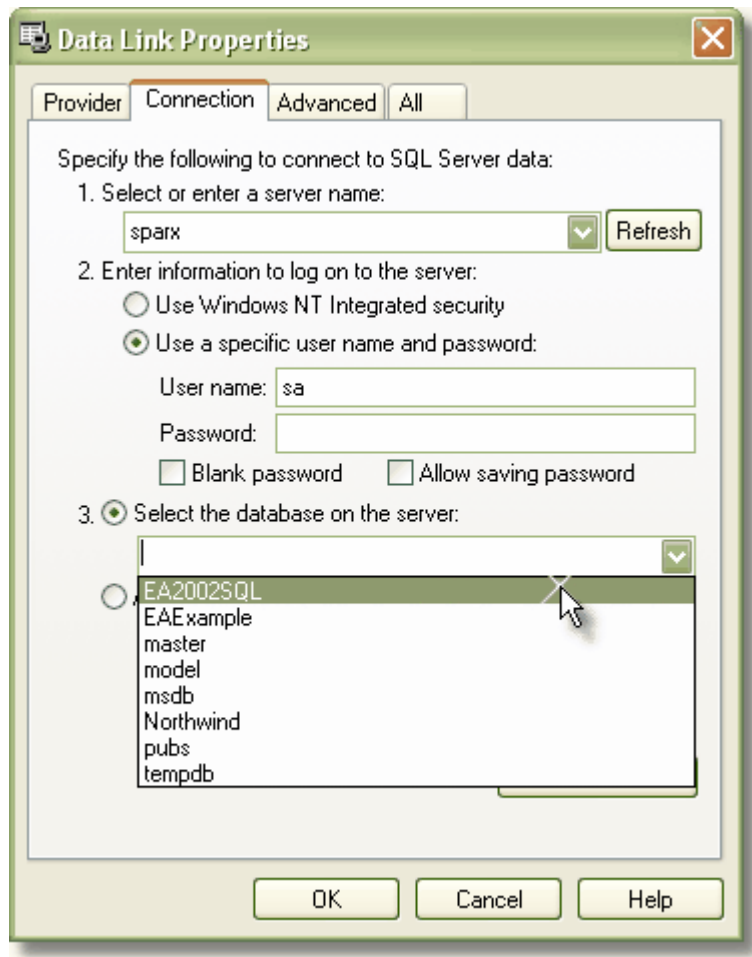
Note: This feature is available in the Corporate Edition only.

In order to use a SQL Server data repository, you must connect to it in EA first. Follow the steps below to connect to your SQL Server data repository in EA:

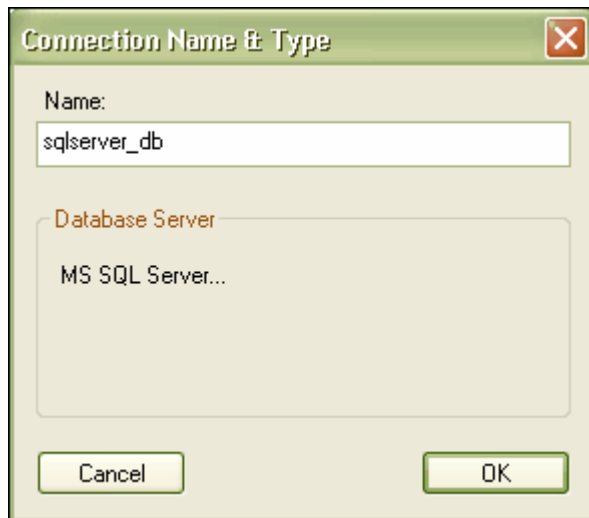
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox.
2. Press the Browse [\[...\]](#) button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for SQL Server" from the list.
4. Press *Next*. This will take you to the *Connection* tab (below).



5. Next enter the server details, including *Server Name*, *User Name* and *Password*.
6. From the *Select the database on the server* drop down list, select the model you wish to connect to.
7. Press *Test Connection* to confirm that the details are correct.
8. If the test succeeds, press *OK*.
9. If the test does not succeed, revise your settings.
10. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



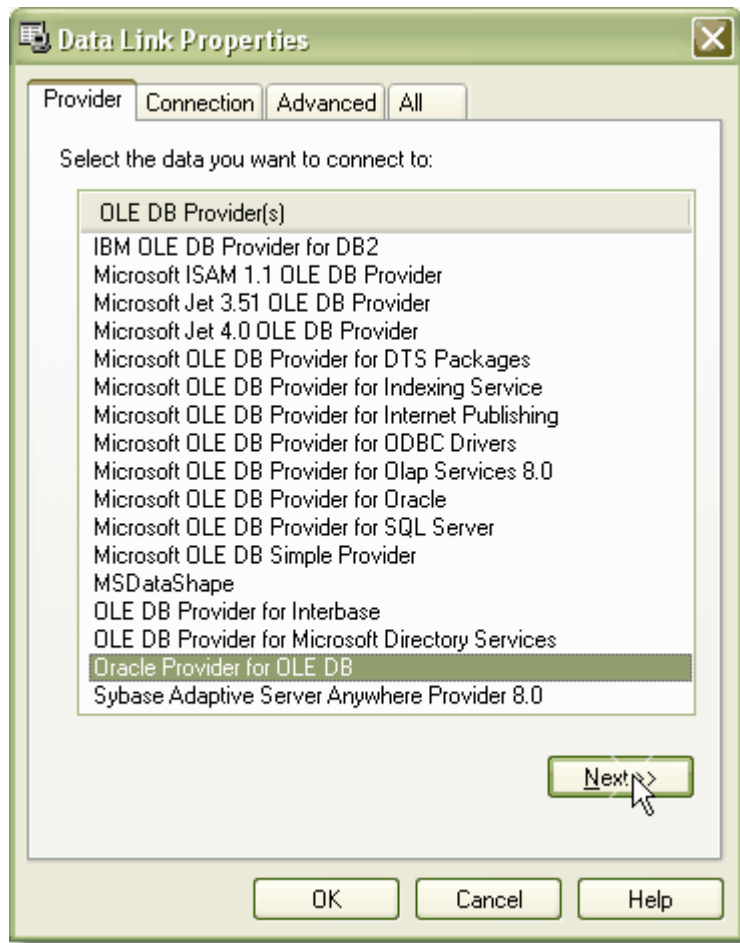
11. Press **OK** to complete the configuration.

7.1.4.2.3 Connect to an Oracle9i Data Repository (Corporate Edition only)

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

In order to use a Oracle9i data repository, you must connect to it in EA first. Follow the steps below to connect to your Oracle9i data repository in EA:

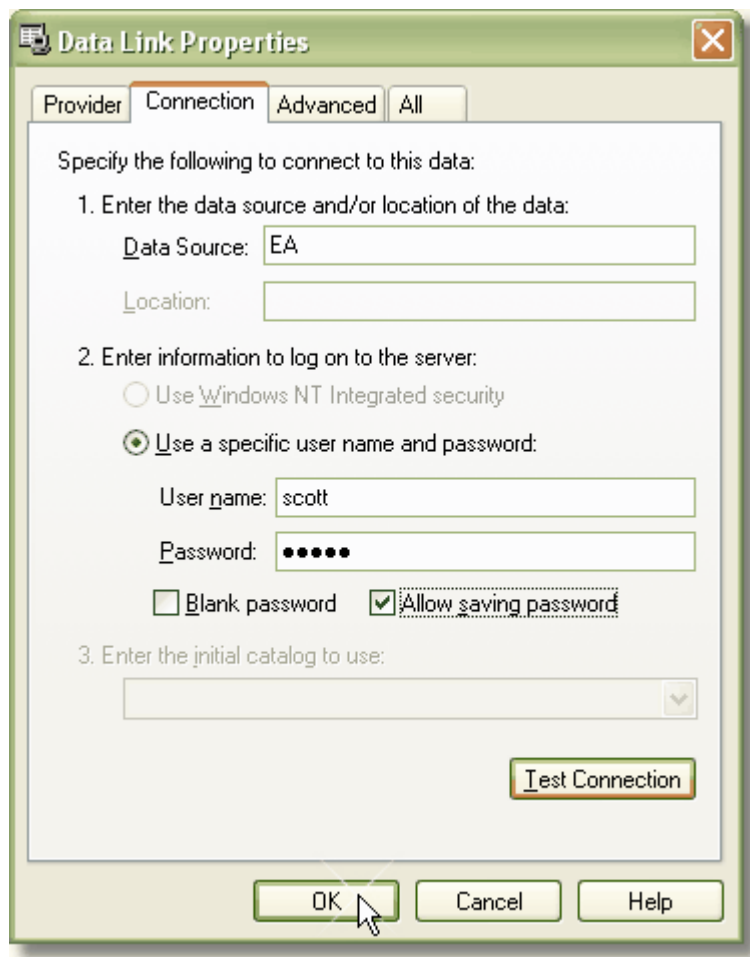
1. In the **Open Project dialog**, check the **Connect to Server** checkbox.
2. Press the Browse [...] button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



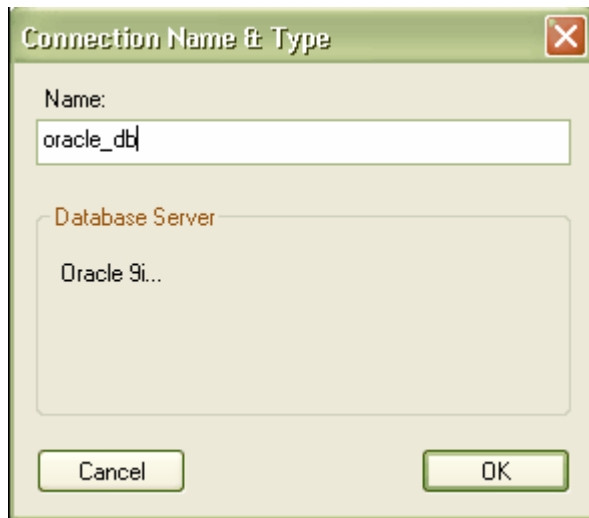
3. Select the "Oracle Provider for OLE DB" from the list.

Note: Do not select "Microsoft OLE DB Provider for Oracle" - EA may not work as expected.

4. Press *Next*. This will take you to the *Connection* tab (below).



5. Next enter the *Data Source* details, *User Name* and *Password*.
6. Press *Test Connection* to confirm that the details are correct.
7. If the test succeeds, press *OK*.
8. If the test does not succeed, revise your settings.
9. After you have pressed *OK*, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



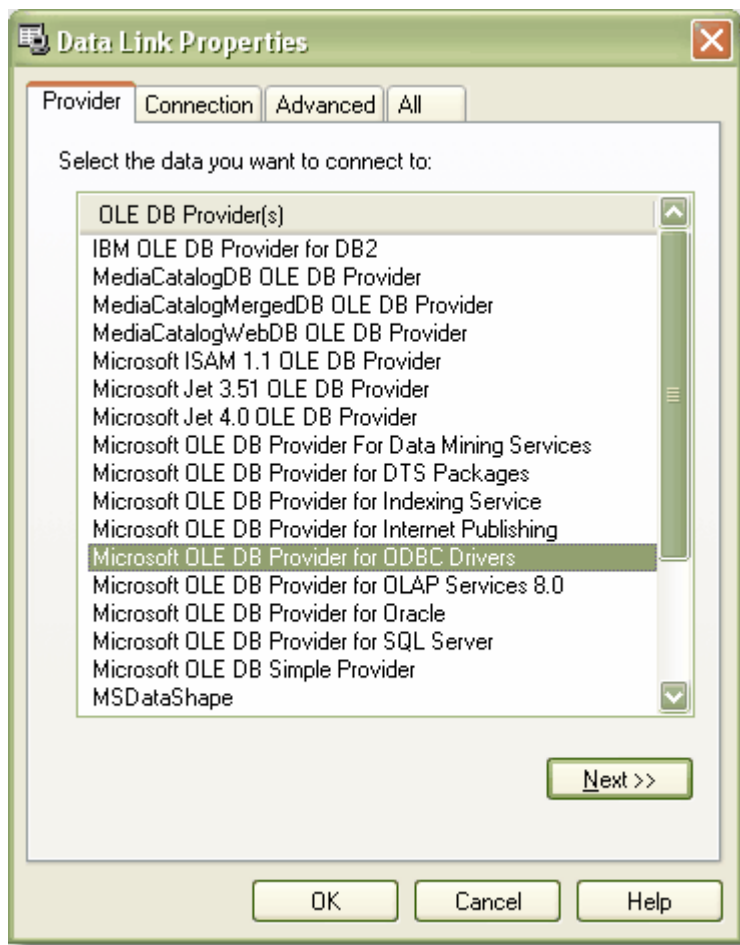
10. Press **OK** to complete the configuration.

7.1.4.2.4 Connect to a PostgreSQL Data Repository (Corporate Edition only)

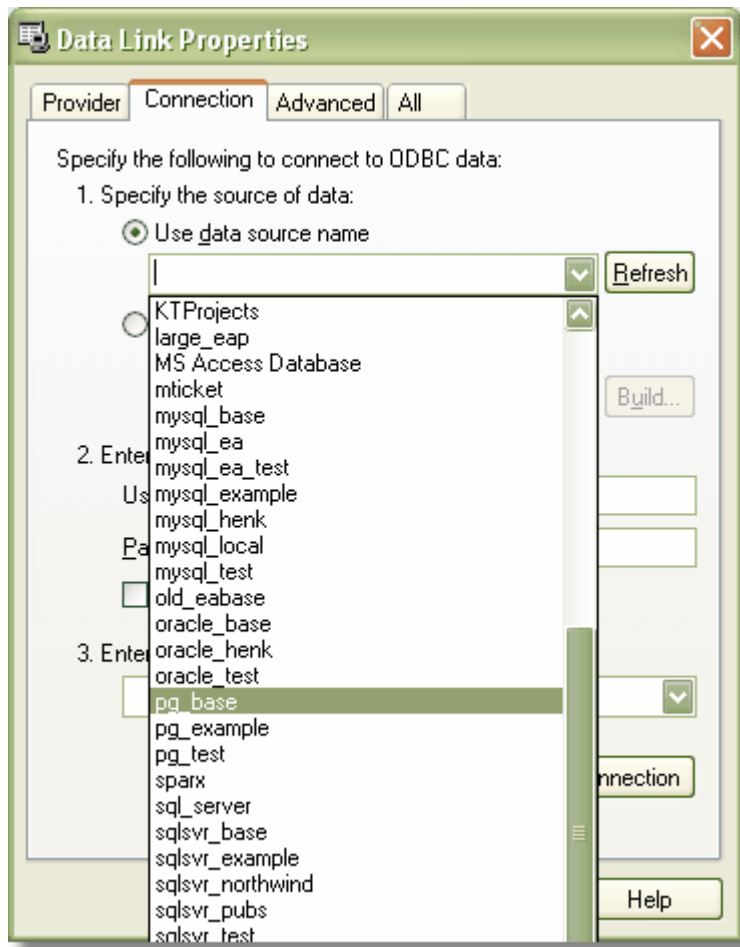
Note: This feature is available in the Corporate Edition only.

In order to use a PostgreSQL data repository, you must connect to it in EA first. Before connecting to the repository, you will need to have [set up a PostgreSQL ODBC driver](#). Follow these steps to connect to a MySQL data repository in EA:

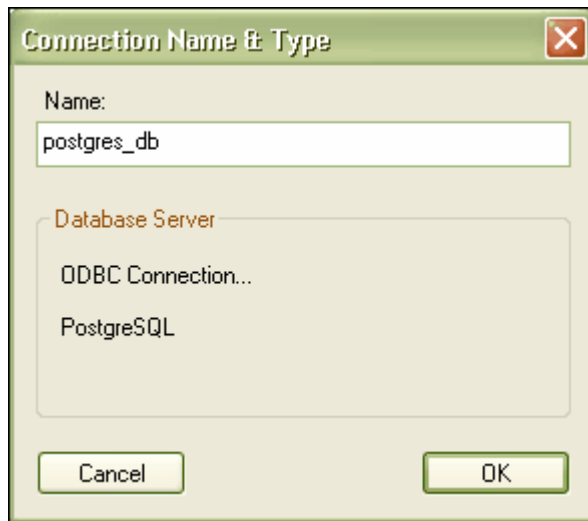
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox, or on the Start Page, click on the **Connect to Server Repository...** link.
2. Press the Browse [\[...\]](#) button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press the *Next* button. This will take you to the *Connection* tab (below).



5. Select the ODBC driver you have set up to connect to your PostgreSQL repository from the *Use data source name* drop down list.
6. Press the *Test Connection* button to confirm that the details are correct.
7. If your test succeeded, press the *OK* button.
8. If your test did not succeed, revise your settings.
9. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



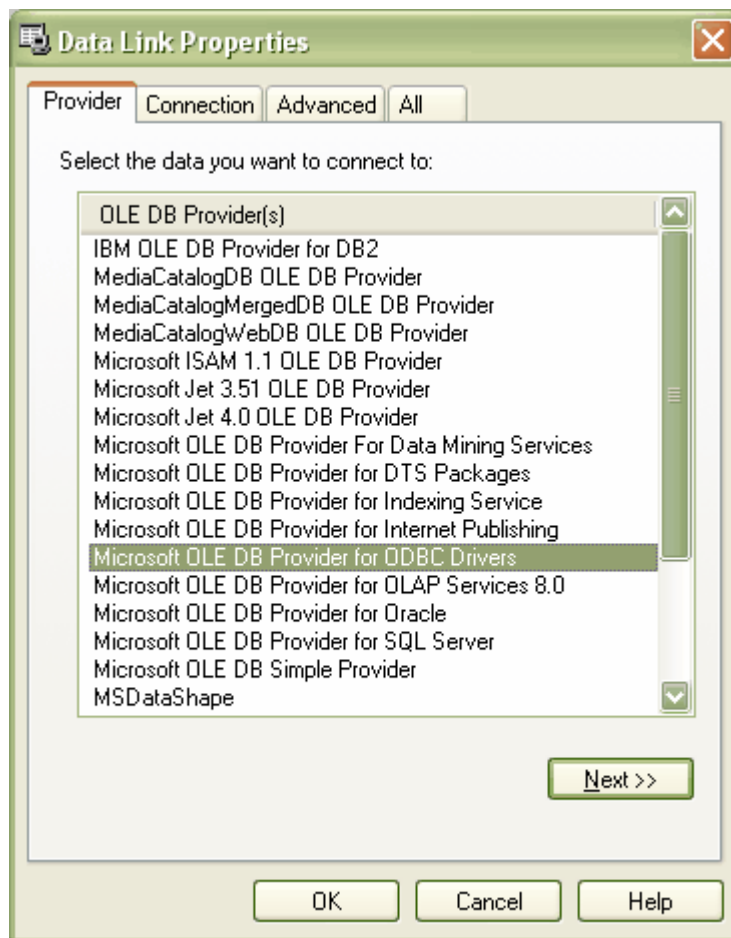
10. Press the **OK** button to complete the configuration.

7.1.4.2.5 Connect to an Adaptive Server Anywhere Data Repository (Corporate Edition only)

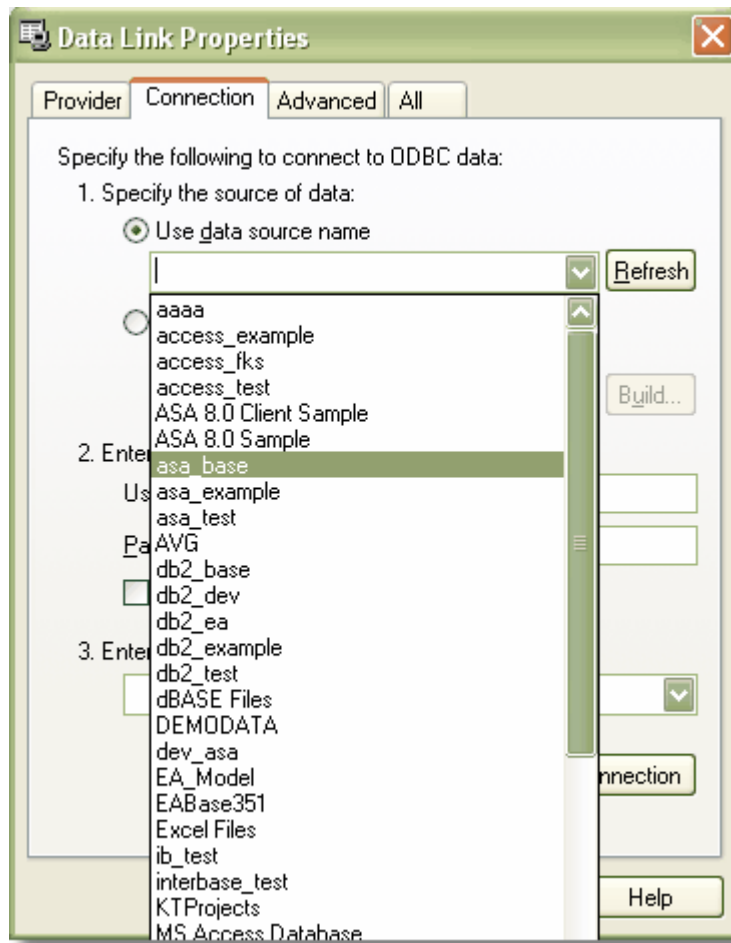
Note: This feature is available in the Corporate Edition only.

In order to use a ASA data repository, you must connect to it in EA first. Before connecting to the repository, you will need to have [set up an ASA ODBC driver](#). Follow these steps to connect to an ASA data repository in EA:

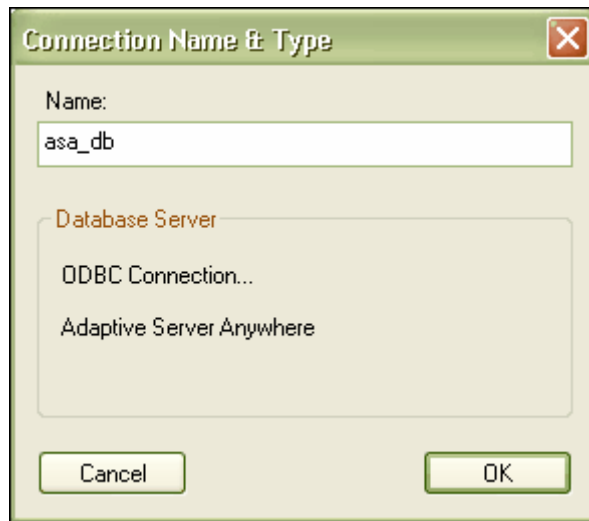
1. In the [Open Project dialog](#), check the **Connect to Server** checkbox, or on the Start Page, click on the **Connect to Server Repository...** link.
2. Press the Browse [\[...\]](#) button (as you normally would to browse for a project). As you have **Connect to Server** checked, the **Data Link Properties** dialog (below) will appear instead of the browse directories dialog.



3. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list.
4. Press the *Next* button. This will take you to the *Connection* tab (below).



5. Select the ODBC driver you have set up to connect to your ASA repository from the *Use data source name* drop down list.
6. Press the *Test Connection* button to confirm that the details are correct.
7. If your test succeeded, press the *OK* button.
8. If your test did not succeed, revise your settings.
9. After you have pressed OK, the *Connection Name* dialog will appear (below). Give the connection a suitable name so you will recognize it in the *Recently Opened Projects* list on the [Open Project dialog](#).



10. Press the **OK** button to complete the configuration.

7.1.4.2.6 Connect to a MSDE Server Data Repository (Corporate Edition only)

Follow the instructions in [Connect to a SQL Server Repository](#).

7.1.4.3 Creating a Repository

This section details how to create to the following data repositories:

- [MySQL Data Repository](#)
- [SQL Server Data Repository](#)
- [Oracle9i Data Repository](#)
- [PostgreSQL Data Repository](#)
- [Adaptive Server Anywhere Data Repository](#)
- [MSDE Server Data Repository](#)

7.1.4.3.1 Create a New MySQL Repository (Corporate Edition only)

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

Before creating a MySQL data repository in EA, you will need MySQL and MySQL ODBC drivers set up. For further information on setting these up, refer to [Setup a MySQL ODBC Driver](#).

To create a new MySQL repository, you will need to first create a database into which you will import the table definitions for EA. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

Creating the Data Repository

Once you have created the database and executed the script, you should have an empty EA project to begin working with. You can transfer data from an existing .EAP file or simply start from scratch.

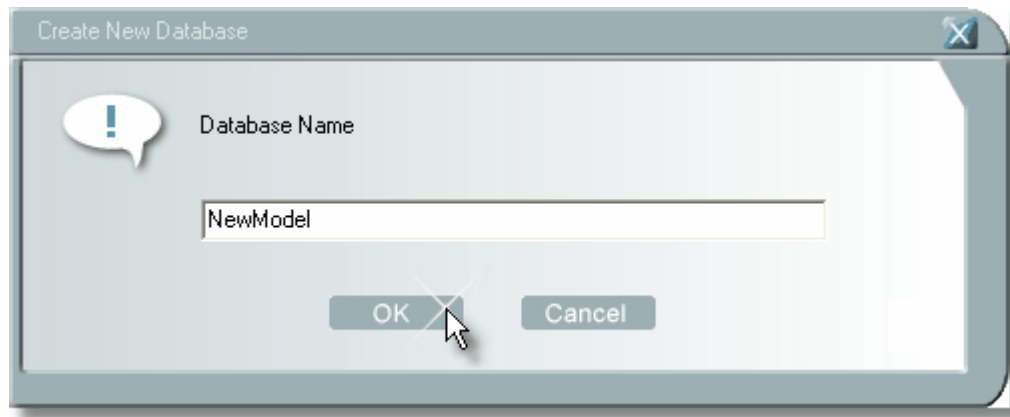
Third Party Tools

If you are unfamiliar with MySQL and DBMS systems in general, you may wish to consider a suitable front end tool.

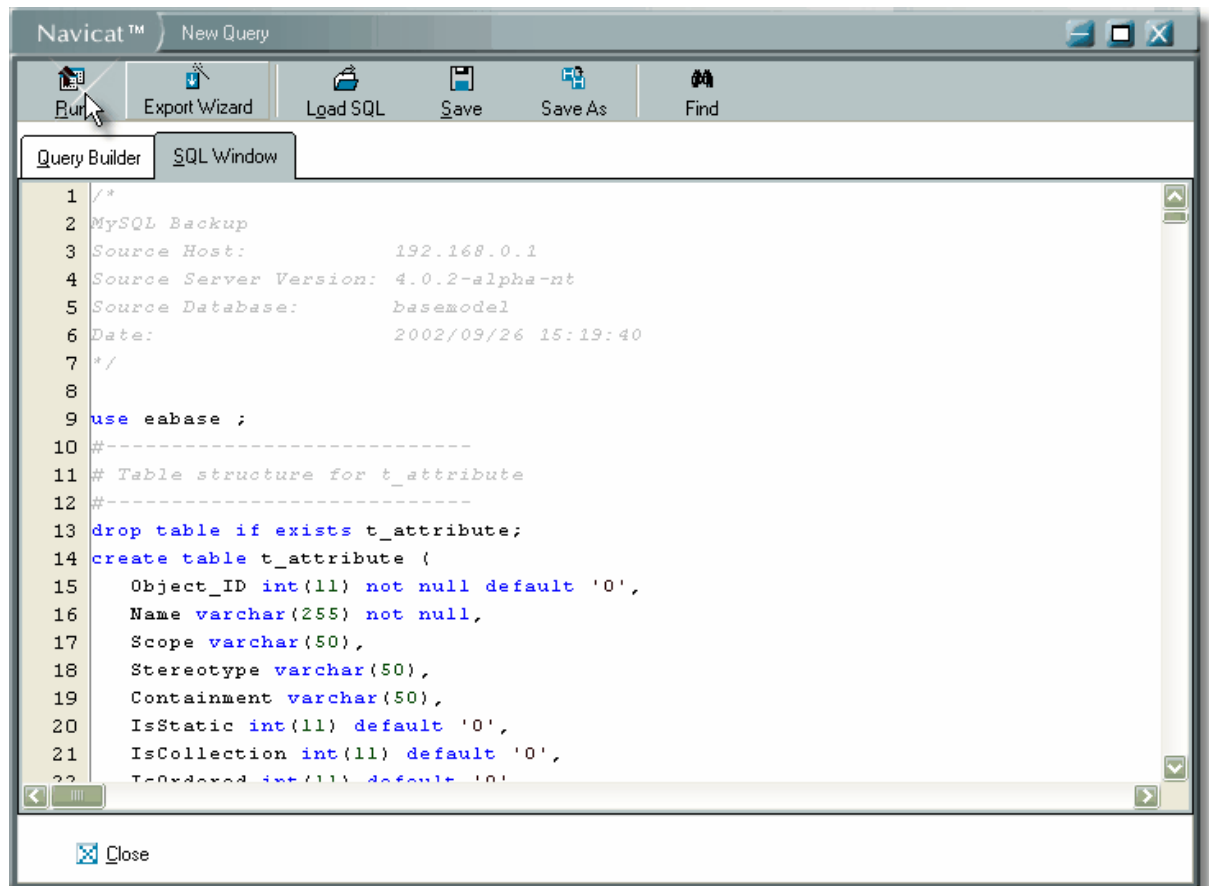
PremiumSoft Navicat (formerly PremiumSoft MySQLStudio) is one such tool, and is available at <http://www.mysqlstudio.com>. Navicat provides a convenient graphical user interface to enable the creation of databases, execution of scripts, backups, restores and more.

Below are some basic instructions to get you started with Navicat.

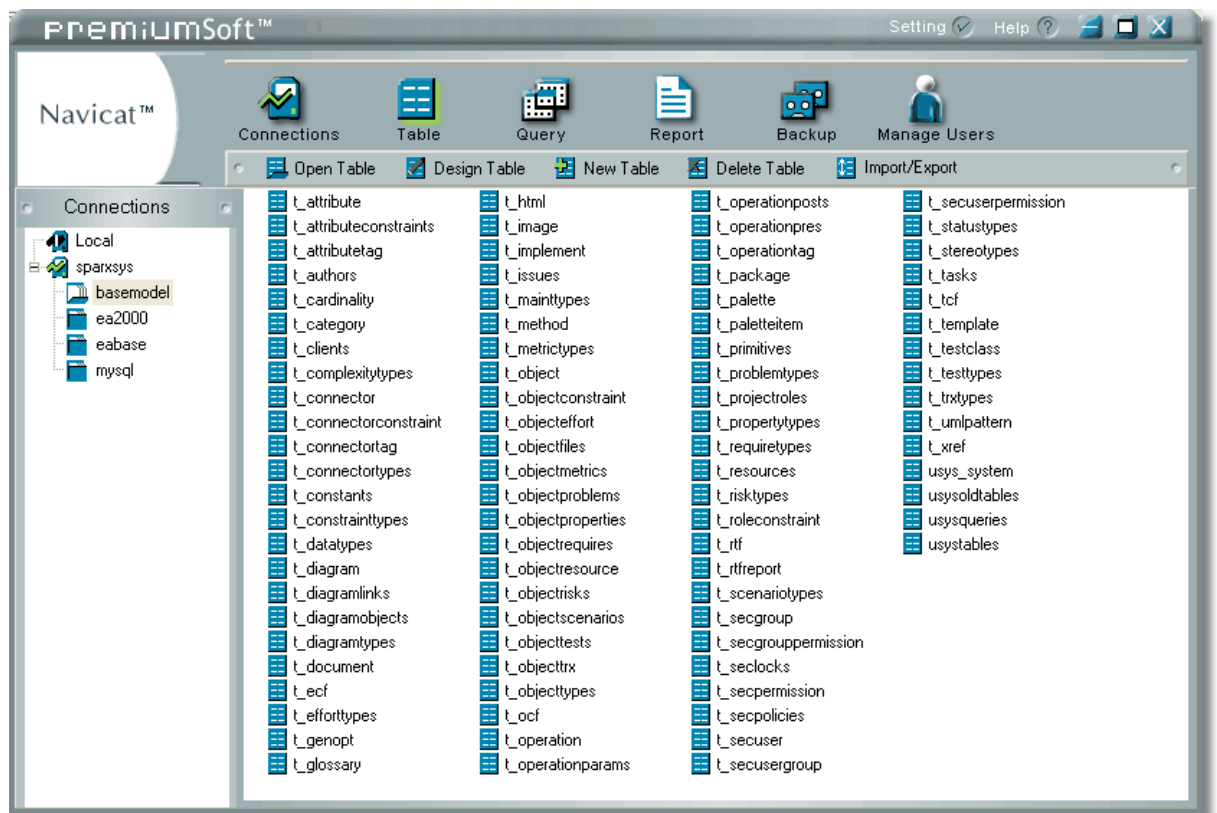
1. Create a new database.



2. Open and execute the MySQL script.



3. Below is an example showing the tables created in the MySQL repository after running the script in Navicat.



7.1.4.3.2 Create a New SQL Server Repository (Corporate Edition only)

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

Before creating a SQL Server data repository, you will need SQL Server installed, MDAC 2.6 or 2.7 installed and permission to create a new database. Please note that setting up SQL Server and the issues involved are beyond the scope of this manual. Consult your program's documentation for a guide to this.

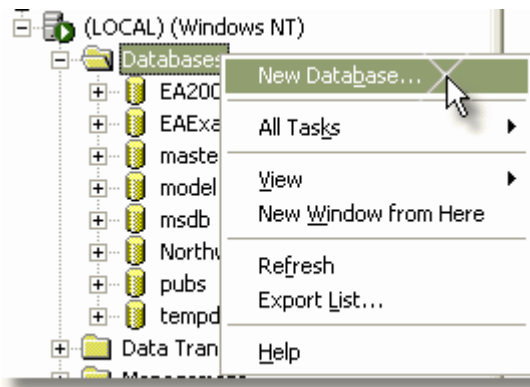
Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

Create a SQL Server Repository

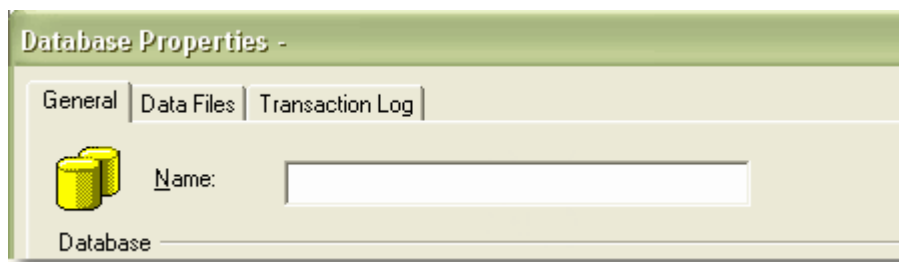
SQL Server repositories are created without any data, so you will need to perform a [data transfer](#) in EA to copy a suitable starter project. If you are starting from scratch, EABase.EAP is a good starting point. If you wish to use an existing .EAP model, you can [upsized](#) it.

You can use SQL Enterprise Manager to create a SQL Server repository by following the below steps:

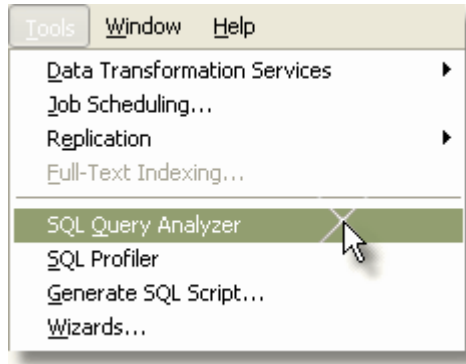
1. In SQL Enterprise Manager, locate the server on which you wish to create your new EA model - in the example below this is (LOCAL).
2. Right click and choose **New Database** from the context menu (below).



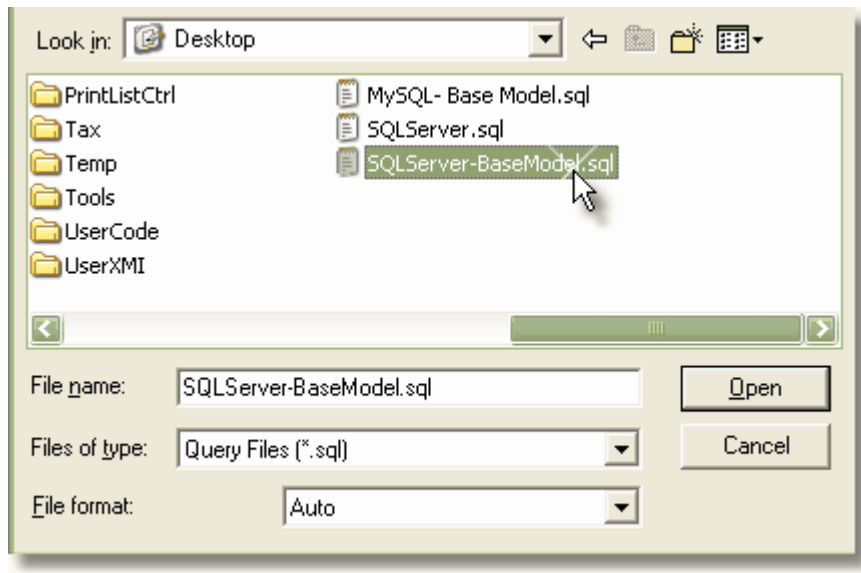
3. Enter a suitable name for the Database. Set any file options etc. as required (below).



4. Click on the database to select it, then go to the *Tools* menu and start *SQL Query Analyzer* (below).



5. In Query Analyzer, use the File Find dialog to locate the supplied EA SQL Server Model script file. Press *Open*.



6. Check that you have selected the correct database to run the script in. In this example the tables are being added to the Base Model database as shown in the drop down menu below.



7. Press **Run** - SQL Server will execute the script which will create the base model for an EA Project.

Tip: Use the [Data Transfer](#) function to upload a basic model into the repository.

7.1.4.3.3 Create a New Oracle9i Server Repository (Corporate Edition only)

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

Before creating a Oracle9i data repository, you will need Oracle9i installed, MDAC 2.6 or 2.7 installed and permission to create a new database. Please note that setting up Oracle9i and the issues involved are beyond the scope of this manual. Consult your program's documentation for a guide to this.

Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

Creating the Data Repository

Oracle 9i repositories are created without any data, so you will need to perform a [data transfer](#) in EA to copy a suitable starter project. If you are starting from scratch, [EABase.EAP](#) is a good starting point. If you wish to use an existing .EAP model, you can [upsized](#) it.

1. Create a new database on the Oracle9i server.
2. Connect to the newly created database with a program such as *Oracle SQL*Plus* or *SQL Plus Worksheet*.
3. Execute the script Oracle9i_BaseModel.sql which will create the base model tables and indexes for an EA Project.

Tip: Use the [Data Transfer](#) function to upload a basic model into the repository.

7.1.4.3.4 Create a New PostgreSQL Repository (Corporate Edition only)

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

Before creating a PostgreSQL data repository in EA, you will need PostgreSQL and PostgreSQL ODBC drivers set up. For further information on setting these up, refer to [Setup a PostgreSQL ODBC Driver](#).

To create a new PostgreSQL repository, you will need to first create a database into which you will import the table definitions for EA. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

Creating the Data Repository

After you create the database and execute the script, the result should be an empty EA project to begin working with. You can transfer data from an existing .EAP file or simply start from scratch.

Third Party Tools

If you are unfamiliar with PostgreSQL and DBMS systems in general, you may wish to consider a suitable front end tool.

EMS PostgreSQL Manager is one such tool, and is available at <http://ems-hitech.com/pgmanager>. It provides a convenient graphical user interface to enable creation of databases, execution of scripts, restores and more.

Below are some basic instructions to get you started with EMS PostgreSQL Manager.

1. Create a new database.

Create Database Wizard

General Options
Create Database

Set new database name:
new_db

Set host properties:

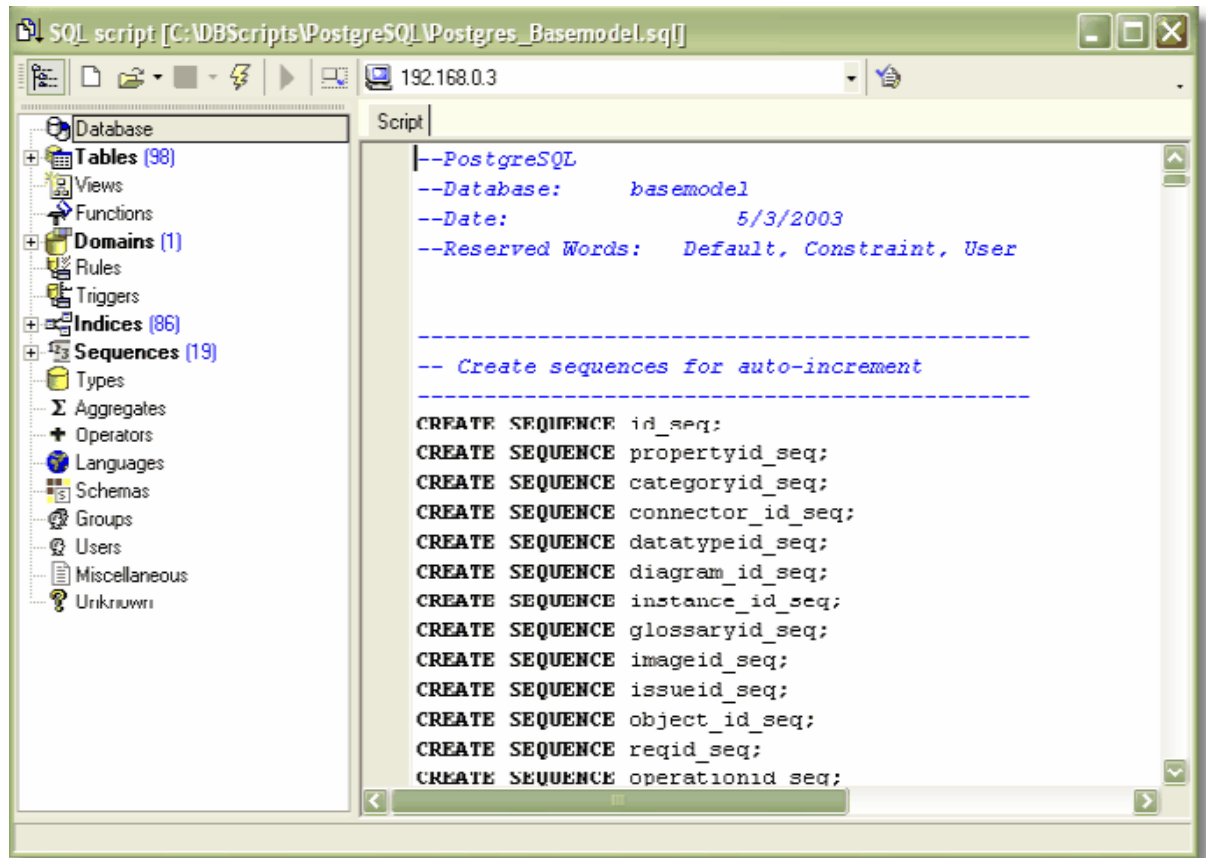
Host: 192.168.0.3 Register After Creating
Port: 5432
Login: postgres
Password:

Set database properties (can be blank):

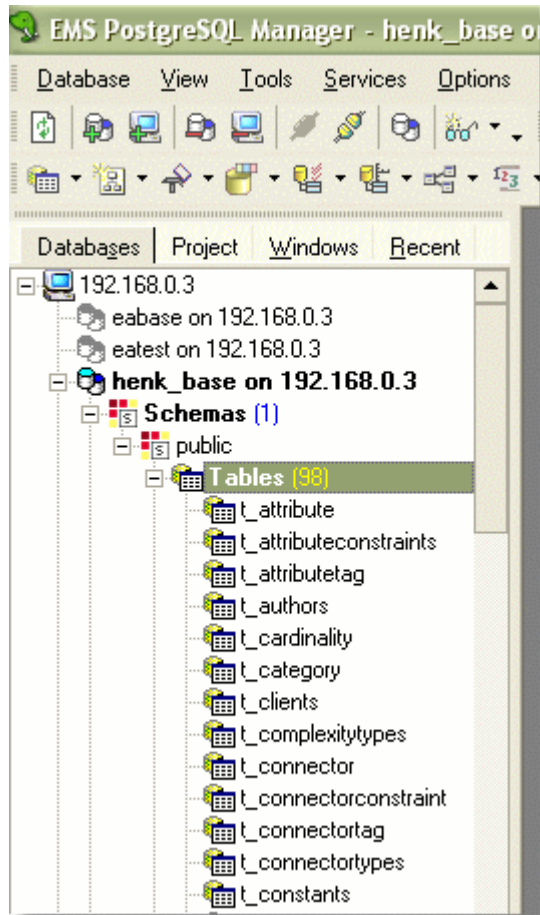
Location:
Template:
Encoding:
Owner (7.3 or higher):

Cancel Back Next Create Help

- Open and execute the PostgreSQL sql script.



- Below is an example showing the tables created in the PostgreSQL repository after running the script in EMS PostgreSQL Manager.



7.1.4.3.5 Create a New Adaptive Server Anywhere Repository (Corporate Edition Only)

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

Before creating a ASA data repository in EA, you will need ASA and ASA ODBC drivers set up. For further information on setting these up, refer to [Setup an Adaptive Server Anywhere ODBC Driver](#).

To create a new ASA repository, you will need to first create a database into which you will import the table definitions for EA. Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

Creating the Data Repository

After you create the database and execute the script, the result should be an empty EA project to begin working with. You can transfer data from an existing .EAP file or simply start from scratch.

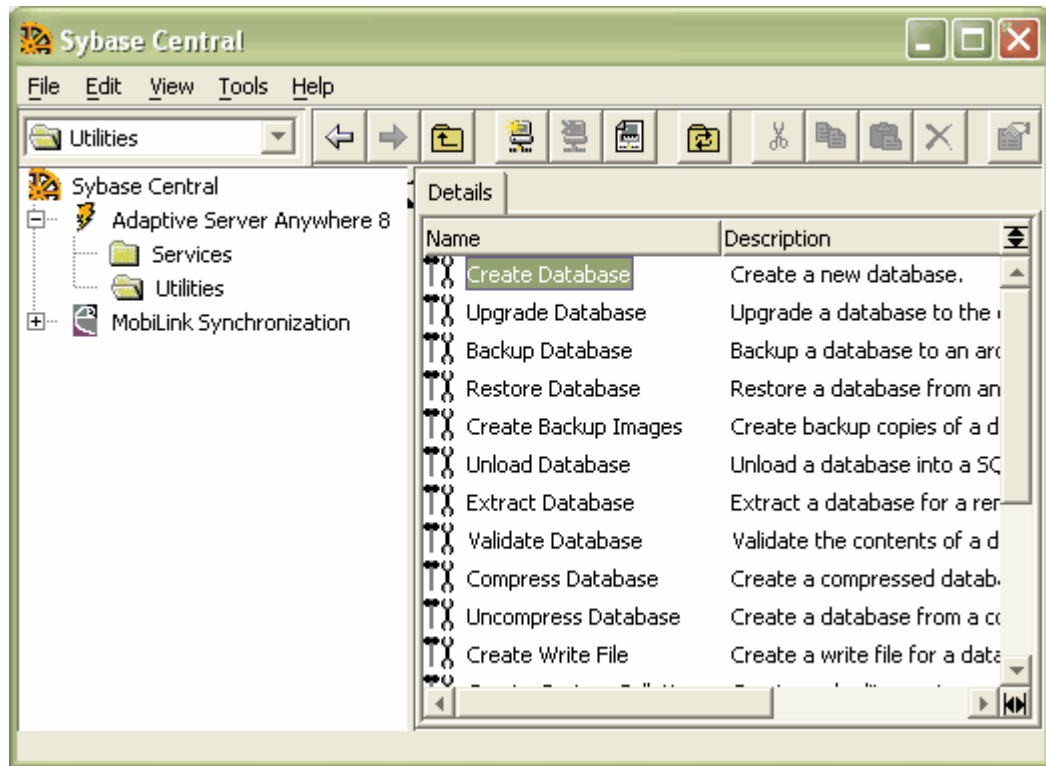
Third Party Tools

If you are unfamiliar with ASA and DBMS systems in general, you may wish to consider a suitable front end tool.

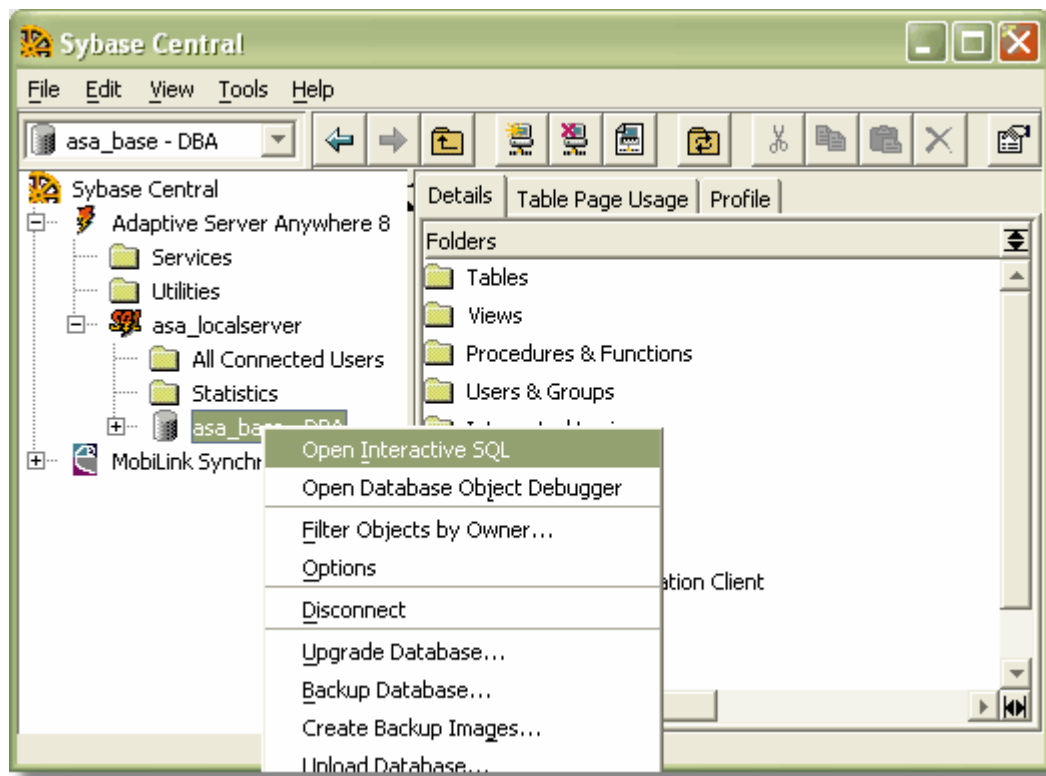
Sybase Central is one such tool, that can be installed along with the DBMS. It provides a convenient graphical user interface to enable creation of databases, execution of scripts, restores and more.

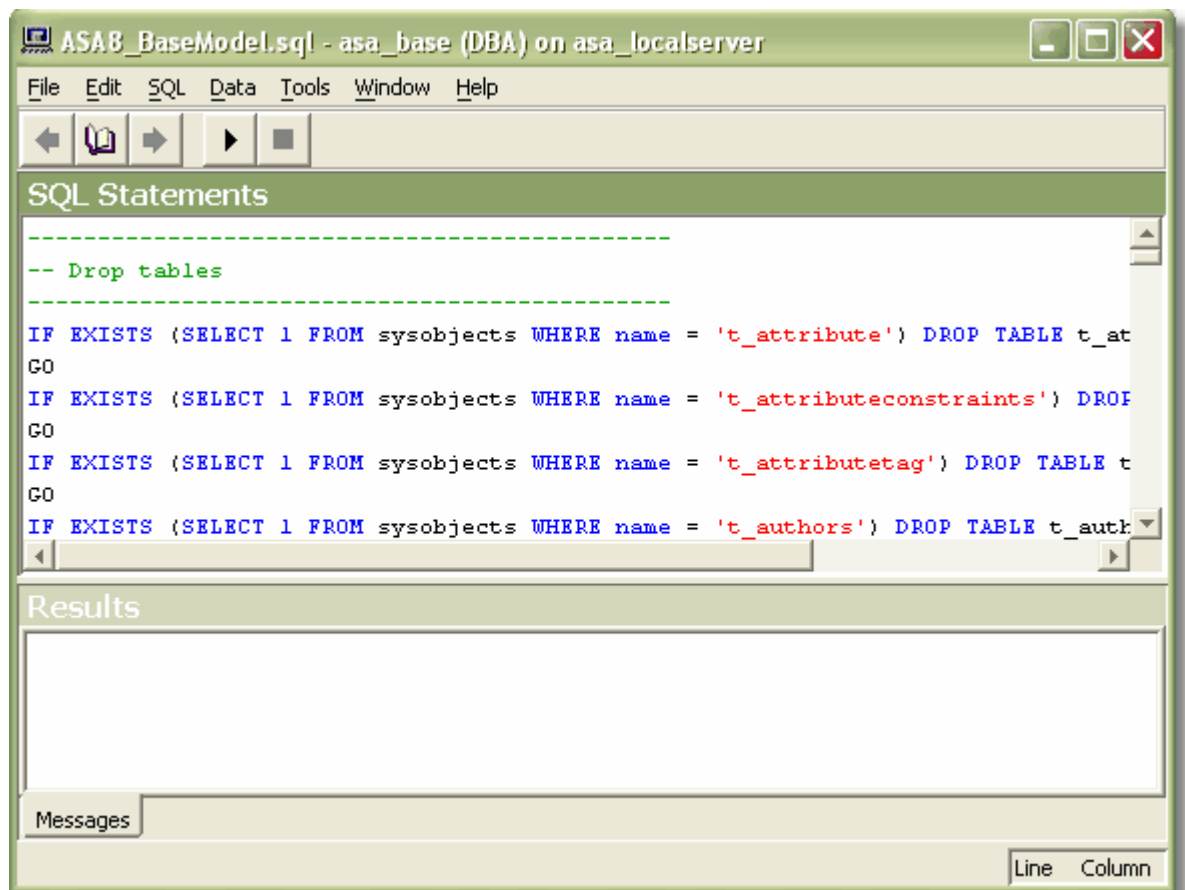
Below are some basic instructions to get you started with Sybase Central.

1. Create a new database.

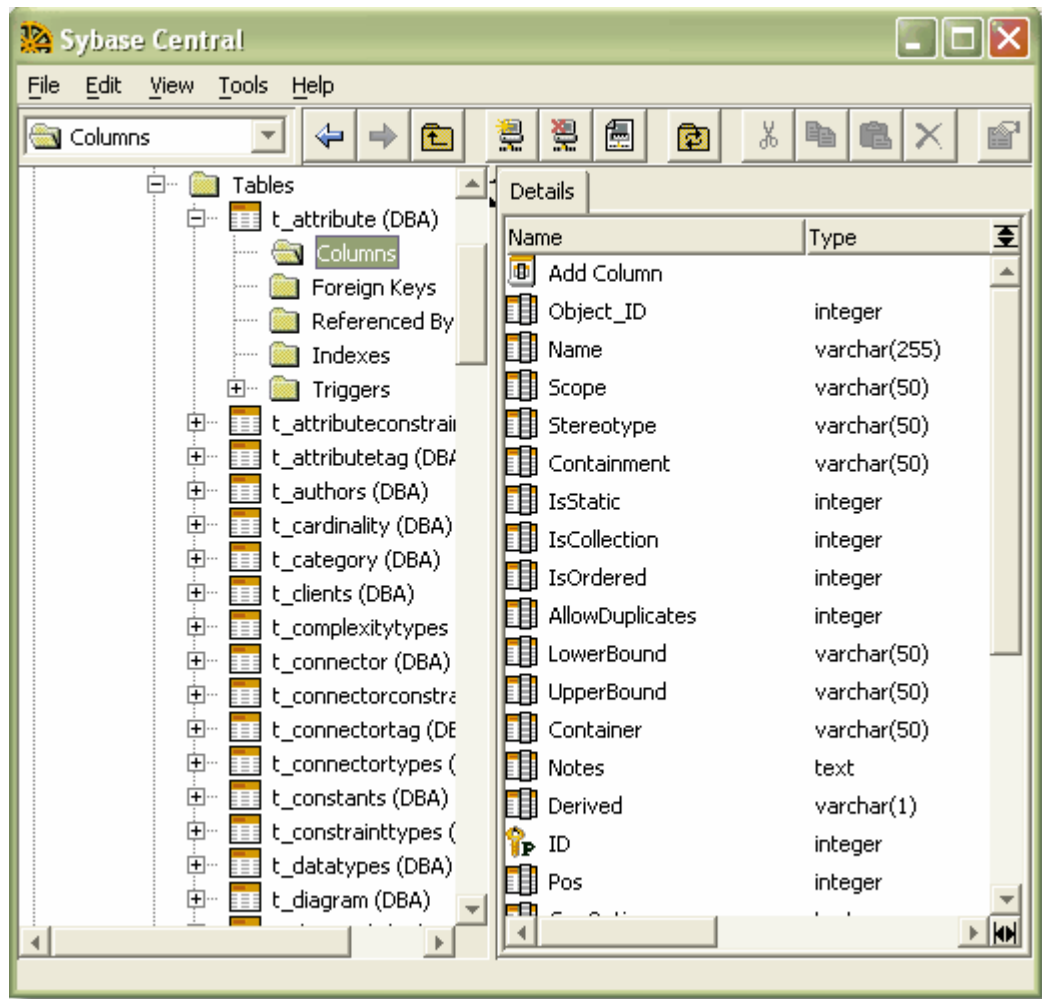


2. Open and execute the ASA SQL script.





3. Below is an example showing the tables created in the ASA repository after running the script in EMS ASA Manager.



7.1.4.3.6 Create a New MSDE Server Repository (Corporate Edition only)

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

Before creating a SQL Server MSDE data repository, you will need MSDE Server installed, MDAC 2.6 or 2.7 installed. Please note that setting up MSDE Server and the issues involved are beyond the scope of this manual. Consult your program's documentation for a guide to this.

Sparx Systems provide SQL scripts to create the required tables - how you create the database and execute that script are up to you. Registered users can obtain the scripts from the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm. Use the SQL Server 2000 script for MSDE, and follow the instructions to [Create a New SQL Server Data Repository](#).

7.1.5 Create and Open Model Files Discussion

MySQL Limitations

Note that use of MySQL has the following limitations:

1. If MyISAM table types are used (default), transactional support is disabled. To enable transactions you will need to set up MySQL to use InnoDB tables and create the database tables as InnoDB type. Sparx provide a suitable script to create InnoDB based repository tables - as well as the more common MyISAM. These are available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.
2. Due to some limitations of the SQL query engine, some advanced search capabilities in the Search Dialog are disabled.
3. Only MySQL 4.0.10 or later is supported.
4. MySQL ODBC Driver 3.50 or higher is required. This is the development version, but again the earlier version does not provide sufficient support to run EA.

SQL Scripts

Sparx Systems provide various SQL scripts to assist with creating data repositories. These are available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

7.2 Upgrading Models

The structure of Enterprise Architect project files is occasionally changed to support more features. When this happens, existing project files must be upgraded to the new format to ensure correct operation and to take advantage of all the new features.

Upgrading is a simple and quick process. You can use the [Upgrade Wizard](#), which will alert you to the upgrade need, and take you through the upgrade process. This will bring your project to the current level to support all the latest EA features.

See also: [Upgrade Replicas](#)

7.2.1 The Upgrade Wizard

When you try to load a project that needs to be updated to the latest format, the upgrade wizard will open and guide you through the required steps. You cannot load a previous EA version model without upgrading.



Note: *Upgrading is essential when a new major version is released. Once upgraded, the project cannot be opened with earlier versions of EA.*

The wizard will:

- Alert you to the need to upgrade.
- Advise you to back up the current project. Backing up before any changes are made is essential.
- Check if the current project can be upgraded. All projects can be upgraded, except replicas which are not Design Masters.
- Perform the upgrade.
- Open the newly converted project.

Note for replicated models: *If the wizard detects the project you are opening is a replica and not a design master, a different upgrade path is required.*

7.2.2 Upgrade Replicas

Models that have replication features added may need to be handled in a way different from regular projects.

If the model is a [Design Master](#) (the root model of all other replicas) then you can upgrade the model to the current version. After upgrading a Design Master you should re-create the replicas, rather than synchronizing.

If the model is not a Design Master, the model cannot be upgraded in the normal manner. First EA must remove the replication features, then upgrade the project in the normal manner. The [Upgrade Wizard](#) will guide you through these steps.

The Upgrade Wizard should handle the upgrade process for you, detected which upgrade method is the best.

7.3 Data and Model Integrity

In the event of a failed XML import, network crash or other unforeseen event that could disrupt the integrity of information in the model, it is recommended to run the [Data Integrity check function](#). This will examine all database records and ensure there are no 'orphaned' records or inaccurate or unset identifiers. You can run the integrity checker first in report mode to discover if anything needs correcting, and then run it again in repair mode.

When EA checks the model, it will attempt to recover lost packages and elements, and will generate a new package at the model root level called '_recovered_'. Check through any elements that are found and if necessary drag them into the model proper. If they are not required, delete them.

Note: *This function does NOT check UML conformance ... only the data relationships and repository structure.*

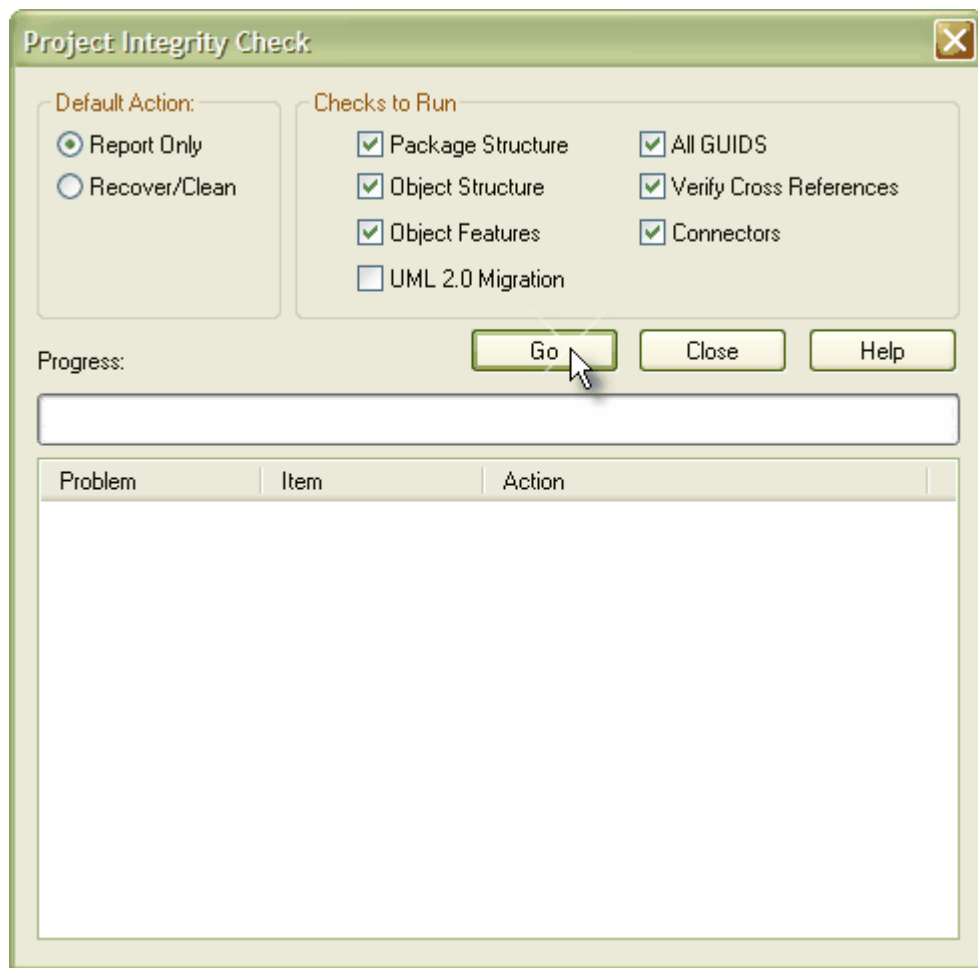
You can select a variety of items to check, and elect to either Report Only on the state of your model - or to try and repair any inconsistencies. The recovery process will try and restore elements where possible, but in some cases, will simply delete the lost records.

See also: [Running SQL Patches](#)

7.3.1 Checking Data Integrity

To check the data integrity of your model, follow these steps:

1. From the **Tools | Data Management** submenu, select **Data Integrity**. This will open the **Project Integrity Check** dialog:



2. Select which checks you want to run - the basic checks available are:
 - Package Structure
 - Object Structure
 - Object Features
 - GUIDs
 - Cross References
 - Connectors
 - UML 2.0 Migration
3. Select whether you just want to view a report of the state of your model, or whether you want to go ahead and attempt to recover/clean your project.

Warning: You should back up your project file first if you select the *Recover/Clean* option.

4. Press **Go** to run the check.

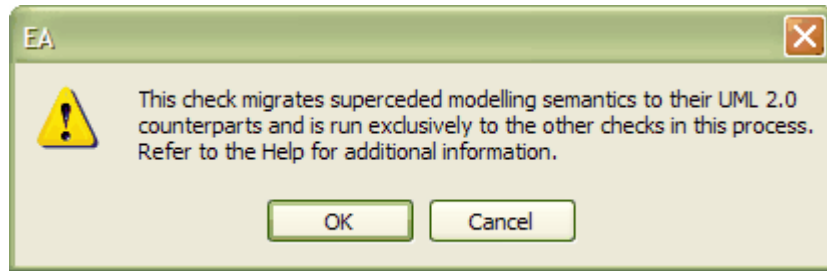
UML 2.0 Migration

The UML 2.0 Migration check allows users to migrate the project from the UML 1.3 semantic's to UML 2.0 semantic's. The Migration process currently converts activities that are invocations of operations, into called operation actions as per the UML 2.0 specification. The UML 2.0 Migration option is an exclusive process which does not allow any of the other checks to be selected. To perform the UML 2.0 migration use the following steps:

1. From the **Tools | Data Management** submenu, select **Data Integrity**. This will open the **Project Integrity**

Check dialog as shown previously.

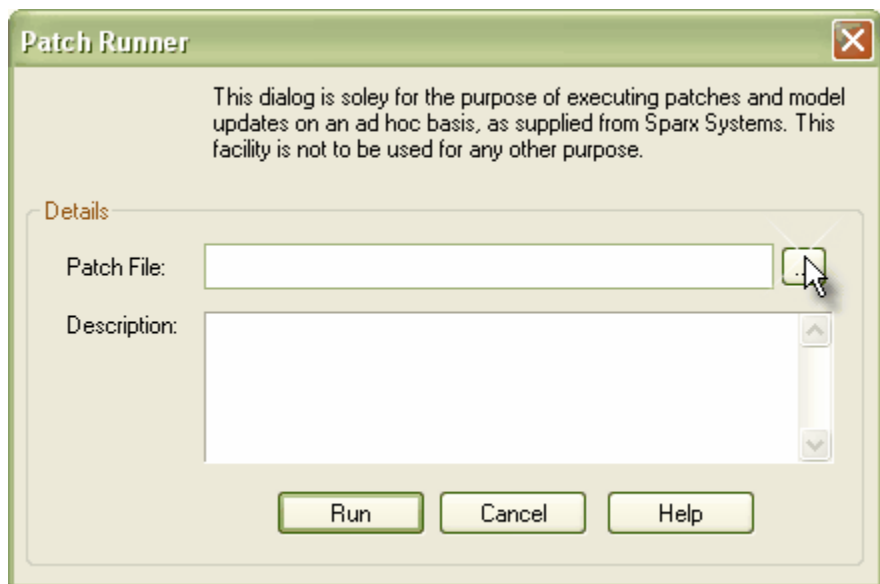
2. Check the *UML 2.0 Migration* checkbox and press the *Go* button. This will display the following dialog, press the *OK* button to proceed or cancel the migration by pressing the *Cancel* button.



2. Press the *Go* button to perform the migration.

7.3.2 Running SQL Patches

Occasionally, Sparx Systems may release a patch to correct a model fault. If a patch is released, you can load it here and run from the *Tools | Run Patch* menu. The patch will generally check how many records will be updated, and report on what will be done.



7.4 Data Transfer

The Corporate Edition of EA supports [SQL Server](#), [MySQL](#) and [Oracle](#) data repositories. At some point, the need may arise to move a complete model from one repository to another, row by row, table by table.

The data transfer function allows you to perform the following tasks:

- Upload an existing EAP file to SQL Server or MySQL
- Download a repository in MySQL or SQL Server to an EAP file

- Move a repository from SQL Server to MySQL or from one server to another
- Move all records from an EAP file with replication to a model with none (Remove Replication)
- Copy all records from an EAP file to another (recommended after serious network crash or repeated database corruption)
- Copy all records from a JET 3.5 to JET 4 (Access 2000 or XP) - or back the other way

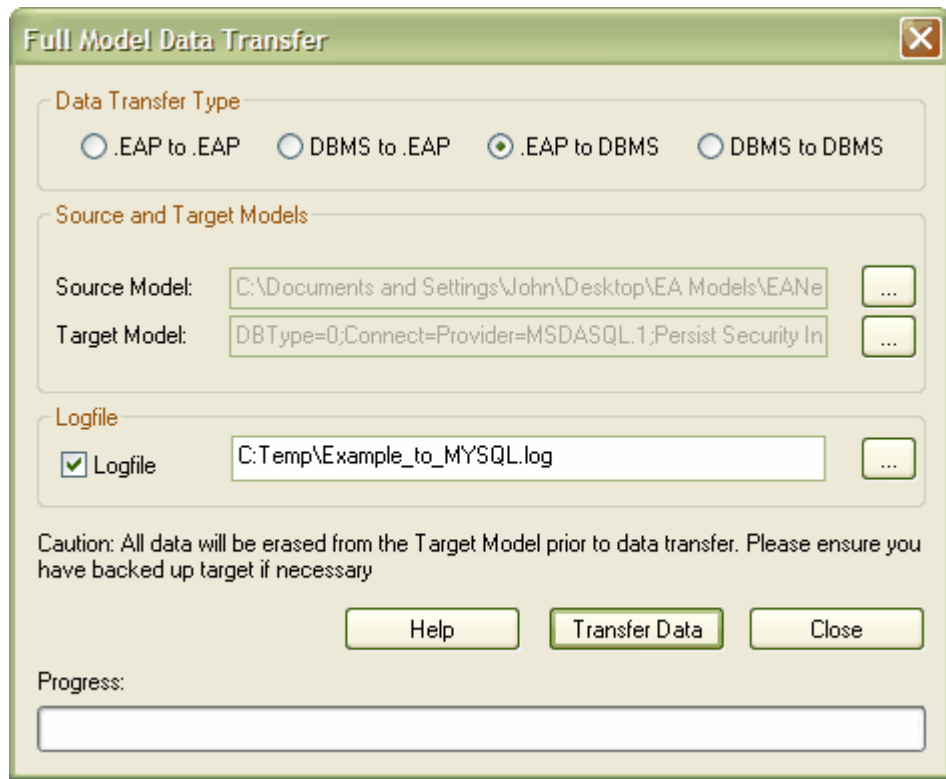
See the [Perform a Data Transfer](#) topic for instructions.

WARNING: ALL RECORDS IN THE TARGET REPOSITORY WILL BE OVERWRITTEN

7.4.1 Perform a Data Transfer

To use the data transfer function, follow these steps:

1. From the **Tools** | **Data Management** submenu, select **Data Transfer**. This will open the **Full Model Data Transfer** dialog:



2. Select the **Data Transfer Type** - you can choose from:
 - .EAP to .EAP
 - DBMS to .EAP
 - .EAP to DBMS
 - DBMS to DBMS
3. Enter the name or connection string for the **Source** and **Target** models.
4. Press **Transfer Data**.

5. It is good practise to do a [Database Compare](#) after this process to verify all records are written.

WARNING: ALL RECORDS IN THE TARGET REPOSITORY WILL BE *OVERWRITTEN*

7.4.2 Why Compare Models?

It is sometimes useful to compare the size and row counts of two models. After a database crash, after import from XML or after performing a deletion of model elements - these are all examples of when it may be useful to compare models.

You can compare EAP files to other EAP files or to DBMS based repositories - or compare two DBMS repositories. EA will examine the number of rows in each database and produce a report indicating the total records in each and the difference between the two. No examination is made of the data in the table - just the record count.

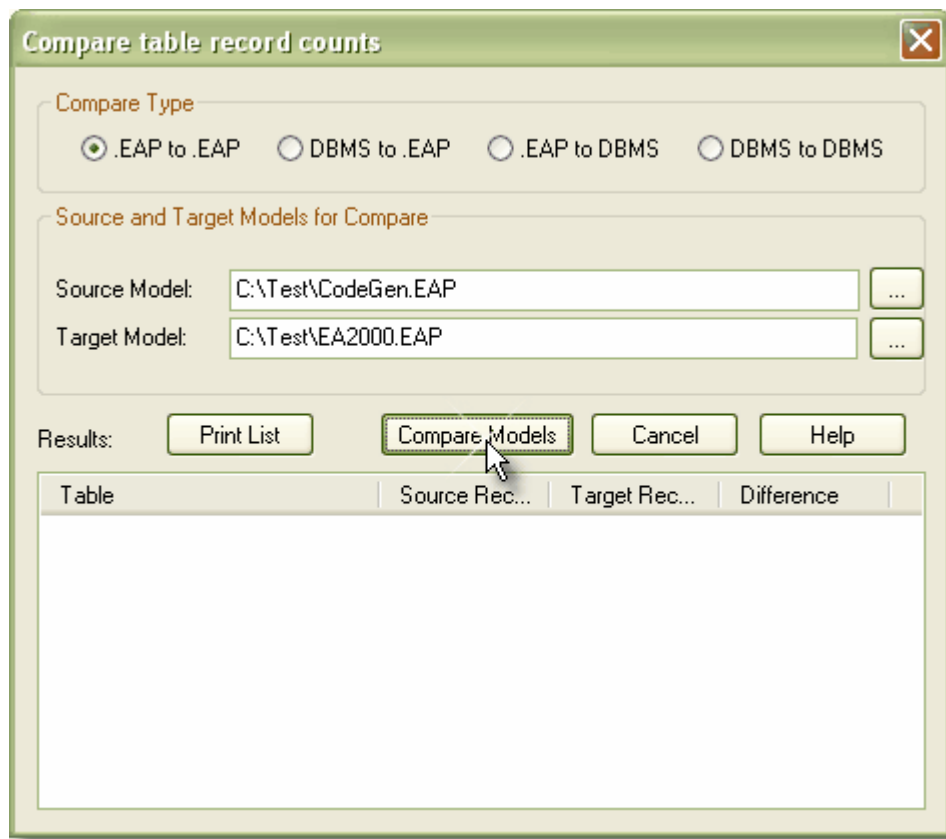
Comparing models this way is a convenient 'sanity check' after restoring a backup or doing a data transfer. If discrepancies are found, you will need to investigate further manually.

See the [Compare Models](#) topic for instructions.

7.4.3 Comparing Models

To compare models, follow the steps below:

1. From the *Tools | Data Management* submenu, select *Data Compare*. This will open the *Compare table record counts* dialog:



2. Select the *Compare Type* - you can choose from:
 - .EAP to .EAP
 - DBMS to .EAP
 - .EAP to DBMS
 - DBMS to DBMS
3. Enter the name or connection string for the *Source* and *Target* models.
4. Press *Compare Models*. The results will appear in the listbox below. You can also print the list by pressing *Print List*.

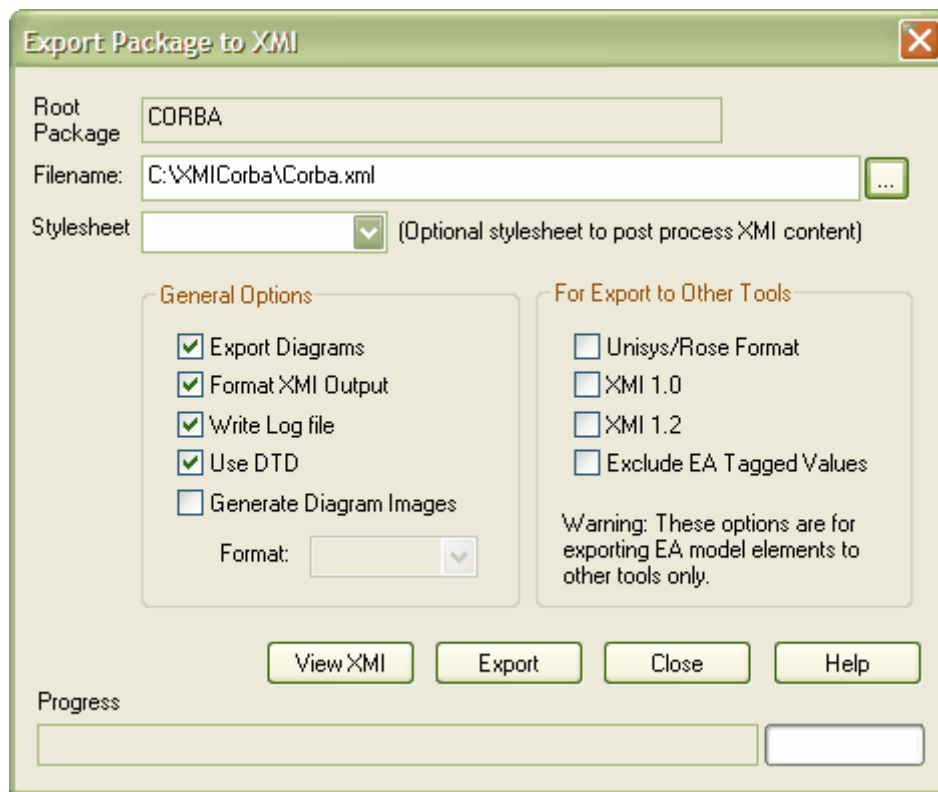
7.4.4 Copy Packages from One Project to Another

Using the XML import/export capabilities of Enterprise Architect, you can copy and move packages between EA models. This allows for a high level of flexibility in building a model from re-usable parts and from elements produced in widely dispersed geographic regions.

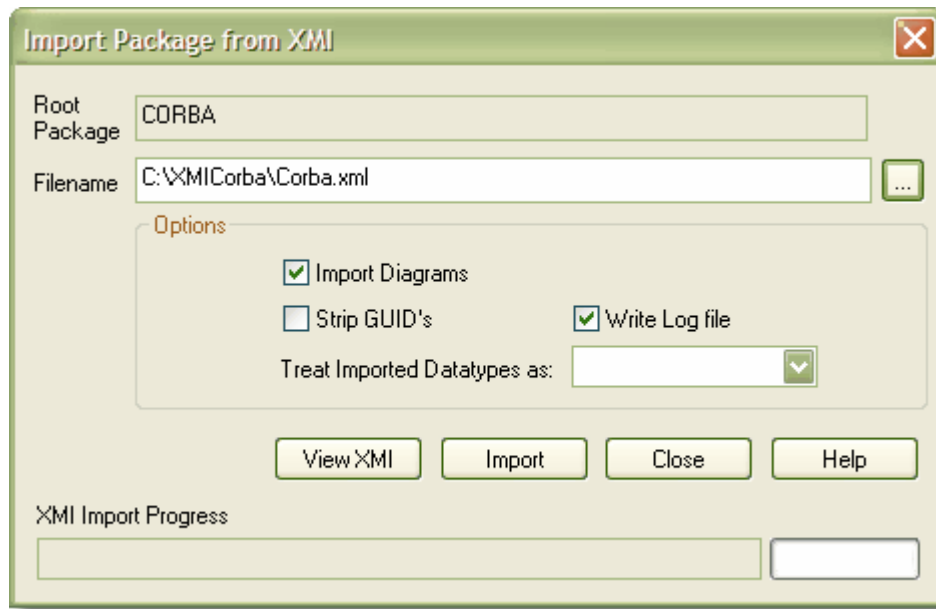
Copy a Package from One EA Model to Another

To copy a package from one EA model to another, follow the steps below:

1. Open the EA model you wish to copy from.
2. In the Project Browser, right click on the package you wish to copy.
3. Select *Export Package to XML File* from the *Import/Export* submenu. This will open the *Export Package to XML* dialog:



4. Select the appropriate options and filename in the *Export Package to XML* dialog (see [Export Package to XML](#) for further information).
5. Press *Export* to begin the export process.
6. When the export is complete, open the recipient EA model. In the Project Browser, navigate to the location you wish to import the package into.
7. Right click to view the context menu, and select *Import Package from XML File* from the *Import/Export* submenu. This will open the *Import Package from XML* dialog:



8. Select the appropriate options and filename in the *Import Package from XML* dialog (see [Import Package from XML](#) for further information).
9. The package has now been copied from the source project to the destination project.

Note: If the package you are importing already exists in the target model (has been imported previously), then you must either import over the existing one, -OR- select to Strip GUIDS - in which case you will get a replica/copy of the original. You may also use this technique to copy and entire package within the same model

7.5 Upsizing Models

The Desktop and Professional versions of Enterprise Architect use an MS JET database as the model repository. If you purchase the Corporate Edition, you have the option to use a [SQL Server](#), [MySQL](#), [Oracle 9i](#), [PostgreSQL](#), [Adaptive Server Anywhere](#) or [MSDE](#) data repository.

The process of upsizing a model is fairly straightforward and involves the following steps:

1. Install the DBMS software and create a database.
2. Run a [script supplied by Sparx Systems](#) to create the required tables.
3. Open EA and use the [Data Transfer](#) function (under the *Tools | Data Management | Data Transfer* menu) to move a model from a .EAP file to the DBMS repository.

This section details how to do this for [MySQL](#), [SQL Server](#), [Oracle9i](#), [PostgreSQL](#), [Adaptive Server Anywhere](#) or [MSDE](#).

7.5.1 Upsizing to MySQL

Before you set up EA for use with MySQL, we recommend you run the *Tools | Data Management | Data Integrity* tool on the base project you wish to upsize to MySQL. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for MySQL. Follow them in order:

Stage One: Install MySQL Components

1. Install MySQL - version 4.0.3 or higher.
2. Install MySQL ODBC 3.51 or higher.
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Set up a MySQL ODBC Driver](#).

Stage Two: Select Table Type

1. If you wish to use InnoDB tables, set up the MySQL .ini file as required and run the MySQL - InnoDB BaseModel script.
2. If you wish to use MyISAM tables, set up the MySQL .ini file as required and run the MySQL - MyISAM BaseModel script.

Note: See discussion on [MySQL limitations](#).

Note: The scripts are available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

Stage Three: Create the Database

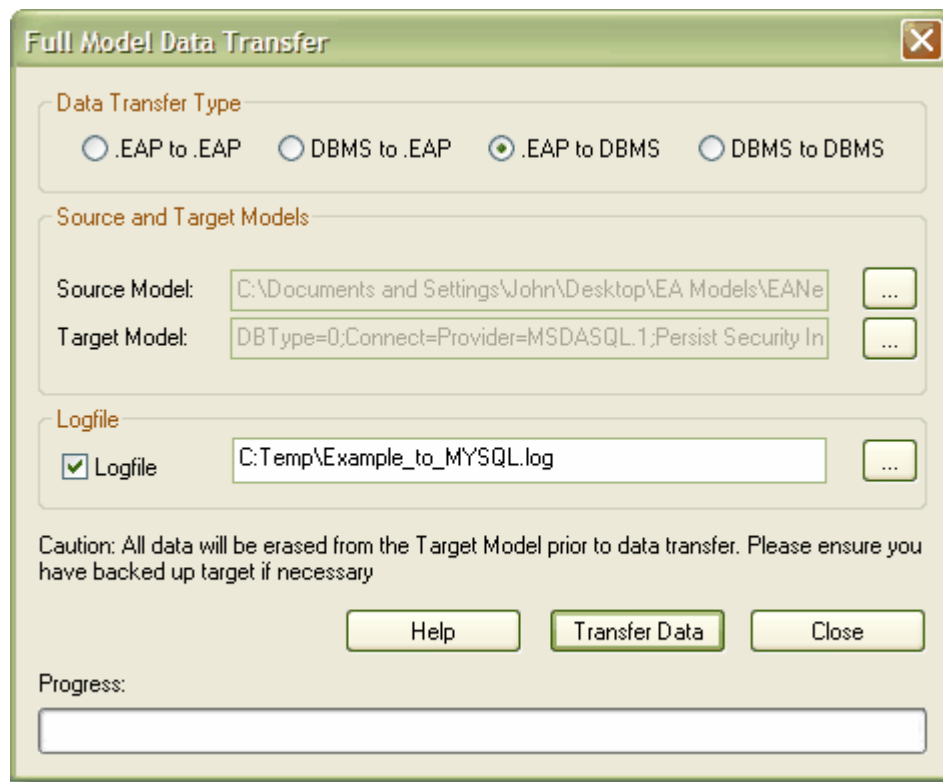
1. Create an empty database.

Note: See [Create a New MySQL Repository](#).

2. Now that have an empty database you can use the *Tools | Data Management | Data Transfer* menu option in EA to transfer an existing model into the server.

Stage Four: Upsize the Data

1. Open EA (you can press *Cancel* at the *Open Project* screen to open with no project loaded).
2. From the *Tools | Data Management* submenu, select *Data Transfer*. This will open the *Full Model Data Transfer* dialog:



3. Select **.EAP to DBMS** as the *Data Transfer Type*.
4. Enter the name of the **.EAP** file to upsize to MySQL as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press *Next*.
7. Select the ODBC Data Source you configured to point to your new database.
Note: See [Connect to a MySQL Data Repository](#) for more information.
8. Press *OK*.
9. The Connection Name dialog will appear, prompting for a name and connection type. Ensure the *MySQL* option is selected.
10. Enter a name if you wish and press *OK*. This will return you to the *Full Model Data Transfer* dialog.
11. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
12. Press *Transfer Data* to begin the data transfer process.

Once the process is completed, you will have upsized your model to MySQL and can now open it from EA.

7.5.2 Upsizing to SQL Server

Before you set up EA for use with SQL Server, we recommend you run the *Tools | Data Management | Data Integrity* tool on the base project you wish to upsize to SQL Server. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are three stages to upsizing your database for SQL Server. Follow them in order:

Stage One: Create an Empty Database

1. Install SQL Server.
2. Create an empty database.

Note: See [Create a New SQL Server Repository](#).

Stage Two: Configure the Database

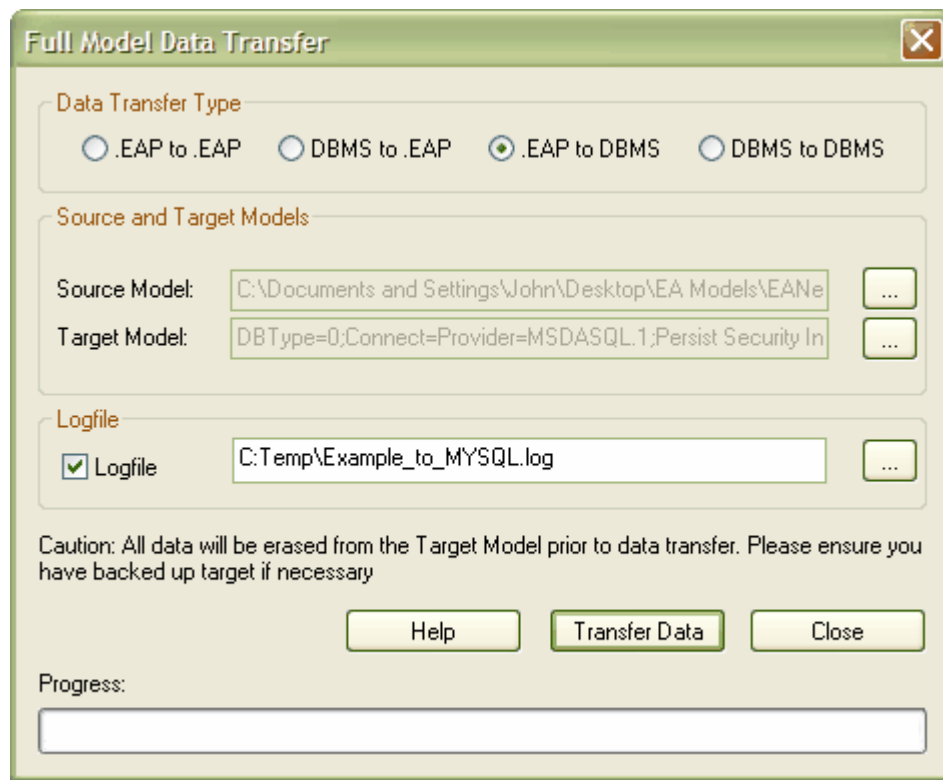
1. Using a tool such as the SQL Query Analyser, load the SQL Server - Base Model.sql file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.
2. Make sure the new database is the currently active database.
3. Run the script to create all required tables/indexes etc.

Note: See [Create a New SQL Server Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the *Tools | Data Management | Data Transfer* menu option in EA to transfer an existing model into the server.

1. Open EA (you can press *Cancel* at the *Open Project* screen to open with no project loaded).
2. From the *Tools | Data Management* submenu, select *Data Transfer*. This will open the *Full Model Data Transfer* dialog.



3. Select **.EAP to DBMS** as the *Data Transfer Type*.
4. Enter the name of the **.EAP** file to upsize to SQL Server as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Microsoft OLE DB Provider for SQL Server" from the list, then press *Next*.
7. On the *Data Source* details page of the connection dialog, enter the server name, database name, security details as required.
Note: See [Connect to a SQL Server Data Repository](#) for more information.
8. Press *OK*.
9. The Connection Name dialog will appear, prompting for a name and connection type. Ensure the *SQL Server* option is selected.
10. Enter a name if you wish and press *OK*. This will return you to the *Full Model Data Transfer* dialog.
11. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
12. Press *Transfer Data* to begin the data transfer process.

Once the process is completed, you will have upsized your model to SQL Server and can now open it from EA.

7.5.3 Upsizing to Oracle

Before you set up EA for use with Oracle9i, we recommend you run the *Tools | Data Management | Data Integrity* tool on the base project you wish to upsize to Oracle9i. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are three stages to upsizing your database for Oracle9i. Follow them in order:

Stage One: Create an Empty Database

1. Install Oracle9i.
2. Create an empty database.

Note: See [Create a New Oracle9i Repository](#).

Stage Two: Configure the Database

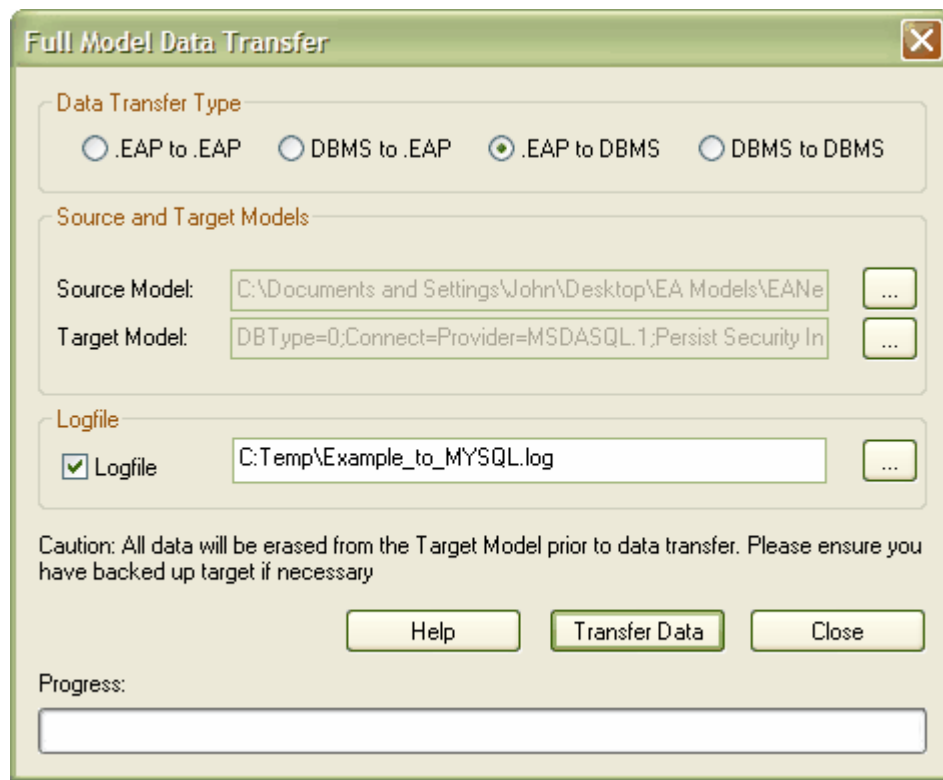
1. Using a tool such as the *SQL*Plus* or *SQL Plus Worksheet*, load the *Oracle9i_BaseModel.sql* file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.
2. Make sure the new database is selected as the current database.
3. Run the script to create all required tables/indexes etc.

Note: See [Create a New Oracle9i Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the *Project | Manage Data | Data Transfer* menu option in EA to transfer an existing model into the server.

1. Open EA (you can press *Cancel* at the *Open Project* screen to open with no project loaded).
2. From the *Tools | Data Management* submenu, select *Data Transfer*. This will open the *Full Model Data Transfer* dialog:



3. Select **.EAP to DBMS** as the *Data Transfer Type*.
4. Enter the name of the **.EAP** file to upsize to Oracle as the *Source Model*.
5. Press the Browse [...] button at the right of the *Target Model* field. This will open the *Datalink Properties* dialog.
6. Select "Oracle Provider for OLE DB" from the list, then press *Next*.
7. On the *Data Source* details page of the connection dialog, enter database name and security details as required.

Note: See [Connect to an Oracle9i Data Repository](#) for more information.

8. Press *OK*.
9. The Connection Name dialog will appear, prompting for a name and connection type. Ensure the *Oracle9i* option is selected.
10. Enter a name if you wish and press *OK*. This will return you to the *Full Model Data Transfer* dialog.
11. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
12. Press *Transfer Data* to begin the data transfer process.

Once the process is completed, you will have upsized your model to Oracle9i and can now open it from EA.

7.5.4 Upsizing to PostgreSQL

Before you set up EA for use with PostgreSQL, we recommend you run the *Tools | Data Management | Data Integrity* tool on the base project you wish to upsize to PostgreSQL. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for PostgreSQL. Follow them in order:

Stage One: Install PostgreSQL Components

1. Install PostgreSQL - version 7.3.2 or higher.
2. Install psqLODBC - version 7.03.01.00 or higher.
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Set up a PostgreSQL ODBC Driver](#).

Stage Two: Configure the Database

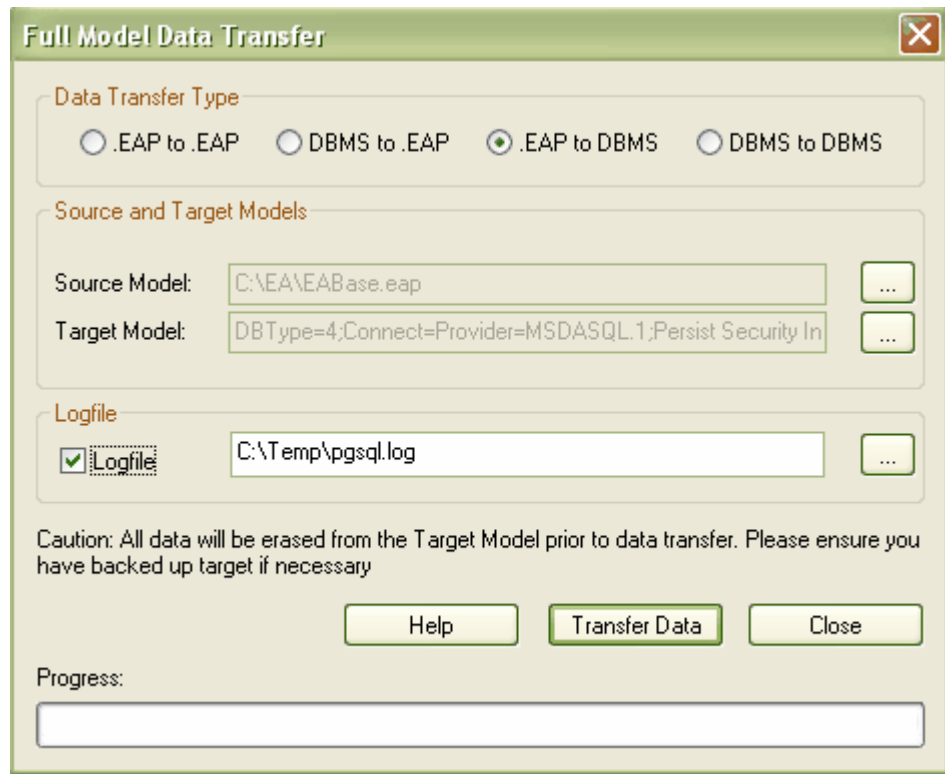
1. From the psql command line, or using a tool such as EMS PostgreSQL Manager, load the Postgres_Basemodel.sql file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.
2. Run the script to create all required tables/indexes etc.

Note: See [Create a New PostgreSQL Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the *Tools | Data Management | Data Transfer* menu option to transfer an existing model into the server.

1. Open EA.
2. Go to the *Tools | Data Management | Data Transfer* menu option. This will open the *Full Model Transfer Dialog*.



3. Select **.EAP to DBMS** as the *Data Transfer Type*.
4. Click the Browse **[...]** button to the right of the *Source Model* and locate the **.EAP** file to upsize to PostgreSQL.
5. Click the Browse **[...]** button to the right of the *Target Model*. This will open the *Datalink Properties* Dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press the **Next** button.
7. From the *Use data source name* drop down list, select the ODBC Data Source you configured to point to your new database.

Note: See [Connect to a PostgreSQL Data Repository](#) for more information.

8. Press **OK**.
9. The *Connection Name* dialog will appear, prompting for a name and connection type. Ensure the **PostgreSQL** radio button is selected.
10. Enter a name if you wish and press **OK**. This will return you to the *Full Model Data Transfer* dialog.
11. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
12. Press the **Transfer Data** button to begin the data transfer process.

Once the process is completed, you will have upsize your model to PostgreSQL and can now open it from EA

7.5.5 Upsizing to Sybase Adaptive Server Anywhere (ASA)

Before you set up EA for use with ASA, we recommend you run the [Tools | Data Management | Data Integrity](#) tool on the base project you wish to upsize to ASA. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

There are four stages to upsizing your database for ASA. Follow them in order:

Stage One: Install ASA Components

1. Install Adaptive Server Anywhere - SQL Anywhere Studio 8 or higher. This will also install the ASA ODBC driver.
2. Create a new database for the EA repository using Sybase Central.
3. Create a suitable ODBC Data Source to point to your new database.

Note: See [Set up an ASA ODBC Driver](#).

Stage Two: Configure the Database

From Sybase Central, right click on the newly created database and open Interactive SQL. Load the ASA8_BaseModel.sql file. This is available to registered users on the Corporate Edition resources page of the Sparx website at http://www.sparxsystems.com.au/registered/reg_ea_corp_ed.htm.

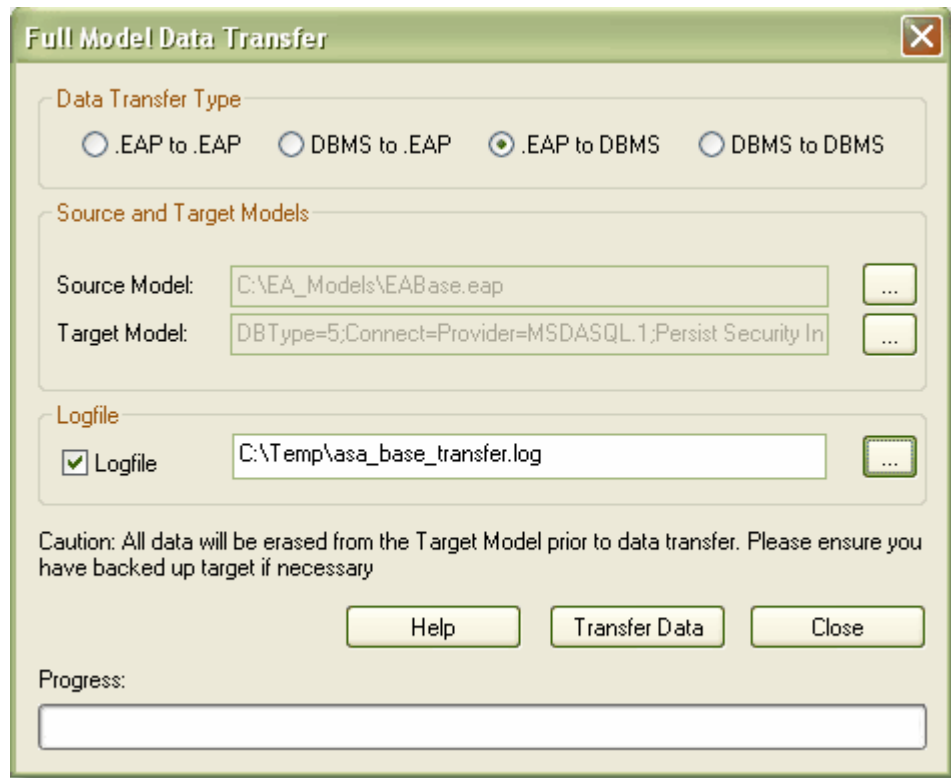
Run the script to create all required tables/indexes etc.

Note: See [Create a New ASA Repository](#).

Stage Three: Transfer the Data

You now have an empty database - you can use the [Tools | Data Management | Data Transfer](#) menu option to transfer an existing model into the server.

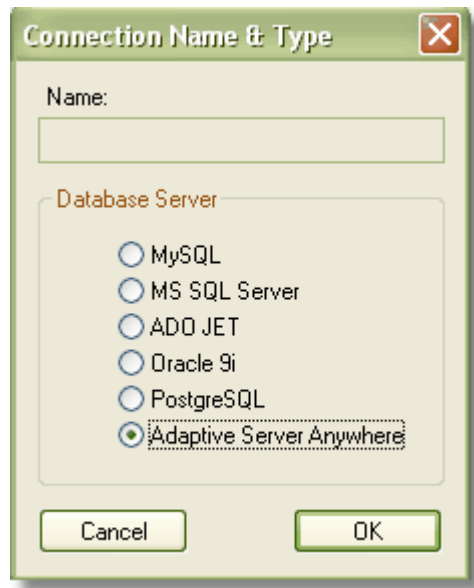
1. Open EA.
2. Go to the [Tools | Data Management | Data Transfer](#) menu option. This will open the [Full Model Transfer Dialog](#).



3. Select **.EAP to DBMS** as the *Data Transfer Type*.
4. Click the Browse [...] button to the right of the *Source Model* and locate the .EAP file to upsize to ASA.
5. Click the Browse [...] button to the right of the *Target Model*. This will open the *Datalink Properties* Dialog.
6. Select "Microsoft OLE DB Provider for ODBC Drivers" from the list, then press the *Next* button.
7. From the *Use data source name* drop down list, select the ODBC Data Source you configured to point to your new database.

Note: See [Connect to a ASA Data Repository](#) for more information.

8. Press *OK*.
9. The *Connection Name* dialog will appear, prompting for a name and connection type. Ensure the *Adaptive Server Anywhere* radio button is selected.



10. Enter a name if you wish and press **OK**. This will return you to the *Full Model Data Transfer* dialog.
11. If desired, check the *Logfile* checkbox and enter a path for the data transfer log file.
12. Press the *Transfer Data* button to begin the data transfer process.

Once the process is completed, you will have upsized your model to Adaptive Server Anywhere and can now open it from Enterprise Architect.

7.5.6 Upsizing to SQL Server Desktop Engine (MSDE)

Before you set up EA for use with MSDE, we recommend you run the *Tools / Data Management / Data Integrity* tool on the base project you wish to upsize to MSDE. This will ensure data is 'clean' before uploading.

Warning: Before proceeding, ensure MDAC 2.6 or 2.7 is installed on your system.

Upsizing Your Database

Follow the steps in [Upsizing to SQL Server](#) to upsize your model to MSDE.

7.6 Model Maintenance

This section highlights some of the administrative functions you may need to carry out to maintain your model.

See:

- [Rename a Project](#)
- [Compact a Project](#)
- [Repair a Project](#)

7.6.1 Rename a Project

Important: The only way to rename an Enterprise Architect project is at the Windows file system level.

To rename an EA project, follow these steps.

1. If you have the project open, shut it down.
2. Ensure no other users have the file open.
3. Open Windows Explorer and navigate to the project.
4. Rename the project file using Windows Explorer.
5. You should *keep the ".EAP" extension the same* to preserve compatibility with the default project type as installed in the registry at installation time.

7.6.2 Compact a Project

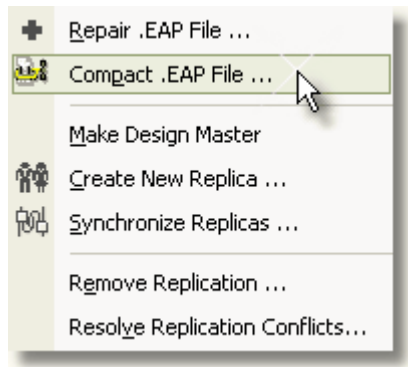
After some time, a project may benefit from *compacting* to conserve space.

Note: *Compacting will shuffle the contents of the model around, eliminating unused space and generally reducing the size of your model file.*

Compact a Project

To compact a project, follow these steps:

1. Ensure that no users have the target project open.
2. Select the **Tools | Manage .EAP File | Compact .EAP File** menu option to compact the selected project.



3. Follow the on-screen instructions to complete the process.

Warning: Always compact and repair projects on a *local* drive, never on a network drive.

7.6.3 Repair a Project

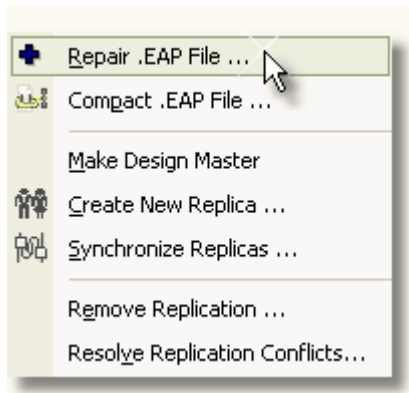
If a project has not been closed properly, often caused by system or network outages, on rare occasions the .EAP file will not open correctly. In this case you will get a message informing you the model is of an unknown database format or is not a database file.

Note: Poor network connections may also cause this symptom.

Repair a Project that has not closed correctly

To repair a project which was not closed properly, follow these steps:

1. Copy the project file to a local drive on your PC.
2. Open Enterprise Architect, but when you are prompted to open a project, press **Cancel**.
3. Select the **Tools | Manage .EAP File | Repair .EAP File** menu item.



4. Follow the on-screen instructions.

Note: All users must be logged off the project you are attempting to repair.

Warning: Never attempt to repair a project over a network connection: copy it to a local drive first.

Ensuring the Integrity of the Repaired Project

An additional step which you can use to ensure the integrity of your model is use the "Remove Replication" feature.

1. Open Enterprise Architect, but when you are prompted to open a project, press **Cancel**.
2. Select the **Tools | Manage.EAP File | Remove Replication** menu item.
3. Follow the prompts, when you are prompted for the **Replica Project** browser for your problem project, you may be given a warning about the project not being the "Design Master", accept this warning . Click **Next**.
4. Browse for the clean project (i.e. EABase.eap). Click **Next**.
5. Enter the path and name of the new project to created, then click **Next**.
6. Click **Run** and the removal process will be run.

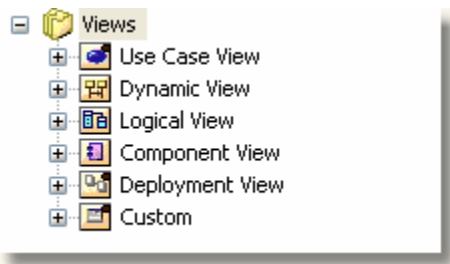
Once the removal process has been completed, open the project and do a check of the project contents. If the data is intact, backup the old project and replace it with the new version.

7.7 Manage Views

The top level packages in EA are referred to as **Views**. This terminology is used simply to designate that the package is at the top level and may be used to subdivide a project into partitions such as Business Process, Logical Model, Dynamic View, etc.

There are 6 main views:

- [Use Case View](#) - eg. [Use Case diagram](#), [Analysis diagram](#), [Robustness diagram](#)
- [Dynamic View](#) - eg. [Activity diagram](#), [Communication diagram](#), [Sequence diagram](#), [State diagram](#)
- [Logical View](#) - eg. [Class Model](#), [Code Engineering](#), [Data Model](#)
- [Component View](#) - eg. [Component diagram](#)
- [Deployment View](#) - eg. [Deployment diagram](#)
- [Custom View](#)



You may wish to rename these views, move them into a different order, or delete the provided views and create your own. Do this by right clicking the mouse on the selected View to open the context menu, and choose whether to rename, move or delete the view.

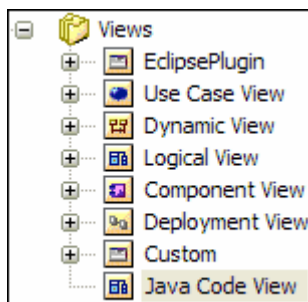
See also: [Add Additional Views](#), [Rename Views](#), [Delete Views](#)

Note: Some views have a special icon - if you delete that view, it cannot be recreated with the special icon - although you can certainly create a new View and give it the name of the deleted view eg. Logical View.

7.7.1 Add Additional Views

You may add additional packages at the **View** level. These will appear at the end of the standard views and be displayed in creation order. The icon for user created views is the same as for the Custom view. User views are a good way to extend the standard model on a per project basis depending on specific requirements and modeling techniques.

The example below shows an additional view called "Java Code View" which has been appended to the main Views list.

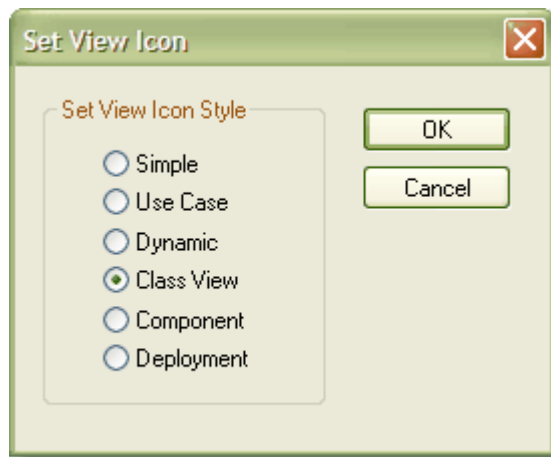


Note: You can delete a user created view. If you do so, ALL contents of the view are deleted as well, so take care.

Create a User View

To create a user view, follow these steps:

1. Right click on the root element of the Project Browser (named *Views*).
2. Select the *New View* option from the menu.
3. Give your view a name and press *OK*.
4. The *Set View Icon* dialog will then allow the user to specify the view icon, select the most appropriate one from the list and then press the *OK* button.



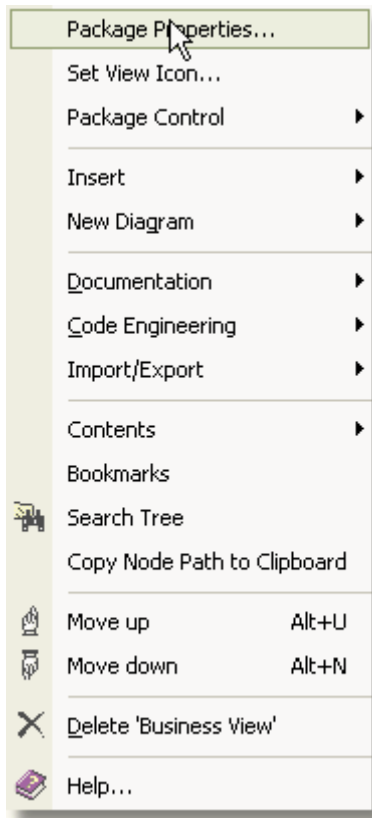
7.7.2 Rename Views

You can rename a view if you wish.

Rename a View

To rename a view, follow these steps:

1. Right click on the view in the Project Browser. Select the *Package Properties* option.



2. Enter the new name and press **OK**.

The image shows a software dialog box titled "Package : Business View". It has several tabs: "General", "Require", "Constraints", "Link", "Scenario", and "Files". The "General" tab is selected. The dialog contains the following fields and controls:

- Name: Business Modeling View
- Stereotype: [dropdown]
- Abstract:
- Author: [dropdown]
- Status: Proposed [dropdown]
- Scope: Public [dropdown]
- Complexity: Easy [dropdown]
- Alias: [text box]
- Language: C++ [dropdown]
- Keywords: [text box]
- Phase: 1.0
- Version: 1.0
- Note: [text area]

At the bottom of the dialog are four buttons: "Apply", "OK", "Cancel", and "Help".

7.7.3 Delete Views

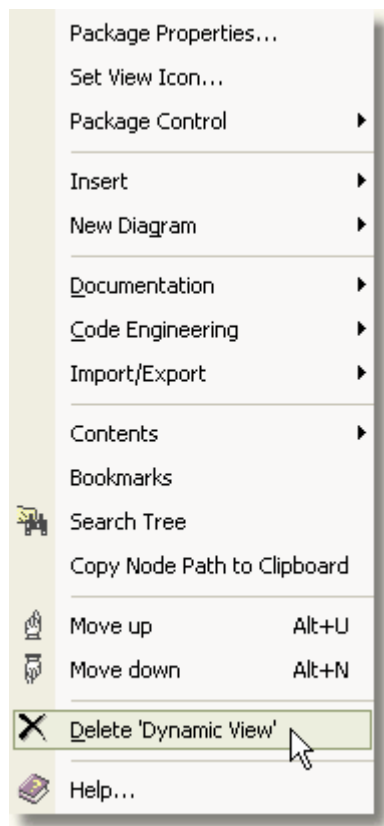
You can delete a view if you wish.

Warning: If you delete a view, all its contents will be deleted at the same time. It CANNOT be restored.

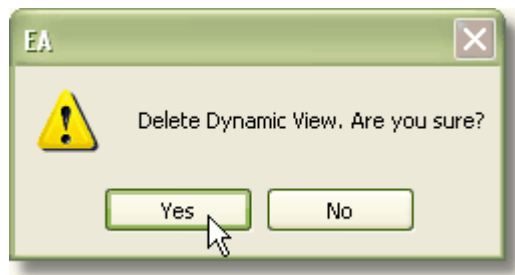
Delete a View

Follow these steps if you wish to delete a view.

1. Right click on the view you wish to delete in the *Project Browser*.
2. Select *Delete* from the context menu.



3. You will see the following warning:



4. If you wish to delete the view and its contents, press **Yes**. If not, press **No**.

7.8 XMI Import and Export

What is XMI?

XML Metadata Interchange (XMI) is an open standard file format, intended to allow the interchange of model information between models and tools. XMI is based on XML, and is defined by the OMG. Enterprise Architect uses XMI as a method of importing and exporting model specifications between different UML packages, EA projects, and other tools that support XMI.

Enterprise Architect supports the XMI 1.1 and 1.2 specifications, but does not fully support the older 1.0 specification. When importing/exporting to XMI 1.0, some loss of data will occur due to the limitations of XMI 1.0. XMI Version 1.1 has support for UML 1.3.

With XMI the transfer of model details may be exchanged between different UML tools and other tools that are capable of using XMI. Limited support for export to Rational Rose is provided using the Rose version of the XMI 1.1 specification, as implemented by Unisys for Rational products. Packages may be [exported](#) from and [imported](#) into EA models. This greatly improves the flexibility and robustness of EA models by allowing Analysts and Modelers to externalize model elements in XMI for [version control](#), distributed development, post processing and transferring packages between models. When performing EA-to-EA transfers, ensure that the XMI version selected is 1.1.

See:

- [XML Options](#) configure the XML options. XMI import/export and package control all rely on saving and loading XML files. There are some options you can set to streamline this process.
- [Export a package](#) to XMI in XMI 1.0, XMI 1.1 and XMI 1.2
- [Import from XMI](#) with support for XMI 1.0, XMI 1.1 and XMI 1.2
- [XMI controlled packages](#)
- [Manually control a package](#) by linking it to an XMI file
- [Batch export](#) controlled packages
- [Batch import](#) controlled packages
- [Limitations of XMI](#)
- [The UML DTD](#)

Important Note: The `UML_EA.DTD` file *MUST* be present in the output directory where XML files are written when the **Use DTD** option has been selected when performing an [XMI export](#). If it is absent an error will occur on trying to read or write an XML file.

Important Note: When you import an XML file over an existing package - ALL information in the current package is deleted first. Please make sure you do not have important changes you do not want to lose before you do this.

For further information on XMI, including specifications, consult the OMG's [XML/XMI Technology page](#)

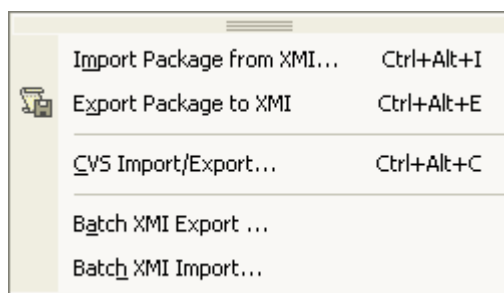
7.8.1 Export to XMI

A Package may be exported to an XMI (XML based) file. This allows EA Model elements to be moved between models, for [distributed development](#), [manual version control](#) and other benefits. It also allows limited export of EA model elements to Rational Rose and other tools which implement the UML1.3 XMI 1.1 / XMI 1.0 standard. For more information regarding the limitations of XMI exporting read the [Limitations of XMI](#) topic.

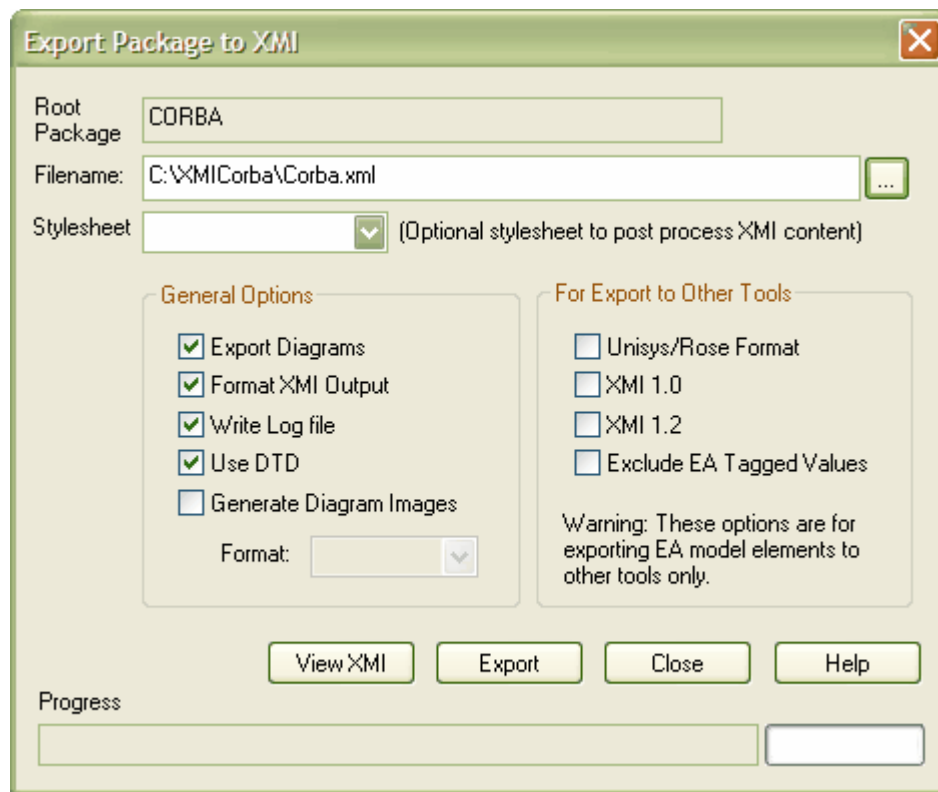
Exporting a Package to XML

To export a package to XML, follow these steps:

1. In the Project Browser, select the package you want to export.
2. Right click and select the **Import/Export** menu OR go to the **Project | Import/Export** submenu. Select the **Export Package to XMI** option.



3. The **Export Package to XMI** dialog will appear.



4. Set the required options - you can configure the following:
 - Filename - where to output the XML file - must be a valid directory/path name.
 - Stylesheet - select a stylesheet to post-process XML content before saving to file.
 - Export diagrams yes/no - check to export diagrams.
 - Use Unisys Rose Format - check to export in Rose UML 1.3, XMI 1.1 format.
 - Format XML output - yes/no - check to format output into readable XML (takes a few additional seconds at end of run).
 - Write log file - check to write a log of export activity (recommended). The log file will be saved in the same directory exported to.
 - Use DTD - check to use the UML1.3 DTD (recommended). Using this option will validate the correctness of the model and that no syntactical errors have occurred. For more information regarding the use of DTD's read the [UML DTD](#) topic.
 - XMI 1.0 - check this to generate output in XMI 1.0 format.
 - XMI 1.2 - check this to generate output in XMI 1.2 format.
 - XMI 1.1 - is the default output, to generate output in XMI 1.1 format level the XMI 1.0 and XMI 1.2 checkboxes unchecked.
 - Exclude EA Tagged Values, excludes EA specific information from the export to other tools.
5. Press **Export**.

Important Note: When exporting and importing with XMI 1.0 with EA some loss of data will occur due to the limitations of XMI 1.0.

Important Note: The `UML_EA.DTD` file **MUST** be present in the output directory where XML files are written when the Use DTD option has been selected when performing an XMI export. If it is absent an error will occur on trying to read or write an XML file.

7.8.2 Import from XMI

A Package may be imported from an XMI (XML based) file. This allows EA Model elements to be moved between models, for distributed development, [manual version control](#) and other benefits.

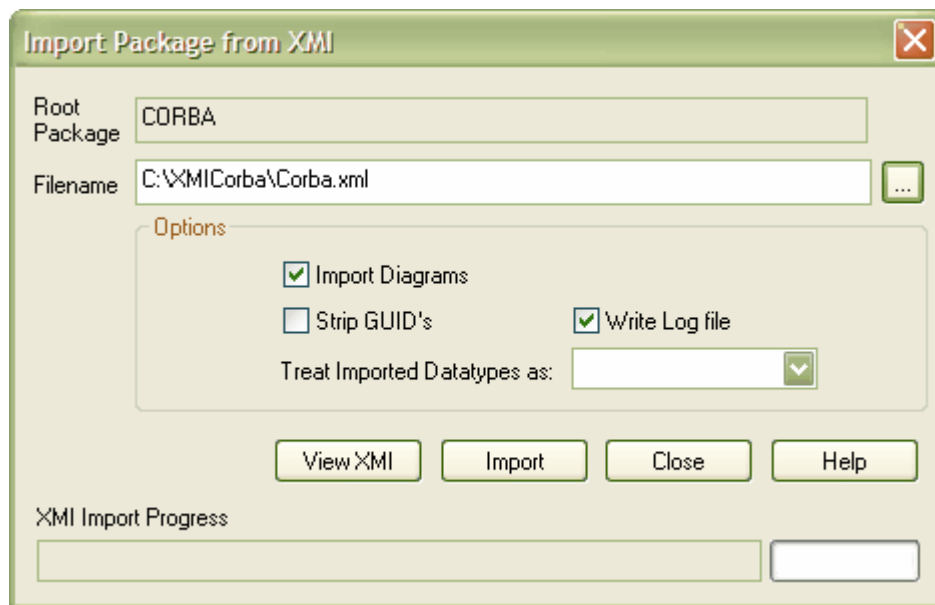
Import From XML

To import a package from XML, follow the steps below:

1. In the Project Browser, select the package you want to import.
2. Right click and select the *Import/Export* menu OR go to the *Project | Import/Export* menu. Select the *Import Package from XMI* option.



3. The *Import Package from XMI* dialog will appear.



4. Set the required options:
 - Filename - where to load the XML file from - must be a valid directory/path name.
 - Import diagrams - check to import diagrams.
 - Strip GUIDS - check this to remove Universal Identifier information from the file on import. This permits the import of a package twice into the same model - the second import will require new GUIDS to avoid element collisions
 - Write log file - check to write a log of import activity (recommended). The logfile will be saved in the same directory imported from.
5. Press *Import*.

Important Note: When you import an XML file over an existing package - ALL information in the current package is deleted first. Please make sure you do not have important changes you do not want to lose before you do this.

7.8.3 Limitations of XMI

While XMI is a valuable means of specifying a UML model in a common format, it is relatively limited in the amount of additional information it will tolerate using the standard syntax. A lot of information from an EA Model must be converted to what are called 'Tagged Values' - which will import into other modeling systems as additional information or be ignored completely.

EA can both generate and read XMI 1.0, 1.1 and 1.2, using UML 1.3 format. Note that round tripping model elements using XMI (for example to version control or for controlled package) is only possible using XMI 1.1/UML 1.3 - EA format which uses the additional tagged values to store the UML 2.0 information.

Notes for Exporting to Rose and other tools

- There are also discrepancies in the Unisys/Rose implementation with regard to spelling mistakes and slightly different syntax to the official XMI 1.1 specification - so problems may occur.
- The way packages are arranged in different models may impact the successful import into other systems, experimentation is the only work around for this problem.
- Some parts of the XMI import/export process do not work as expected in products like Rational Rose - for example, note links are not supported, state operations import but do not appear in diagrams and some other issues. In addition, Rational Rose only supports import of a full project - not a single package
- It is recommended that for best results you keep the model elements you wish to export to Rose simple and conforming as closely as possible to the UML 1.3 specification.

7.8.4 The UML DTD

When you import or export EA packages to XML, the import or export process may be validated using a DTD or Data Type Definition. This document is used by the XML parser to validate the correctness of the model and that no syntactical errors have occurred. It is always best to use a DTD when moving packages between EA models as it ensures correctness of the XML output, and prevents attempted imports of incorrect XML.

Several DTD's for XMI/UML exist. The OMG defines a standard UML1.3 DTD for use in XMI 1.1. Enterprise Architect uses an extension to this with some additional element extensions for non-standard UML types - such as testing details.

Whenever you write or read an XML file, the XML parser looks in the current directory for the DTD specified in the DOCTYPE element in the XML file itself. If the DTD cannot be found, an error occurs and processing aborts. You must ensure the UML_EA.DTD file is in the current XML output path (generated by default).

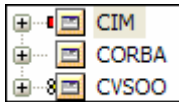
7.8.5 Controlled Packages

Controlled packages are a powerful means of 'externalizing' parts of an EA model. Using controlled packages you can:

- Support widely distributed development by having team members submit packages in the form of XML for

- import into a central EA repository
- Support version control by writing model elements in XML text files, suitable for version controlling using standard version control software. Using XMI in this manner allows users to manually connect to 3rd party version control software outside of the EA environment, EA internally supports the configuration of version control via SCC and CVS.
- Support import/export of model elements between different models - for example a class library may be re-used in many models, and kept up to date in target models using controlled packages - reloading packages as required when new versions of the class model become available.
- Package XML is standard XMI compliant output which may be loaded into any XML viewer, or used by any XML based tool to perform manipulations and extracts - such as document or code generators.

Controlled packages appear in the browser with a small red rectangle to the left of the package icon as in the example below:

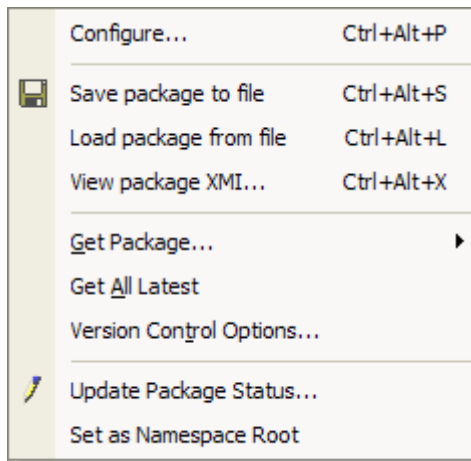


A controlled package is one configured to save and load in XML format to a named file. The XML output is UML1.3 compliant XMI, with Enterprise Architect extensions to support diagrams and additional model elements.

Important Note: The *UML_EA.DTD* file **MUST** be present in the output directory where XML files are written when the *Use DTD* option has been selected when performing an *XMI export*. If it is absent an error will occur on trying to read or write an XML file. This file contains the definition for UML1.3 XMI output based on UML models.

7.8.5.1 Controlled Package Menu

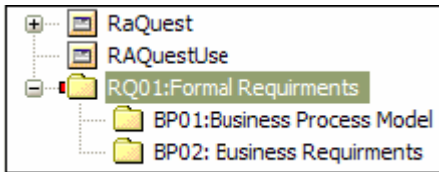
The *Controlled Package* menu is found by right clicking on a version controlled package in the Project Browser and selecting *Package Control*.



Menu Item	Functionality
Configure	Shows the package control dialog which allows you to specify whether this package (and its children) is controlled, and which file it is controlled through.
Save Package to file	Saves a controlled package to an XMI file.
Load package from file	Loads a previously saved XMI file.
View package XMI..	Allows the user to view the package XMI after the package has been exported to XMI.
Get Package	Allows the user to gain access from packages in the version controlled repository that is in currently available in the users model.
Get All Latest	Retrieves the latest version of the package from the repository. Available only for packages which are checked in. Get Latest is not intended for sharing .EAP files and should only be used when people have their own individual databases.
Version Control Options	Displays the Version Control Options dialog .
Update Package Status	Allows the user to provide a bulk update on the status of a package, this includes status options such as Proposed, Validate, Mandatory etc.
Set as Namespace Root	Sets the namespace root for languages which support namespaces, for more information see the Namespaces section.

7.8.5.2 Configure Packages

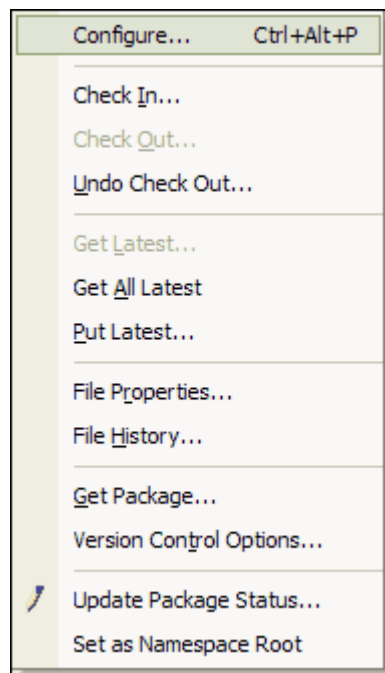
Before you can use a Controlled Package, you must configure it with some options. These include the filename to save to/load from, the type of export, version number, etc. Once a package is configured and marked as controlled, it will appear in the Project Browser with a small red rectangle next to the package icon - indicating it is a controlled package. The image below shows one controlled package called "RQ01: Formal Requirements":



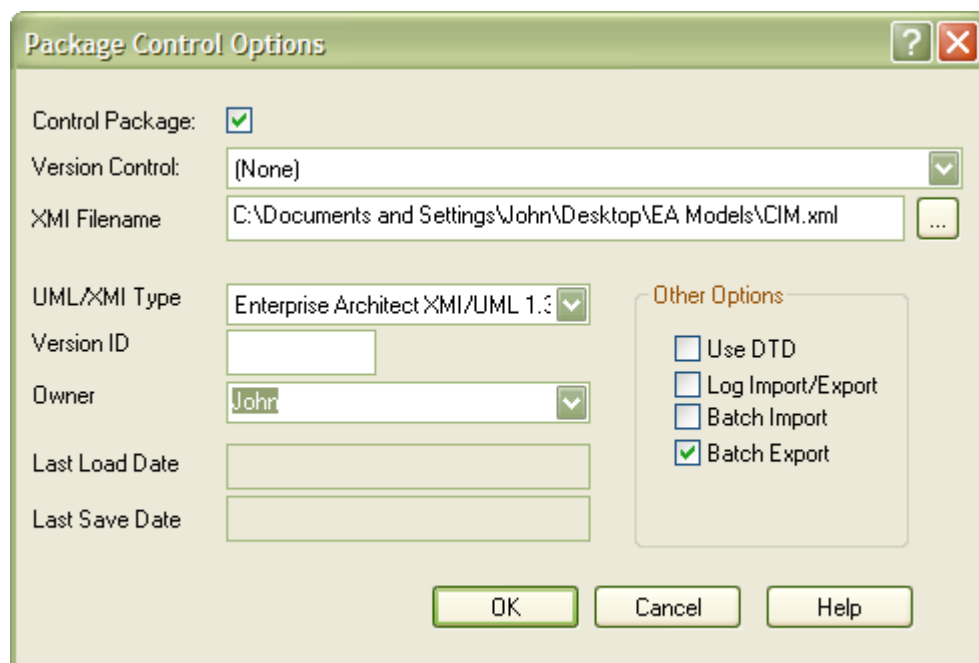
Configure a Controlled Package

To configure a controlled package, follow the steps below:

1. In the Project Browser, right click on the package you wish to control or configure.
2. From the *Package Control* submenu, select *Configure*.



3. The *Package Control Options* dialog will be displayed.



4. Set the required options - you can configure the following:
 - Tick the *Control Package* checkbox to indicate this is a controlled package
 - Select the *Version Control* repositories from the *Version Control* dropdown list, this connects this package to a specific version control configuration.
 - Set the *XMI Filename* for import/export
 - Set the *UML/XMI Type* to determine the type of XMI generated, this includes Enterprise Architect XMI/UML 1.3, Rational Rose/Unisys UML 1.3 and Generic XMI 1.0/UML 1.3. Currently only Enterprise Architect UML1.3 is supported for complete import/export round tripping of packages

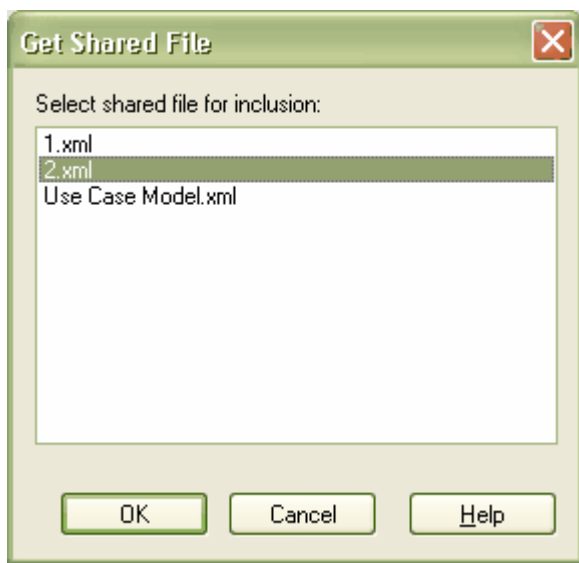
- Set the *Version ID* (manual process)
- Set the package *Owner*
- Set whether to *Use a DTD*; and
- Set whether to *Log Import/Export* activity to a log file
- Whether the package is marked as a *Batch Import* package
- Whether the package is marked as a *Batch Export* package

Note: When you load a controlled package from file - ALL the current model contents are first deleted, before the file contents are loaded in their place. Please make sure you are certain you wish to revert to a previous version before continuing.

Note: For batch import, the file date of the XML file is stored and the batch import can be bypassed if the file date of the last import matches that of the current file (ie. there is no change).

7.8.5.3 Get Shared File from Repository Dialog

The *Get Shared File* dialog allows you to select a version controlled XML file for inclusion in your project. Normally you would receive instructions from the creator as to the name of the file which you would then select from this list. Once you have included a package from this list in your model it no longer appears in the list of files to be retrieved. To access this dialog the user must one or more version controlled configurations.

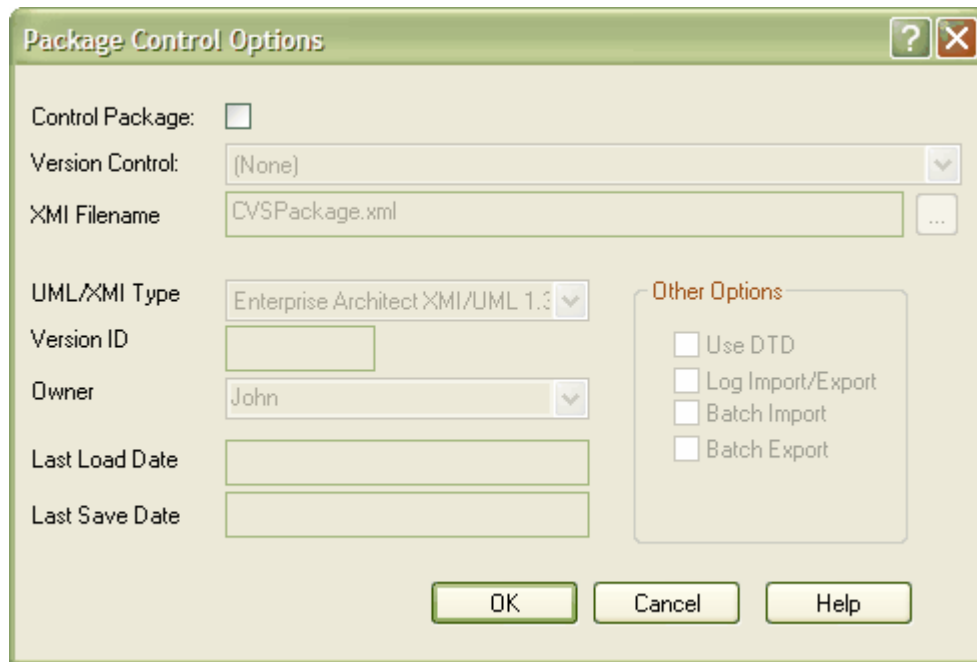


7.8.5.4 Disconnect Controlled Package.

It is possible to remove the control from a package. Before removing the package control the package must first be [checked in](#) if it is being used for version control. .

Remove Package Control

1. In the *Project Browser*, right click on the controlled package.
2. From the *Package Control* submenu, select *Configure* or press the *Ctrl + Alt + P* hotkey combination.
3. Uncheck the *Control Package* checkbox.

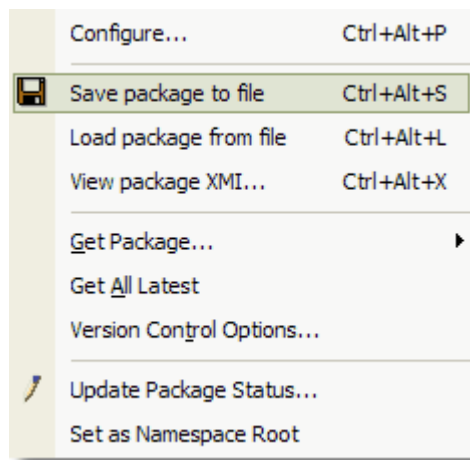


4. Press **OK** to remove package control.
5. The package control for the selected package will now be removed.

7.8.5.5 Save a Package

You can Save a controlled package to an XMI file. Once the package has been correctly [configured](#), follow the steps below:

1. In the Project Browser, right click on the package you wish to save.
2. From the **Package Control** submenu, select **Save Package to File**.



3. The export process will occur automatically according to your configured preferences, overwriting any existing file.

Note: If you are using a version control package in conjunction with the exported package files, you will need to check out the XMI file first to allow EA to overwrite the existing version.

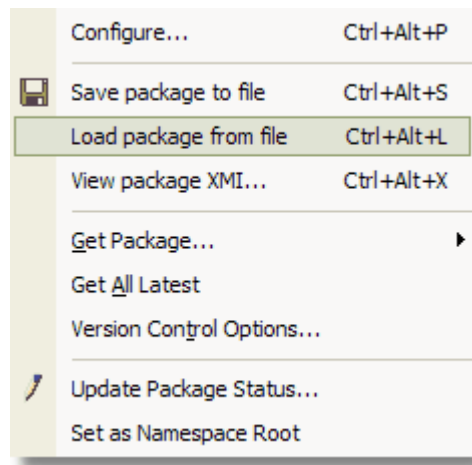
7.8.5.6 Load a Package

Using the Controlled Packages feature you can Save and Load packages to a named file. If a package has been marked for control it will appear with a red rectangle in the left of the package icon in the project view. If you have previously saved a controlled package, you may reload it using the Load Package feature.

Load a Controlled Package

To load a controlled package, follow the steps below:

1. In the Project Browser, right click on the package you wish to load.
2. From the *Package Control* submenu, select *Load Package from File*.



3. If you have configured the package control details you will be asked to confirm the import.

Warning: Importing deletes the current package entirely from the model, and the action cannot be undone - so care must be taken not to lose any current changes or information.

4. Press **Yes** to confirm the import. This will delete the current package and proceed with importing the saved package.

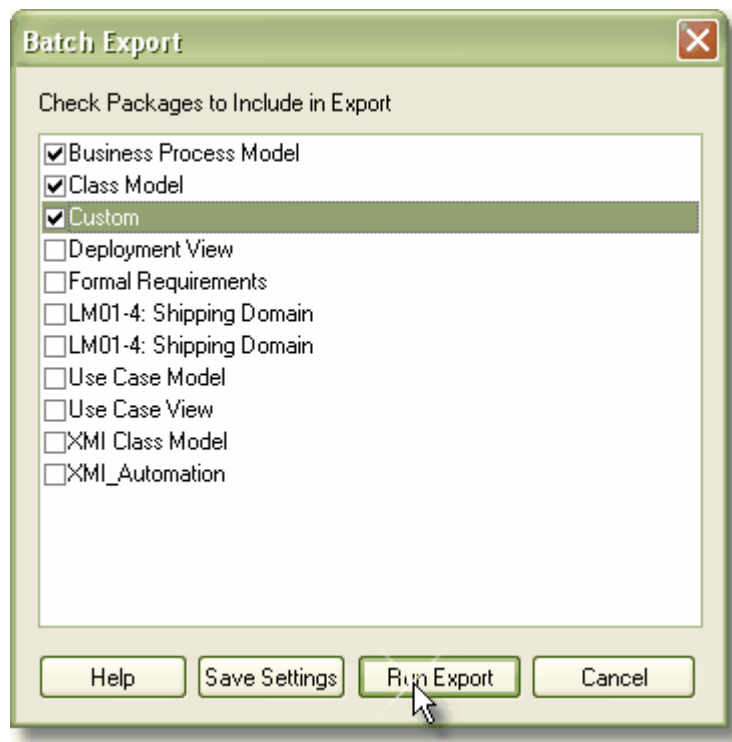
7.8.5.7 Batch XMI Export

A group of controlled packages can be exported in one step, using the Batch XMI Export function.

Batch XMI Export

To export a group of controlled packages, follow the steps below:

1. From the *Project | Import/Export* submenu, select *Batch XMI Export*.



2. In the *Batch Export* dialog, check the packages to include in this export run.
3. Press *Save Settings* if you wish to save this configuration as the default.
4. Press *Run Export*.

EA will cycle through each checked package and export using the options specified in the [Controlled Package](#) dialog. As long as a valid filename exists, EA will export the package to XML and proceed to the next.

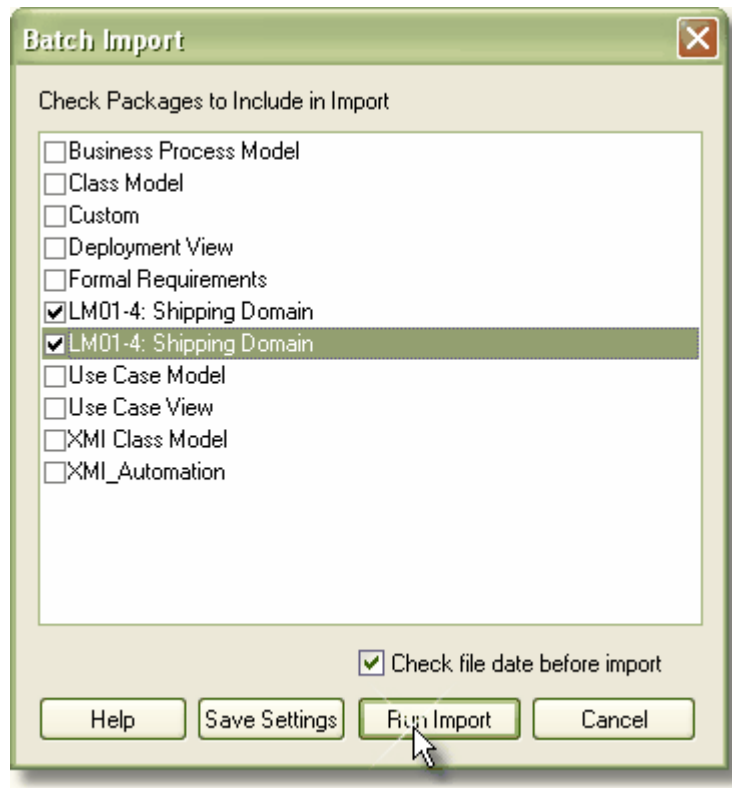
7.8.5.8 Batch XMI Import

A group of controlled packages can be imported in one step, using the Batch XMI Import function.

Batch XMI Export

To export a group of controlled packages, follow the steps below:

1. From the *Project | Import/Export* submenu, select *Batch XMI Import*.



2. In the *Batch Import* dialog, check the packages to include in this import run.

*Tip: If the **Check file date before import** checkbox is ticked, EA will not import a file if the last import file date matches that of the one currently on disk (thus avoiding re-importing the same module multiple times).*

3. Press *Save Settings* if you wish to save this configuration as the default.
4. Press *Run Import*.

7.8.5.9 Manual Version Control with XMI

XMI may be used to support version control by writing model elements in XML text files, suitable for version controlling using standard version control software. Using XMI in this manner allows users to manually connect to 3rd party version control software outside of the EA environment. EA internally supports the configuration of version control via SCC and CVS configurations. To use XMI for version control, the following conditions must be met:

1. Select suitable packages in the Project Browser that will be marked as controlled packages.
2. Configure these with filenames that are visible to a version control system of your choice.
3. Save the controlled packages to establish a model base and check these into the version control system.

When Versioning is Required

Continue working on a package until versioning is required then follow these steps:

1. Check out the package XMI file from the version control system.

2. Save the relevant package using the controlled package support.
3. Check the package back into the version control system

Recover an Earlier Version

To recover an earlier version, follow these steps:

1. Save the current version first if required (important - because the package will be completely deleted during the import process) - and manually update version control system if necessary.
2. Get the required package version from the version control system.
3. Select the package to reload.
4. Use the Package Control/Load Package menu option to import the previous version.
5. EA will delete the controlled package and restore the previous version.

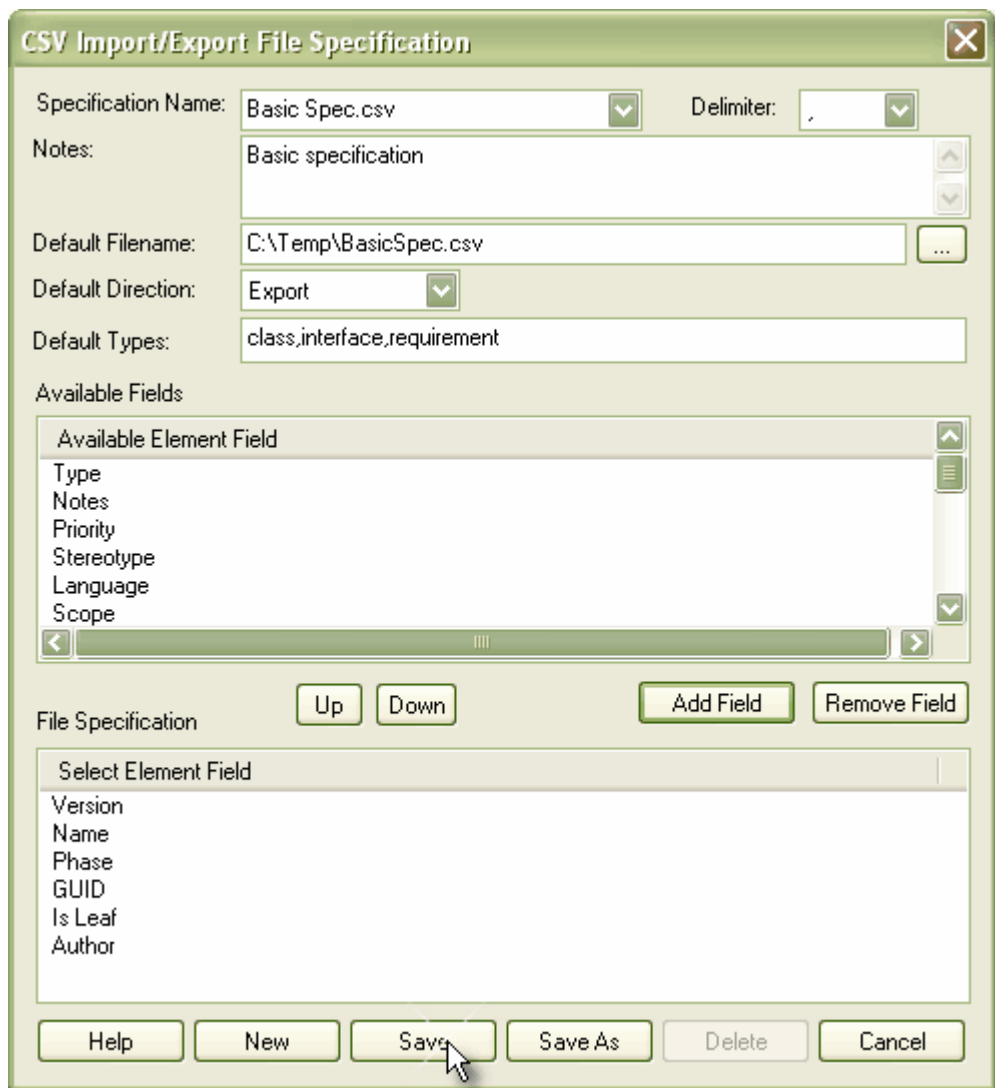
7.9 CSV Import and Export

You can [import](#) and [export](#) information about EA elements in CSV format. You will need to define [CSV specifications](#) to do this.

7.9.1 CSV Specifications

To [import](#) and [export](#) element data from EA using CSV files, you need to first setup one or more file specifications. A file specification lists the fields in the order they will be imported or exported, the filename (optional) and the delimiter between columns. Once you have defined one or more specifications, it can be selected in the CSV import/export dialog as the current specification to apply during an import or export action. CSV will import and only objects and their properties, items such as class attributes cannot be imported or exported through this mechanism. [XML](#) provides a solution to this limitation as does use of the [Automation Interface](#).

To define a specification, open the specification dialog shown below by selecting **CSV Import/Export Specifications** from the **Configuration** menu.



The *CSV Import/Export File Specification* dialog provides the following functionality:

Element	Description
Specification Name	Unique name to apply to this specification. Will be used to select a specification from the drop list in the import/export dialog.
Delimiter	Character delimiter to use between record fields. Note that if a field contains an instance of the delimiter, the field will be exported wrapped in " (quotation marks) ... and all instances of " in the field will be doubled (ie. " becomes "").
Notes	Description of the specification - only used in this dialog for descriptive purposes.
Default Filename	Default filename.
Default Direction	Set for Import or for Export. A specification may be used in either direction, but this allows you to set the default type.
Default Types	Limit the element types being exported by entering a comma separated list here: eg. class,requirement,component,node,object.
Available fields	List of possible record fields, not yet allocated.
File Specification	List of record fields (in order) already assigned.
Add Field	Move all selected fields in top list to bottom list.
Remove Field	Move all selected fields in bottom list back to available list.
New	Create a new specification.
Save	Save changes to the currently selected specification.
Save As	Save the current specification with a new name.
Delete	Delete the current specification.
Cancel	Close this dialog.

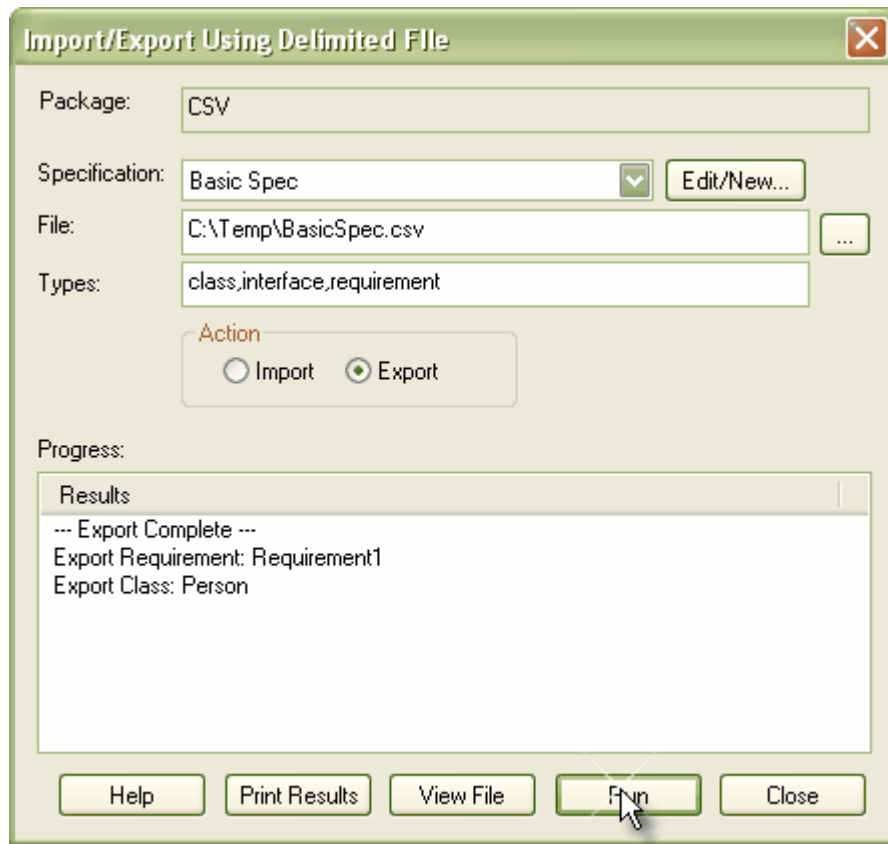
7.9.2 CSV Export

It is possible to export information about EA elements in CSV format. Once you have defined a [CSV export specification](#) it is possible to write out major element attributes to a CSV text file.

Export Data in CSV Format

To export data in CSV format, follow these steps:

1. In the Project Browser, right click on the package containing the elements to export.
2. Select the *Import/Export* sub-menu.
3. Select the *CSV Import/Export* menu item.
4. The following dialog will appear.



5. Set the required options - the *Import/Export Using Delimited File* dialog provides the following functionality:

Element	Description
Package	Name of the current selected package.
Specification	Name of export specification to use.
Edit/New	Press to edit the export specification or create a new one.
File	The filename to export to.
Type	List of types to export - leave blank for all, or enter a comma separated list of types.
Action	Check Export to export to file, Import to import from file.
Include Subfolders	Include all elements in all child packages when exporting.
Export column names	If ticked, EA will write out the column names as the first row.
Print Results	Print out the result list.
View File	View CSV file with default windows application for CSV files.
Run	Perform the export.
Close	Exit this dialog.

7.9.3 CSV Import

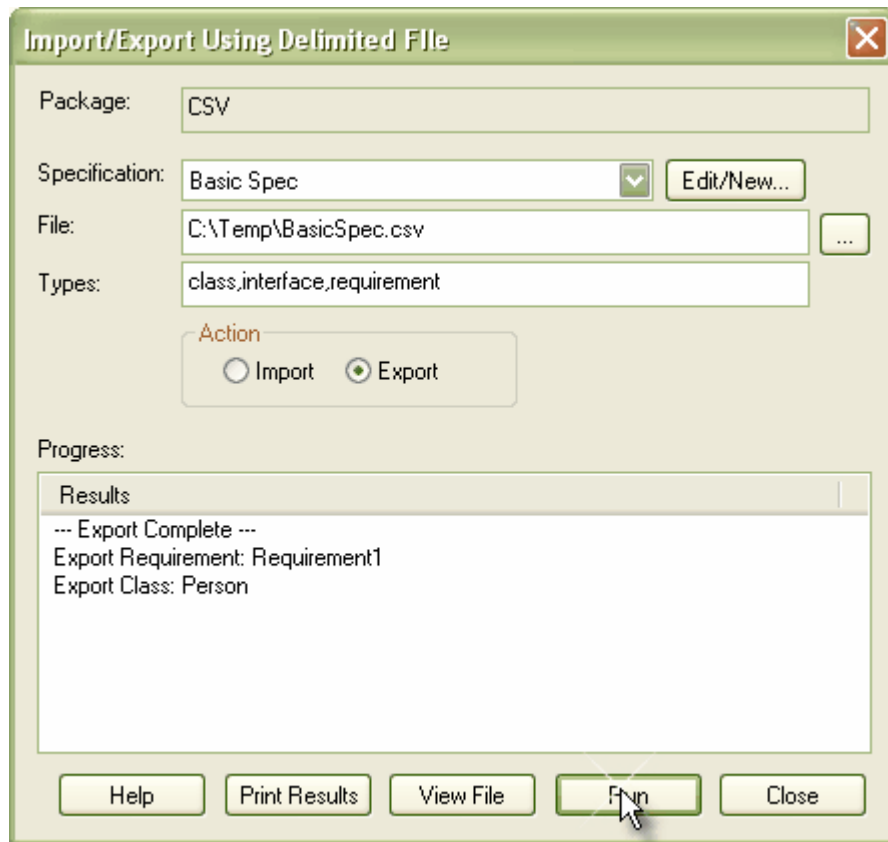
It is possible to import information about EA elements in CSV format. Once you have defined a CSV [import specification](#) it is possible to read in major element attributes from a CSV text file.

When importing, EA will check the specification to see if there is a GUID field included. If there is, EA will attempt to locate the element identified by the GUID, and if successful will perform an update on the current element, rather than creating a new one. If no GUID field is defined, or EA cannot locate the identified element, a new element is created and placed in the current package. Note that during import, "Type" is a mandatory field and must match one of the legal EA element types.

Import Data in CSV Format

To import data in CSV format, follow these steps:

1. In the Project Browser, right click on the package you wish to import to
2. Select the *Import/Export* sub-menu.
3. Select the *CSV Import/Export* menu item.
4. The following dialog will appear.



5. Set the required options - the *Import/Export Using Delimited File* dialog provides the following functionality:

Element	Description
Package	Name of the current selected package.
Specification	Name of import specification to use.
Edit/New	Press to edit the import specification or create a new one.
File	The filename to import from.
Type	Not used for import.
Action	Check Import to import from file.
Include Subfolders	Not used for import. During import, all new records are placed in the current folder.
Export column names	If ticked, EA will treat the first row as column names.
Print Results	Print out the result list.
View File	View CSV file with default windows application for CSV files.
Run	Perform the import.
Close	Exit this dialog.

7.10 Version Control Options

Version Control can be achieved by using either SCC configurations or CVS configurations. When using SCC a third party interface must be used. When using CVS, users may choose to make use of the inbuilt CSV configuration option.

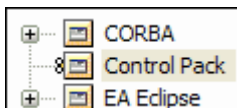
See the following Topics:

- [Version Control Overview](#)
- [Version Control Setup](#)
- [Using Version Control](#)
- [Version Control Reference](#)

7.10.1 Version Control

Enterprise Architect supports version control of packages and their component sub-packages to a central repository. This repository is maintained by third-party version control applications that control access and record revisions. Version controlled packages are controlled packages that have been configured for use for version control and may be configured with CVS or with SCC.

Controlled packages configured for version control appear in the project browser with a small figure 8 to the left of the package icon as in the example below:



7.10.1.1 Version Control Overview

Features

The Version Control feature of Enterprise Architect provides two key facilities:

- Saving a history of changes to EA packages, including the ability to retrieve previous versions.
- Coordinating the sharing of packages between users.

System Requirements

To use version control in EA the following is required:

- A third-party source-code control application which provides support the Microsoft Common Source Code Control standard, version 1.1 or higher.
- Alternatively a concurrent version control system may be used. CVS may be downloaded from www.cvshome.org.

Set-Up

Before using the version control facility, a suitably compliant version control software and repository must be installed on every PC which intends to share packages. Once the version control software has been installed and has been configured for SCC or CVS on individual machines involved in version control, users may share packages via a nominated project inside the SCC or CVS repository. For more information, see [Version Control Set-up](#).

Note: *If you are using the Corporate version of EA with security enabled, you will also need to set up permissions to configure and use version control. See [List of Available Permissions](#) for further information.*

Usage

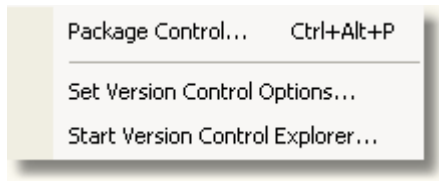
There are four basic ways in which the version control facility might be employed:

Use	Description
Shared model	Users sharing a central EAP file or SQL database. This configuration allows users to see other users' packages without explicitly having to retrieve them. Version control regulates access to packages, and maintains package revision history.
Duplicate models	An EAP file is created by a single user who configures it for version control. The file is then distributed to other users who connect using their own user name. Other users' packages are retrieved using the Get Package command.
Shared packages	Individual users create their own EAP files but share one or more packages by connecting to the same version control repository and using the Get Package command.
Standard packages	A company may have a standard set of packages which are broadly shared (on a read-only basis). Individual users use the Get Package command to connect to the shared package.

7.10.1.2 Version Control Setup

Version control may be assigned to individual packages in EA, each package can only be linked to one version control configuration at a time, although it is possible to connect multiple control configurations for each model. The Version Control Configurations dialog may be used to connect to an SCC provider or to a CVS configuration.

The Version Control Setup Menu is available under *Project | Version Control* in the main menu. To set the version control configurations choose the *Set Version Control Options...* item from the menu.



See the following Topics:

- [Version Control Options SCC](#)
- [Version Control Options CVS](#)
- [Configure Package for Version Control](#)

7.10.1.3 Using Version Control

Outlined below are the steps required to perform the activities most common to using the version control features of Enterprise Architect.

See the following Topics:

- [Version Control Options SCC](#)
- [Version Control Options CVS](#)
- [Add a Previously defined Version Control Configuration to a package](#)
- [Checking In and Checking out Packages](#)
- [Review Package History](#)
- [Remove Package from Version Control](#)
- [Upgrade Existing SCC configurations to EA version 4.5](#)

General Notes

- The export/import facility is somewhat slow and submitting packages containing many sub-nodes to version control should be avoided.
- Replication should not be combined with version controlled packages.

7.10.1.4 Version Control Reference

The following references are available for Version Control.

Menus

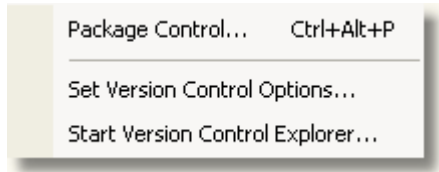
- [Version Control Setup Menu](#)
- [Version Control Menu](#)

Dialogs

- [Version Control Providers Dialog](#)
- [Get Shared File from Repository Dialog](#)
- [Version Control Options Dialog](#)

7.10.1.4.1 Version Control Setup Menu

The Version Control Setup Menu is available under *Project* | *Version Control* in the main menu.



Menu Item	Functionality
Package Control	Opens the package control dialog which allows you to specify whether this package (and its children) is controlled, and which file it is controlled through.
Set Version Control Options	Displays the Version Control Options dialog .
Start Version Control Explorer	Asks the SCC provider to display an interface allowing you to review all SCC projects and their contents directly.

7.10.1.4.2 Version Control Options Dialog

The Version Control Options dialog allows you to specify the options required to connect to a Source Code Control (SCC) provider or to use CVS through EA's internal CVS configuration. It is possible to have multiple version control configurations in the same model.

Setting Up Version Control

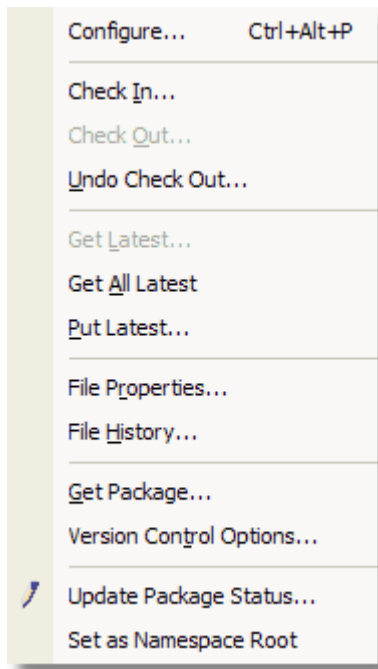
Version control may be assigned to individual packages in EA, each package can only be linked to one version control configuration at a time, although it is possible to connect multiple control configurations for each model. The Version Control Configurations dialog may be used to connect to an SCC provider or to a CVS configuration.

See the following Topics:

- [Version Control Options SCC](#)
- [Version Control Options CVS](#)
- [Configure Package for Version Control](#)

7.10.1.4.3 Version Control Menu

The *Version Control* menu is found by right clicking on a version controlled package in the Project Browser and selecting *Package Control*.

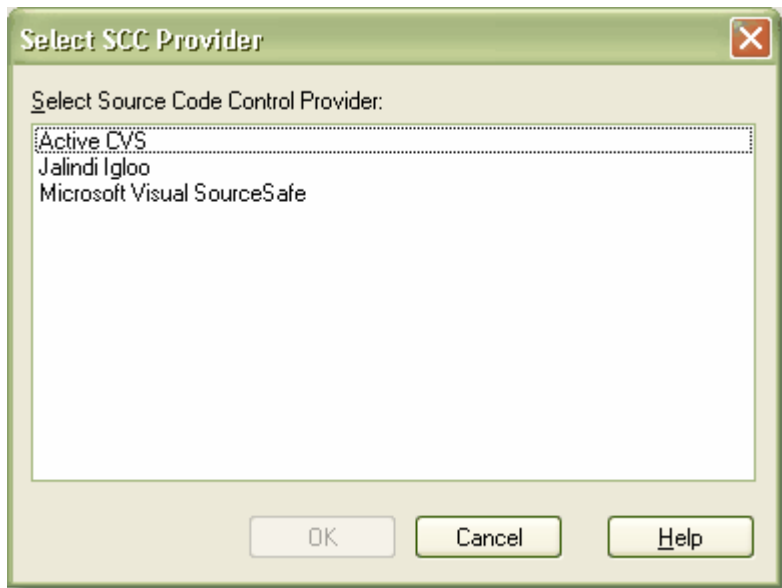


Menu Item	Functionality
Configure	Shows the package control dialog which allows you to specify whether this package (and its children) is controlled, and which file it is controlled through.
Check In	Submits the currently selected package and all sub-packages to the central repository. You will be prompted to enter optional comments describing changes to the packages.
Check Out	Retrieves the latest version of the currently selected package and sub-packages from the central repository, overwriting the current packages. After check out the packages are available for editing.
Undo Check Out	Cancel all changes you have made to the currently selected package and sub-packages. Restores the model to the state it was in before package was checked out, leaving the select package and sub-packages locked.
Get Latest	Retrieves the latest version of the package from the repository. Available only for packages which are checked in. Get Latest is not intended for shared .EAP files and should only be used when people have their own individual EA models.
Put Latest	Updates the central repository with the currently selected package (which you have checked out), while retaining checkout status on the package. This is equivalent to checking a package in and immediately checking it back out again.
File Properties	Asks the version control provider to show the version control properties associated with the XML export file pertaining to the currently selected package.
File History	Where the controlling package has been configured by an SCC provider, this provider will show a change history for the package. Refer to your provider's documentation for details on how to use the control. Otherwise if the version control is CVS the history will be shown via EA's internal CVS history menu.
Get Package	Allows the user to gain access from packages in the version control repository that is not currently available in the users model.
Version Control Options	Displays the Version Control Options dialog .
Update Package Status	Allows the user to provide a bulk update on the status of a package, this includes status options such as Proposed, Validate, Mandatory etc. Note: <i>this option is a generic package element not specific to version control.</i>
Set as Namespace Root	Sets the namespace root for languages which support namespaces, for more information see the Namespaces section. Note: <i>this option is a generic package element not specific to version control.</i>

7.10.1.4.4 SCC Providers Dialog

When using a SCC version control, the **Select SCC Provider** dialog allows the selection from a list of the installed SCC configuration providers that may be used for this package.

Note: *All users of the shared database are required to specify the same SCC provider.*



7.10.1.4.5 Version Control Options SCC

To set up SCC version control configuration use the following steps:

Set Up the Source Code Control Provider with SCC

To setup the third-party source code control provider, refer to the documentation provided with that application. A repository must be set up using the SCC provider and access to that repository must be available to all intended users.

Connect an EA Model to Version Control with SCC

1. Open/Create the EA model you wish to place under version control.
2. From the *Project | Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button, then ensure that the *Select Type* radio button is set to *SCC*.

Version Control Configurations

Select Type: SCC CVS

Unique ID: MySSCConfiguration

Local Project path: C:\SCC

Current User: gsparks Change User...

SCC Provider: PushOk CVS Proxy

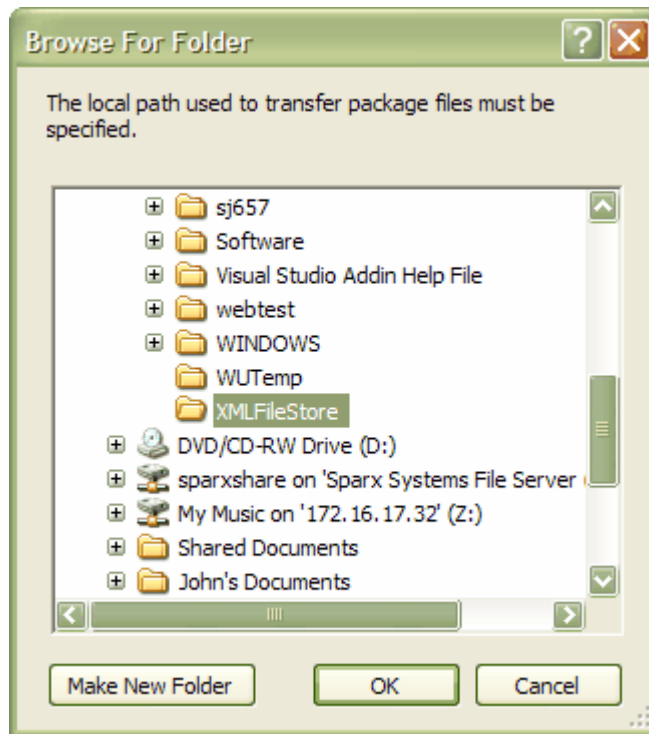
SCC Project: "\$/SharedProject", LBAAAAAA

Defined Configurations: New Save Delete

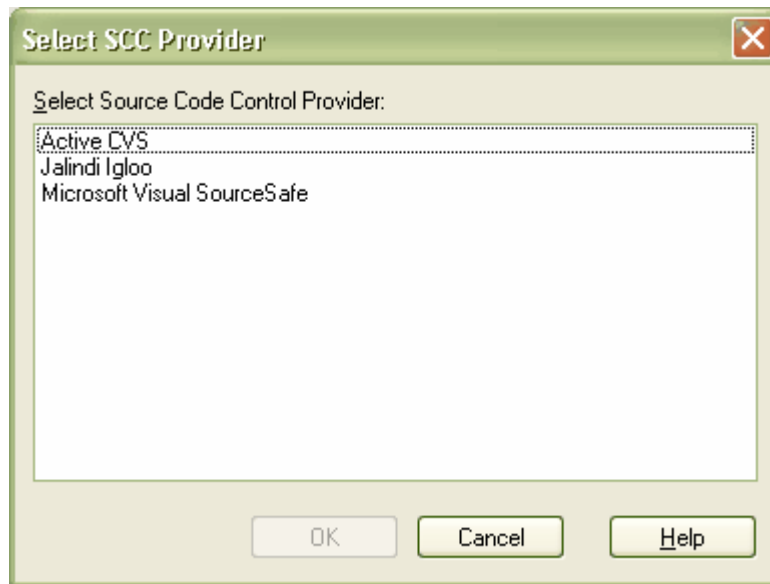
Unique ID	Type	Location
MySSCConfiguration	SCC	"\$/SharedProject", LBAAAAAA

Close Help

4. In the Unique ID field give the configuration an appropriate unique name.
5. The *Local SCC Project path* field is specified by clicking on the browse button (...). Select the local folder which will be used to keep XML files to be stored in the Version Control repository.



6. Select a SCC provider from the dialog which opens, and press the **OK** button.



7. The SCC provider is likely to prompt you for various details including the name of the project you want to connect to, and perhaps the user name you want to use when you log in.
8. Press the **Close button** to confirm the configuration.

Menu Item	Functionality
Unique ID	Specify a configuration name that will readily distinguish it from other configurations. The Unique ID will appear as a selection in list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in the current model.
Local SCC Project Path	The folder where the XML files representing the packages are stored. This folder should already exist before it is specified here. Every PC using version control should have its own local SCC project folder - this should not be a shared network folder. Particularly bear this in mind if you are creating an EAP file which is to be shared (eg. A SQL database).
Change User	Allows the user of an EAP model to change the user name by which he is logged into the source code control provider. It is not necessary to use this option when first specifying the version control parameters as the SCC provider should do this automatically.
Current User	Read only. Shows the user name of the user currently logged into the SCC provider.
SCC Provider	Read only. This is the name of the provider specified in the database.
SCC Project	Read only. This is the project selected during the initial setup of the connection to the SCC provider.

7.10.1.4.5.1 SCC Version Control User Options

Change User

It may be necessary to change the SCC user name. While the following procedure allows this to happen, it is not generally recommended as it may result in a confused checkout states for packages.

1. From the *Project | Version Control* submenu, select *Set Version Control Options*.
2. Select *Change User*.

Users of SCC

Some SCC providers require you to specify a user name, in which the following points apply:

- The version control user name has no relationship to the users set up as part of EA security.
- A single user can be logged into multiple machines simultaneously.
- It is not a "roaming" facility - logging into a PC does not grant access rights to packages checked out on other machines. Each machine has its own record of which packages are checked out and will only allow access to those.
- User name is stored per PC and is stored across EA sessions. If you want to change to a different user press *Change User* in the version control screen.

7.10.1.4.5.2 SCC Version Control Show Interface

Show the SCC Interface

1. From the *Project | Version Control* submenu, select *Start Version Control Explorer*.
2. The third party source code controller should appear listing all the projects contained in its database.

7.10.1.4.6 CVS with Remote Repositories

To set up CVS version control with a remote repository use the following steps:

1. Connect to the remote CVS repository. An example connection command is shown below.

```
C:\>cvs -d :pserver:user@ServerName:/cvs login
```

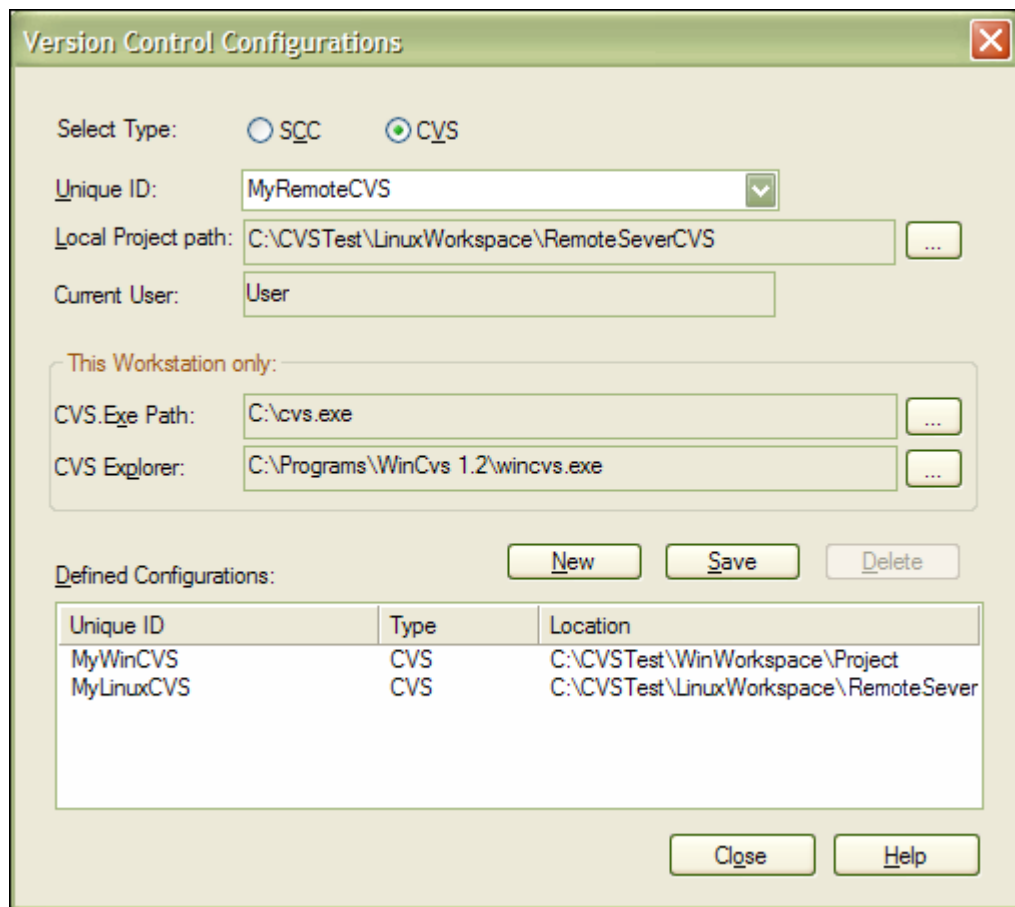
Note: replace *user* with your username and *ServerName* with the name of your server.

2. Create a local CVS workspace derived from the remote repository. An example command is shown below.

```
c:\myworkspace>cvs -d :pserver:user@ServerName:/cvs checkout MyProject
```

Note: refers to the remote installation of CVS for the <My Project> name was assigned. This name is case sensitive.

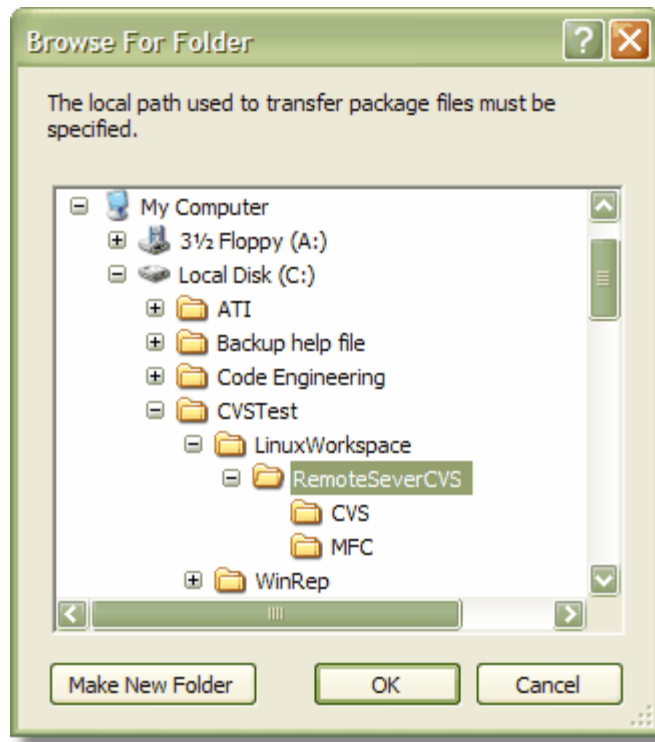
3. Open/create the EA package you wish to place under version control.
4. From the *Project | Version Control* submenu, select *Set Version Control Options*.
5. Press the *New* button, then ensure that the *Select Type* radio button is set to *CVS*.



6. In the *Local Project path* field, enter a file path, or Browse [...] to locate the local CVS working folder. The name in the *Current User* field should reflect the user name used to log in to the remote CVS repository, if this does not happen it indicates that there is a problem with the configuration. The Local

Project path by default will be the local temporary file location. This local path reflects the <My Project> in the remote installation of CVS on the server.

7. In the Unique ID field give the configuration an appropriate name.
8. In the *CVS.Exe Path*, enter the file path for the cvs.exe or browse [...] to locate the executable.



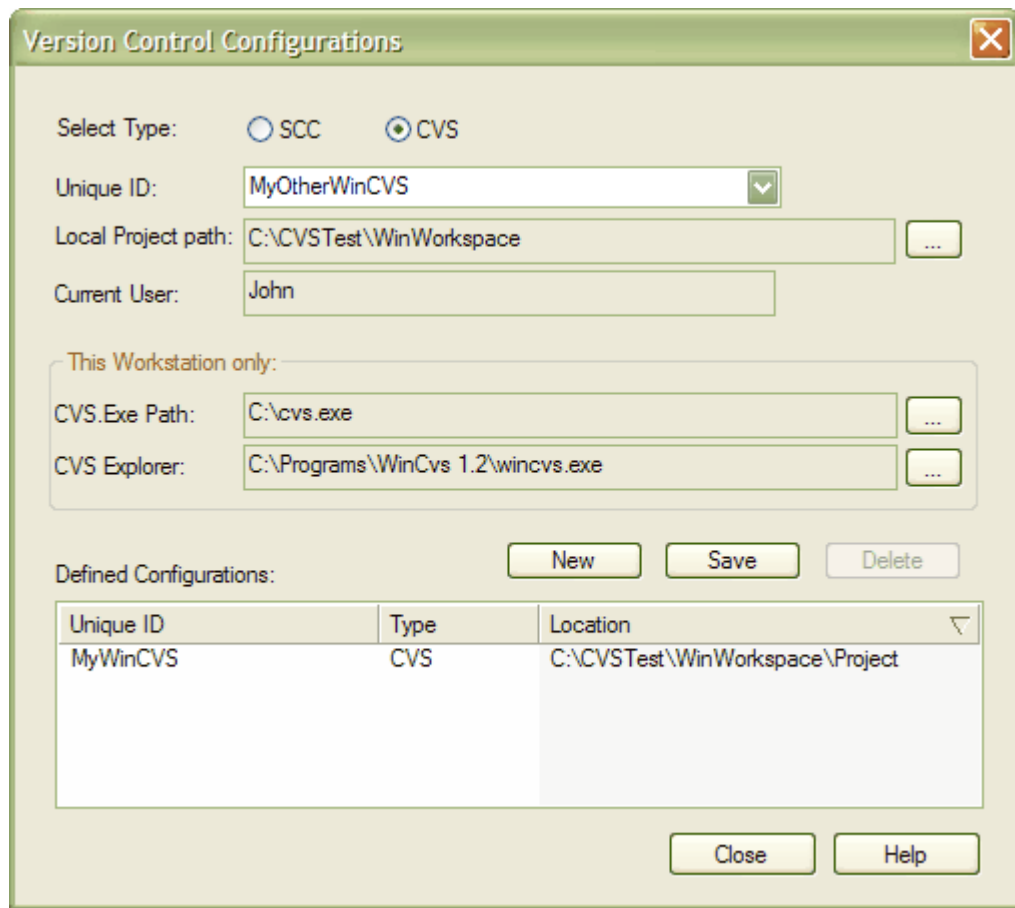
9. Optionally the user may specify a path to a CVS Explorer by adding the CVS Explorer path in the CVS Explorer field or by pressing the Browse [...] button to locate the file.
10. Press the *Save* button to confirm the configuration.

Menu Item	Functionality
Unique ID	Specify a configuration name that is will readily distinguish it from other configurations. The Unique ID will appear as a selection in list of Version Control configurations a package can connect to. In addition it is possible to select a previous version control configuration from this drop down menu providing the configuration is not in the current model.
Local Project Path	The Local Project path by default will be the local temporary file location. This local path reflects the <My Project> in the remote installation of CVS on the server.
Current User	User of the Repository
CVS.EXE Path	Full Path to the local cvs application e.g cvs.exe
CVS Explorer	Full Path to a third party CVS explorer e.g. wincvs.exe

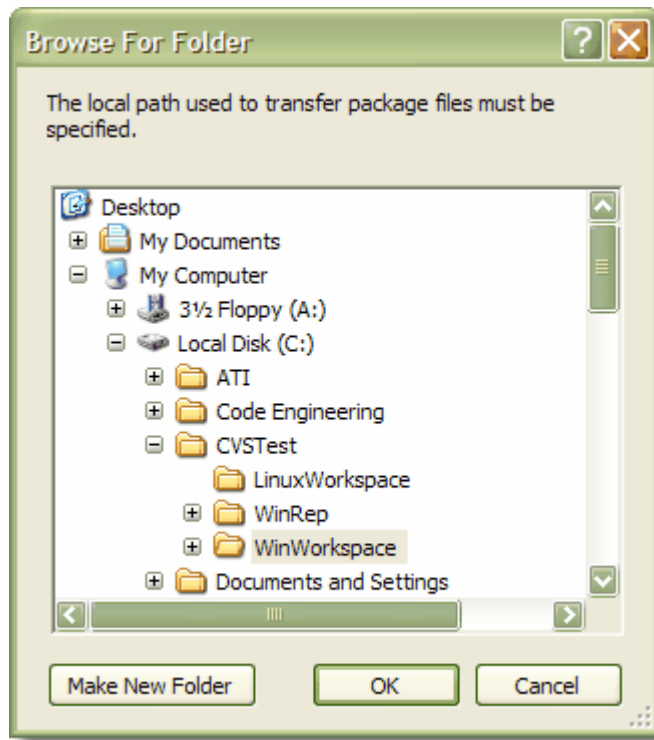
7.10.1.4.7 CVS with Local Repositories

To set up CVS version control use the following steps:

1. Open/create the EA model you wish to place under version control.
2. From the *Project | Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button, then ensure that the *Select Type* radio button is set to *CVS*.



4. The *Local Project path* field is specified by clicking on the browse button (...). Select the CVS local working folder.

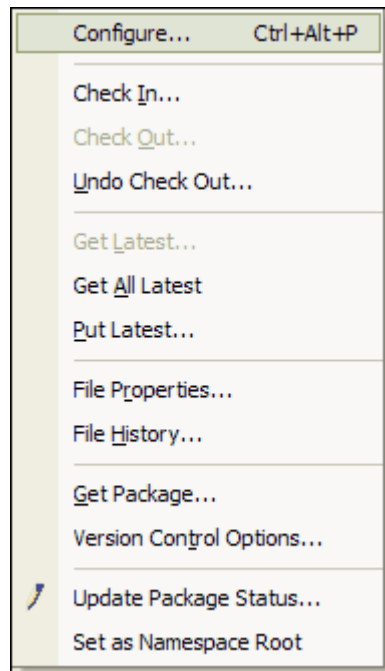


5. In the Unique ID field give the configuration an appropriate unique name.
6. In the *CVS.Exe Path*, enter the file path for the cvs.exe or Browse [...] to locate the executable.
7. Optionally the user may specify a path to a CVS Explorer by adding the CVS Explorer path in the CVS Explorer field or by pressing the Browse [...] button to locate the file.
8. Press the *Save* button to confirm the configuration.

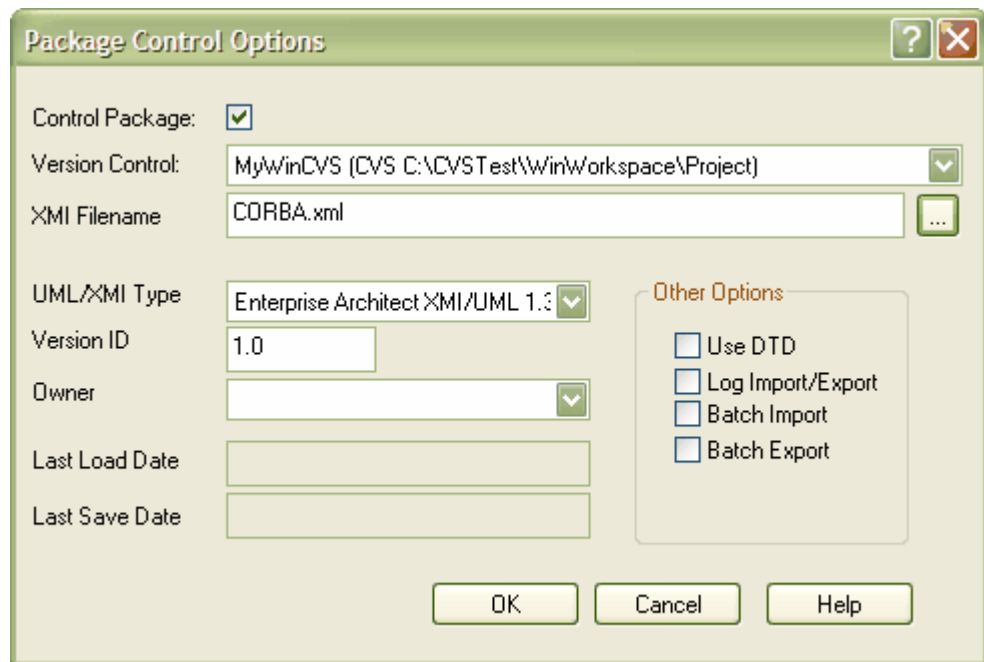
7.10.1.4.8 Configure Package for Version Control

To configure a version controlled package, follow the steps detailed below:

1. In the Project Browser, right click on the package you wish to control or configure.
2. From the *Package Control* submenu, select *Configure*.



3. The *Package Control Options* dialog will be displayed.



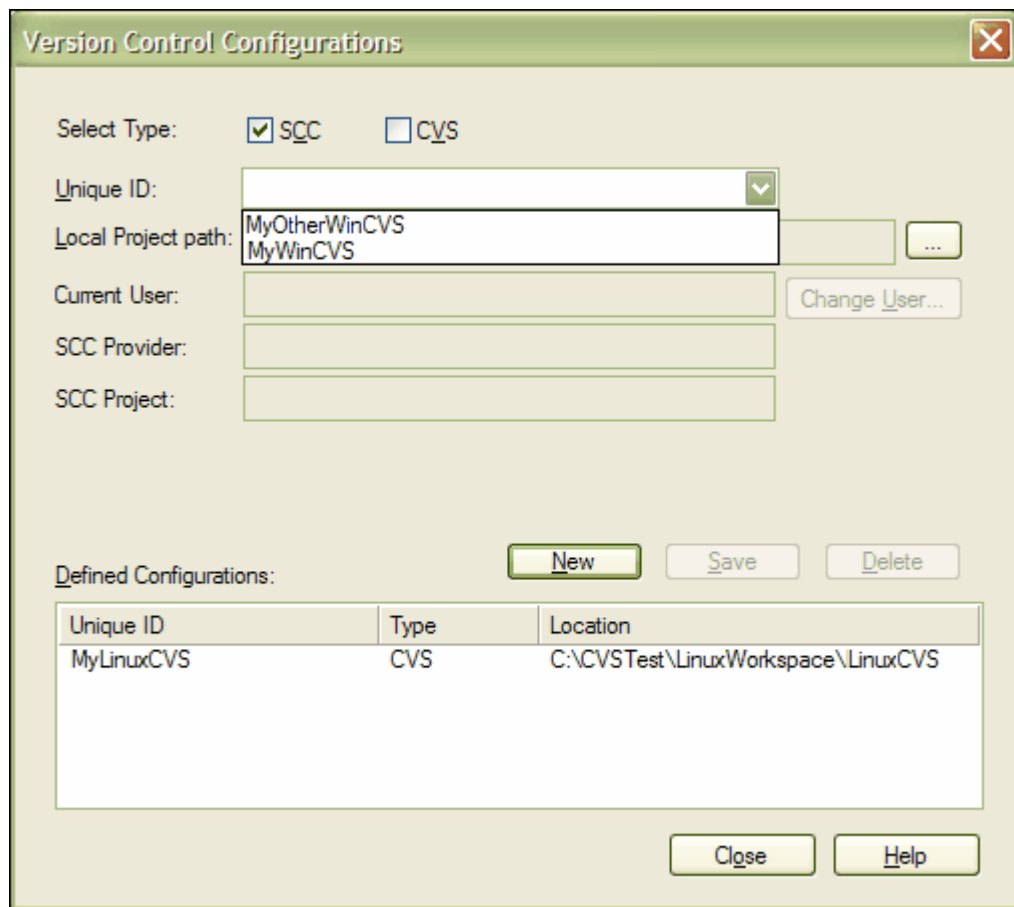
4. Ensure that the *Control Package* checkbox is selected.
5. Select a version control configuration from the *Version Control* drop down menu.
6. Specify an XMI filename or press the ... button to browse to an xml file (EA will automatically create one with the package name when the ... button is selected).
7. Fill in any of the remaining options on the dialog - you can configure the following:

- Set the *UML/XMI Type* to determine the type of XMI generated, this includes Enterprise Architect XMI/UML 1.3, Rational Rose/Unisys UML 1.3 and Generic XMI 1.0/UML 1.3. Currently only Enterprise Architect UML1.3 is supported for complete import/export round tripping of packages
- Set the *Version ID* (manual process)
- Set the package *Owner*
- Set whether to *Use a DTD*; and
- Set whether to *Log Import/Export* activity to a log file
- Whether the package is marked as a *Batch Import* package
- Whether the package is marked as a *Batch Export* package

7.10.1.4.9 Add Previously defined Version Control Configurations

Once a version control configuration has been defined in one model it is possible to add the configuration to other models. To use this feature follow the instructions detailed below:

1. Open the model with the package that is to be configured with the predefined version control configuration.
2. Select the package in the *Project View* and right click on the package to bring up its context menu. Mouse over the *Package Control* item and select *Version Control Options* from the *Package Control* sub menu .
3. Press the *New* button.
4. From the *Unique ID* dropdown menu select one of the previously defined version control configurations.



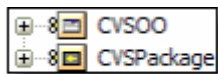
5. Press the **Save** button to confirm the version control configuration.

7.10.1.4.10 Checking In and Checking out Packages.

To work on a version controlled package the user must have the package checked out, when a package is checked out to a specific user other users may not make changes to the package until it has been checked in.

Check In/Check Out

1. In the Project Browser, right click on the package icon.
2. From the **Package Control** submenu, select **Check In**, **Check Out** or **Undo Checkout** as appropriate.
3. Optionally enter a comment if prompted to do so.
4. The Project Browser icon should change the user, when a package is checked in the package will be represented in the **Project View** as a package with a figure 8 to the left of the package and the package icon displayed normally, if the package has been checked in the icon changes to a small green horizontal cross. The example below shows the upper package as being a checked out package, whilst the lower package is checked in.

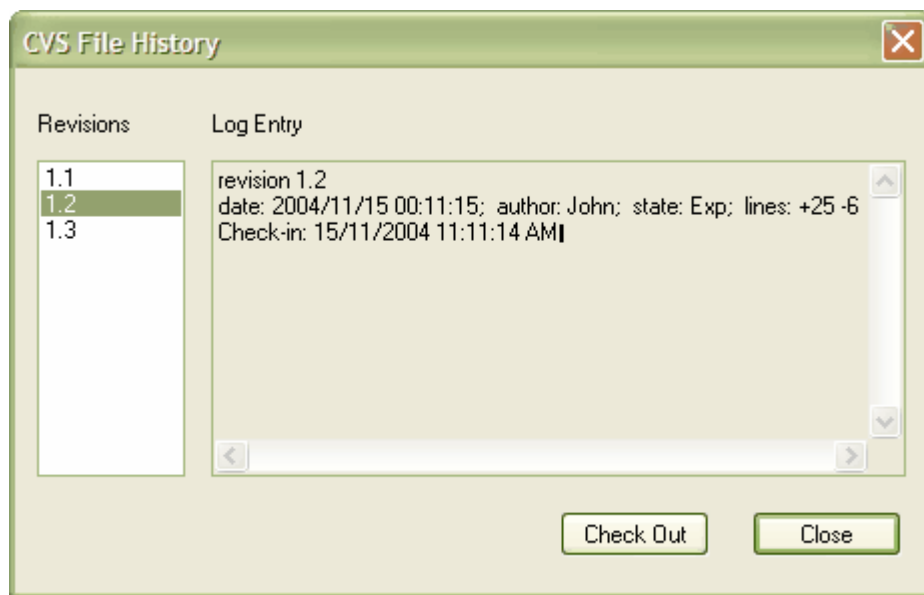


7.10.1.4.11 Review Package History

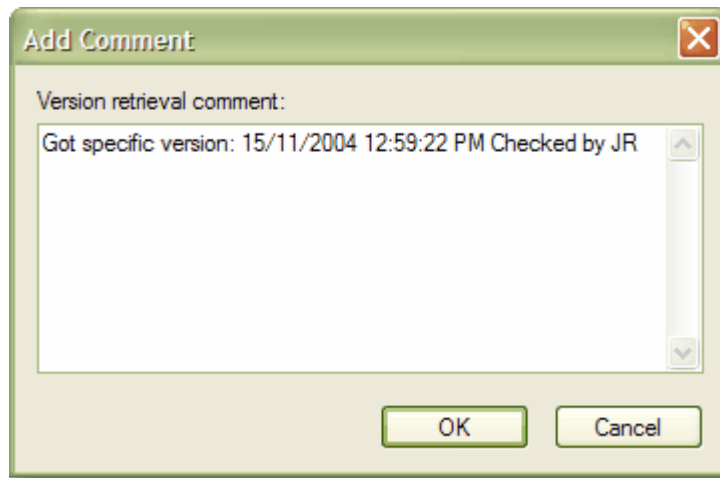
Reviewing package history allows users to view the history of checked in package revisions. This will open up the package in read-only mode allowing users to view the package contents, but not make any changes to the package. To review the package history use the following steps:

Review Package History

1. In the *Project Browser*, right click on the package configured for version control.
2. From the package context menu mouse over the Package Control submenu and select the File History option.
3. From the *Package Control* submenu, select *File History*. If the package has been configured for CVS the following instructions apply. If the package has been configured through SCC, the history will be accessed through the mechanism offered by the third party SCC provider.
4. This will open the *CVS File History* dialog, from here the user may view the log entries by clicking on the revision numbers in the *Revisions* field. To view the package history select a revision and then press the *Check Out* button.



5. This will bring up a dialog warning that the model will be opened in read-only mode. Press the *Yes* button to continue or *No* to cancel the action.
6. The *Add Comments* dialog allows the user to add comments regarding the retrieval of the package history. Enter the comments in the text field then press the OK button to proceed.



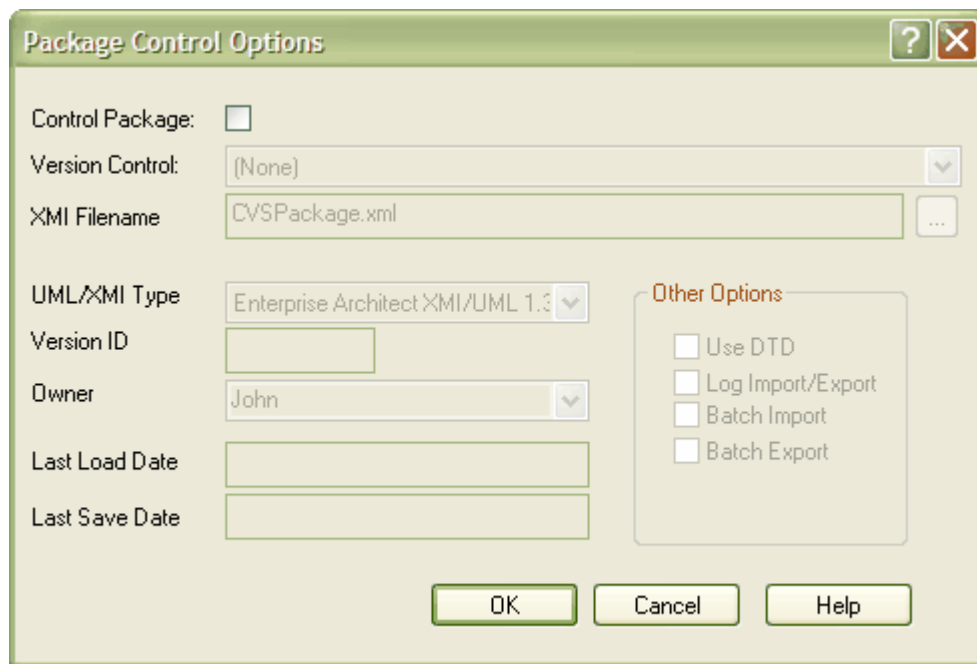
7. The package will then be retrieved in read only mode to allow the user to view the history of the package at the specified version. To go back to the latest version check in the package by using the check in command from the Project View by right clicking on the package and from the context menu mouse over the *Package Control* item and from the submenu select the *Check In..* option.

7.10.1.4.12 Remove Package from Version Control

Packages may have their connections to version control removed by using the following steps:

Remove a Package from Version Control

1. In the *Project Browser*, right click on the package configured for version control.
2. From the *Package Control* submenu, select *Configure* or press the *Ctrl + Alt + P* hotkey combination.
3. Uncheck the *Control Package* checkbox.



4. Press **OK** to remove from version control.
5. The package will now be removed from version control.

Note: before removing a package from version control the package must be checked in.

7.10.1.4.13 Including Other Users Packages

Include Other Users' Packages in Your Model

Other users may be creating packages that you may wish to import into your model. If you are not sharing a SQL database or EAP file you will not automatically detect the presence of such packages.

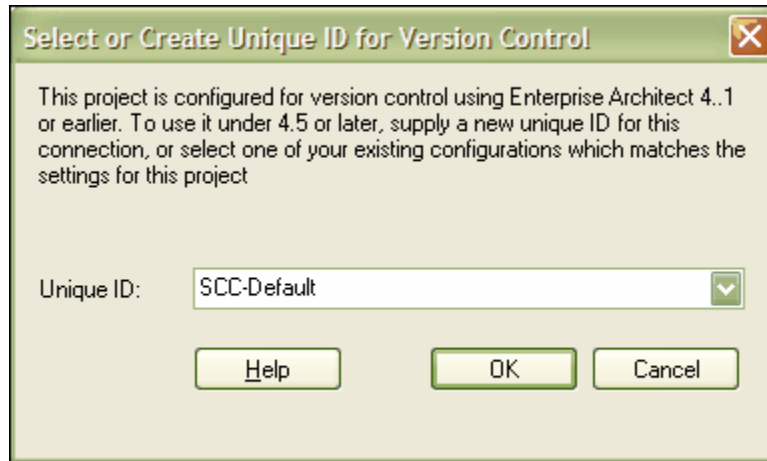
1. In the Project Browser, right-click on a package which you wish to make a parent of the incoming package.
2. From the **Package Control** submenu, select **Get Package**.
3. Double click on the name of a file. You may have received information from other users as to the name of the export file they have attached to their package, or you may simply want to retrieve any newly available packages from the central database.

7.10.1.4.14 SCC Version Control Upgrade for 4.5

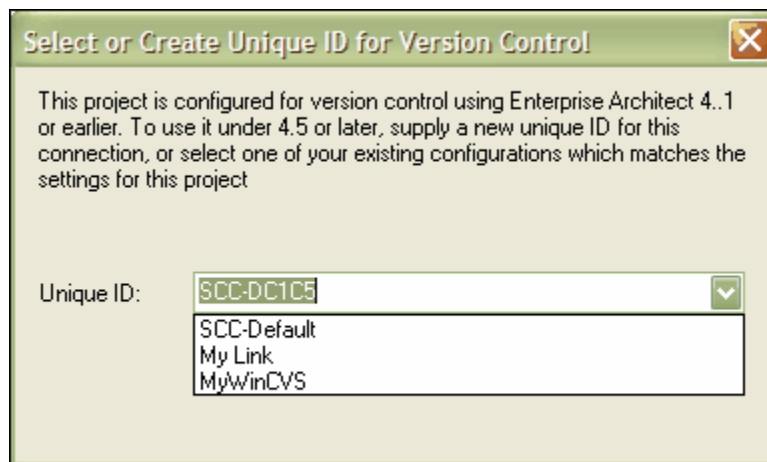
When a version controlled project created under a version of EA earlier than 4.5 is opened in version 4.5 (or later) the SCC connection must be identified with a new unique ID. The user may assign a name to the existing SCC configuration or associate the project with a configuration that has previously been assigned a unique ID. By having a unique ID for version control configurations, the user may assign a version control configuration quickly and efficiently. This is achieved by using configurations that have been created previously for other version controlled repositories. This allows the many packages to be configured to use an existing version control repository, this may apply to more than packages created for more than just one model allowing for a great deal of flexibility. To upgrade an existing SCC version control project created before version 4.5, in

version 4.5(or later) of EA use the following steps.

1. Open a project with a SCC version control configuration created in EA earlier than version 4.5 (or later).
2. The *Select or Create Unique ID for Version Control* dialog will prompt the user to create an ID for an existing configuration or choose a previously created one from the *Unique ID* drop down list.
3. The existing SCC configuration will be the initial value which will be represented by the value SCC-XXXXX, this number is not especially meaningful, therefore it is recommended that the configuration be given a meaningful name.



4. The user may associate the version controlled package with previously defined configuration by selecting an existing configuration from the *Unique ID* drop down list (if one exists).



5. After the selections have been made press the *OK* button to proceed loading the model.

7.11 Model Sharing and Team Deployment

Introducing Team Development

EA offers a diverse set of functionality designed specifically for team development supporting team development using project sharing and distributed development. Project sharing can be achieved through several mechanisms including network deployment, replication, XMI Import/Export, Version Control, Package Control and User Security.

Network deployment may be undertaken using two different schemas for deployment, either using:

- .EAP based repositories or
- DBMS server based repositories.

Replication requires the use .EAP based repositories, and cannot be performed on repositories stored on a DBMS server. DBMS server based repositories offer better response times than .EAP files on networks due to the inherent structure of the DBMS. DBMS also offers a better solution when networking problems are encountered, as they have the ability to backtrack transactions caused by external breakdowns.

Replication

Replication is a simple process which allows for data interchange between .EAP based repositories (not DBMS) and is suitable for use in situations where many different users work independently. Modelers merge their changes into a Design Master on an "as required basis". With replication it is recommended that a backup is carried out prior to replication.

XMI Import Export

The XMI Import/Export can be used to model discrete packages that may be exported and be shared between developers. XMI allows the export of packages into XML files which may then be imported into any model.

Package control can be used to set up packages for version control and allow for the batch export of packages using XMI. Version Control allows for a repository to be maintained by a third -party source code control application which is used to control access and record revisions.

Security

User security is used to limit the update access to model elements. It provides control over who in a project can make changes to model elements.

For more information regarding the use of EA with shared models and team deployment please see the Deployment of Enterprise Architect white paper available from http://www.sparxsystems.com.au/WhitePapers/EA_Deployment.pdf.

Note: *DBMS Repository support and User Security are only available with the Corporate Edition of Enterprise Architect.*

See:

- [Distributed Development](#)
- [Project Sharing](#)
- [XMI Import/Export](#)
- [Replication](#)
- [Package Control](#)
- [Version Control](#)
- [User Security](#)

7.11.1 Sharing an Enterprise Architect Project

Note: *Project Sharing and Replication are only enabled in the Professional and Corporate versions of Enterprise Architect*

Sharing a project among a team of designers, developers and analysts is the most efficient way of using EA to manage a team development. Many people can work on the model at the same time and contribute their particular skill. Team members can always see what the latest changes are, keeping the team informed and up to date with the project status.

Sharing an EA Project

You may share an Enterprise Architect project in three ways:

1. Using a [shared network directory](#). In this scenario you place the project file on a shared network drive. Individual developers and analysts can then open and work on the project concurrently. Note that some project views (especially the Project Browser) will require occasional refreshing to see changes made by other users.
2. Using replication. [Replication](#) is a powerful means of sharing projects between isolated or mobile users. In the replication scenario a project is converted to a design master, then replicas made of the master. Users take the replicas away, modify the project, then bring their replicas back to be synchronized with the master file.
3. Using a shared DBMS based repository (Corporate Edition only).

7.11.2 Sharing a Project

The easiest way to share a project amongst a work group of developers and analysts is to place the project file on a shared network drive and have people connect concurrently from their Workstation.

Note: Enterprise Architect will accept a number of concurrent connections without issue, although there may be occasional 'lock-outs' when one user wishes to access or update something another user is in the process of modifying.

Network Issues

The main issues with shared network access are:

- Changes to the Project Browser are not automatically updated. To compensate for this, users must occasionally reload their project to view any project changes at this level.
- If two or more people work on the same diagram concurrently, unexpected results may occur. It is best to allow only one analyst to work on a diagram at a time.
- If a User's machine crashes, the network suffers an outage or a machine is turned off unexpectedly, the project file may require repair, to compensate for the sudden inconsistency. A [repair](#) facility is provided (select *Repair .EAP File* from the *Tools | Manage .EAP File* submenu) to carry out this task. This only applies to the file based version of EA - the DBMS based version does not suffer this problem.

7.11.3 Distributed Development

Enterprise Architect supports distributed development using two different techniques as described below.

Replication

Use the Replication features to allow geographically separated analysts to update and modify parts of the model in replicas, then merge these back together at a central location.

For further information see the [Replication](#) section.

XMI Import/Export

Use the XMI based Import/Export facility to model discrete packages, export to XML and share among the development team. This approach has several benefits over replication:

1. You can assemble a model from only the parts you need to get your job done.
2. You can assemble a full model if required.
3. You can assemble a model from different package versions for different purposes (eg. customer visible, internal release only etc.).
4. You can roll-back parts of a model on an as required basis.
5. There is less chance of 'collisions' between developers if each works on a discrete package.

- The process is controllable using a [version control](#) system.

Use the [Import and Export menu options](#) (below) to access this feature - they are available through the [Project | Import/Export](#) submenu. Also see the [XMI section](#) for further information about XMI based import and export.



The [Controlled Package](#) feature can also be used to assist in the process.

Note: XMI based import/export is UML 1.3 / XMI 1.1 compliant. Users can also write XML based tools to manipulate and extract information from XML files to enhance the development process.

7.11.4 User Security

What is User Security in EA?

User security in EA can be used to limit the access to update functions within the model. Elements may be locked on a per-user or per-group basis and where user security is enabled a password is required to log in to the model. Security in EA is not designed to prevent unauthorized access: rather it is intended to provide a means of improving the collaborative design and development by preventing concurrent editing and limiting the possibility of inadvertent model changes by users not designated as model authors.

User Security basics

User security can be switched on in the Corporate Edition of Enterprise Architect. The user security in EA offers two security policies, the standard security model and the rigorous security model. In the standard security model all elements are considered unlocked, and as the need arises, a user may lock any element or set of elements at the user or group levels depending on permission rights that the user has to the model. The rigorous security model assumes that everything in the model is locked until explicitly checked out with a user lock. In this mode, an EA model is read-only until a user applies an editing lock on one or more elements. For more detailed information regarding the security policies view the [security policy](#) topic.

User Security Tasks

There are various security tasks which may only be performed by users with Administrative rights to the model, these tasks include:

- [Enabling Security](#)
- [Maintaining Users](#)
- [Set up User Groups](#)
- [View All User Permissions](#)
- [Maintaining Groups](#)
- [List of Available Permissions](#)
- [View and Manage locks](#)
- [Password Encryption](#) (only available for Oracle and SQL Server Repositories).

Other Security tasks can be performed by the users who do not have Administrative rights, these tasks include:

- [Locking Model Elements](#)

- [Locking Packages](#)
- [Apply a User Lock](#)
- [Managing Your Own Locks](#)

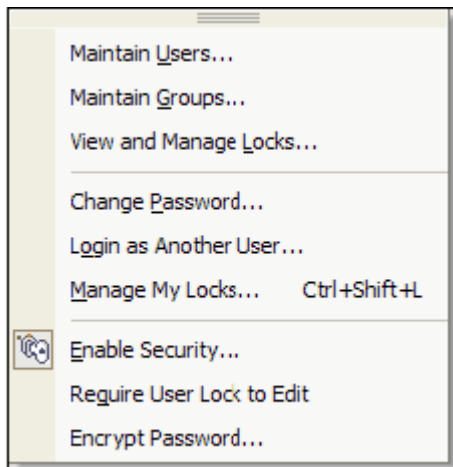
Note: User security is not enabled by default in EA you must [enable it](#) first.

7.11.4.1 Security Policy

There are two possible security policies in Enterprise Architect:

1. In the first mode, all elements and diagrams are considered unlocked, and as the need arises, a user may lock any element or set of elements at the user or group level. This mode is good for cooperative work groups where there is a solid understanding of who is working on which part of the model, and locking is used mainly to prevent further changes, or to limit who has access to a part of the model.
2. The second mode is more rigorous. It assumes everything is locked until explicitly checked out with a user lock. In this mode, an EA model is read-only until a user applies an editing lock to one or more elements. A single 'check out' function is available on a diagram to check out the diagram and all contained elements in one go. There are also functions on the context (right-click) menus of packages, diagrams and elements in the model browser to apply a user lock when this form of mode is in use. You would use this mode when there is a strict need to ensure only one person can edit a resource at one time. This is suitable for much larger projects where there may be less communication between users.

Toggle between these modes using the *Require User Lock to Edit* menu option in the *Project | Security* submenu:



Note: When you add new elements in Mode 1 (elements editable by default), no user lock is created automatically for the newly created element.

Note: When you add new elements in Mode 2 (elements locked by default), a user lock is created on the new element to allow instant editing.

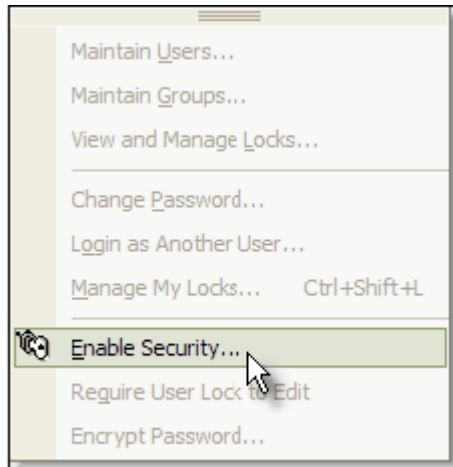
7.11.4.2 Enabling Security

User security is not enabled by default in EA. To enable security you can follow the instructions below.

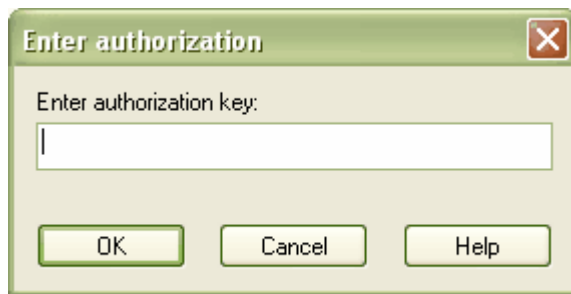
Enable Security

To enable security in EA, follow these steps:

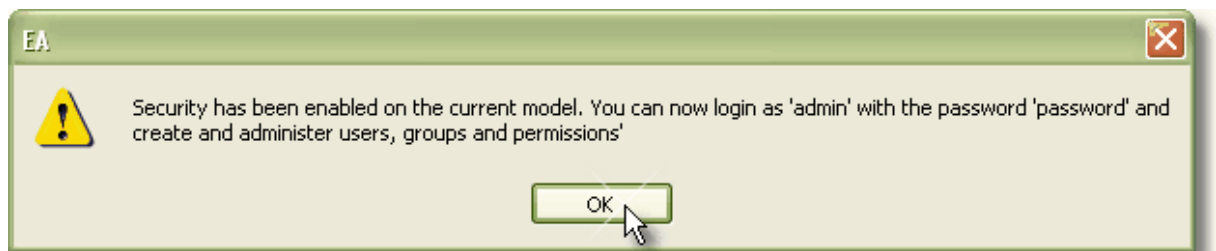
1. Open the *Project | Security* submenu.
2. Select the *Enable Security* option.



3. In the Authorization dialog that appears, enter the key provided when you purchased EA.



4. If you enter the correct code and press **OK**, security is enabled. An Admin user is created with full permissions and a password of 'password'.
5. Close EA then reopen it, logging in as Admin.
6. You may now set up users and permissions as required.



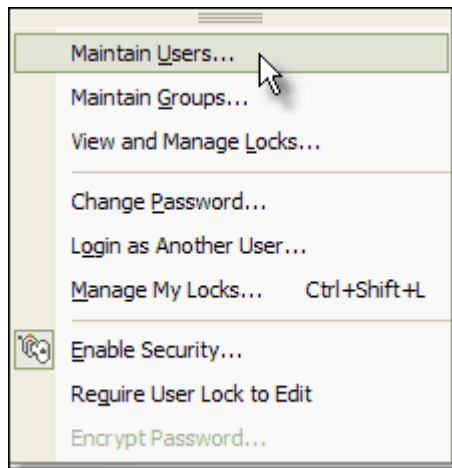
7.11.4.3 Maintaining Users

If you enable security, you will have access to the *System Users* dialog which you can use to set up more users for your model.

Set Up a User

To set up a user for your model, follow the steps below:

1. From the *Project | Security* submenu, select the *Maintain Users* option.



2. The *System Users* dialog will appear.

System Users

User Details

Login: admin

Firstname: The

Surname: Administrator

Department: Administration

Change Password

Set Permissions

Group Membership

Single Permissions

View All

Users: New Save Delete

Surname	Firstname	Login
Administrator	The	admin

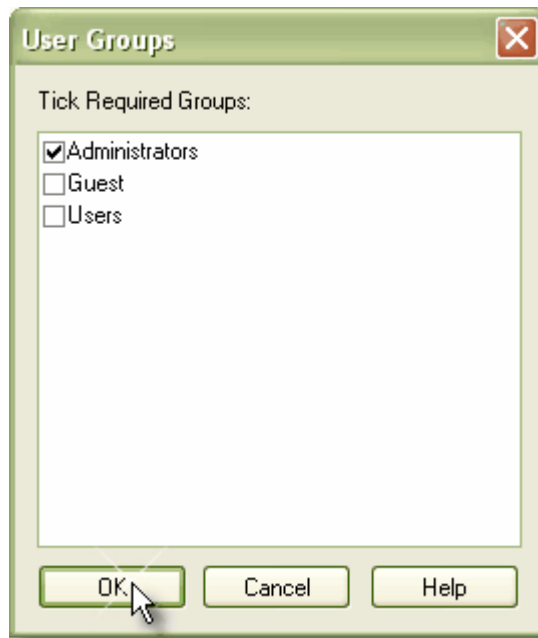
Help Close

3. You can use the *System Users* dialog to set up new users by providing their name and other details. You can also assign them to groups, set up their *Single Permissions* or *View All* permissions for the currently selected user

7.11.4.4 Set Up User Groups

To set up user groups follow these steps:

1. From the *Project | Security* submenu, select the *Maintain Users* option.
2. Click on *Group Membership* on the *System Users* dialog.
3. In the *User Groups* dialog, tick all of the groups this user will belong to.



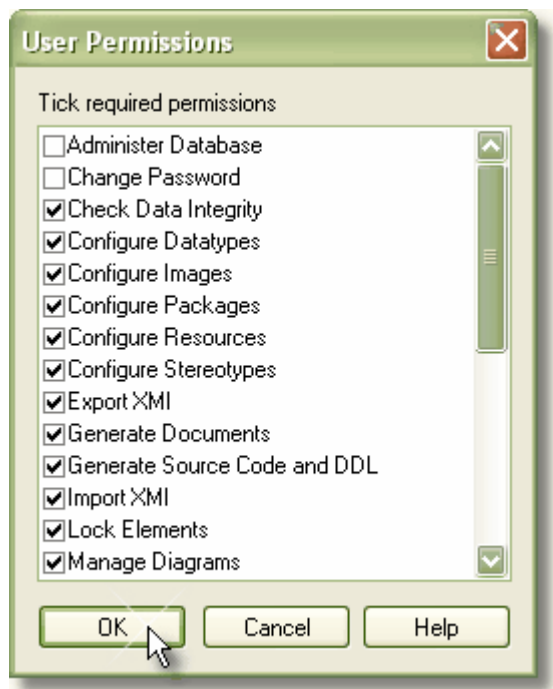
4. Press **OK** to assign the user to the group/s.

Note: To create new user groups, see the [Maintaining Groups](#) topic.

7.11.4.5 Set Up Single Permissions

You can set specific user permissions from the *User Permissions* dialog. Specific user permissions are added to permissions from group membership to provide an overall permission set. To set up single permissions for a user follow the steps below:

1. From the *Project | Security* submenu, select the *Maintain Users* option.
2. Click on *Single Permissions* on the *System Users* dialog.
3. In the *User Permissions* dialog, select the specific permissions that you want to apply to this user.

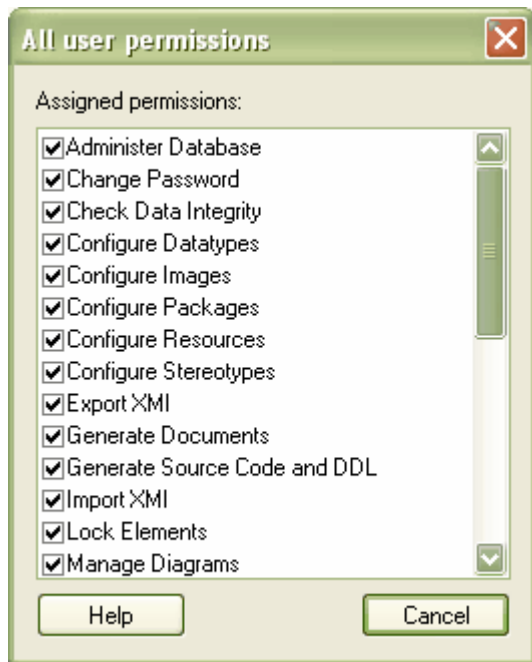


4. Press **OK** to assign the selected permissions to the user.

Note: A user's total permissions are those granted by Group Membership + those granted by Specific permission assignment.

7.11.4.6 View All User Permissions

The **All user permissions** dialog shows a list of all permissions a user has - derived from their individual profile and from their membership of security groups. It can be found by selecting **Maintain Users** from the **Project | Security** submenu, then selecting the required user and clicking on **View All**.



7.11.4.7 Maintaining Groups

Security groups make it easy to configure sets of permissions and have them apply to a number of users in one action.

Set Up a Security Group

To set up a security group, follow the steps below:

1. Select *Maintain Groups* from the *Project | Security* submenu.
2. The *Security Groups* dialog will appear.

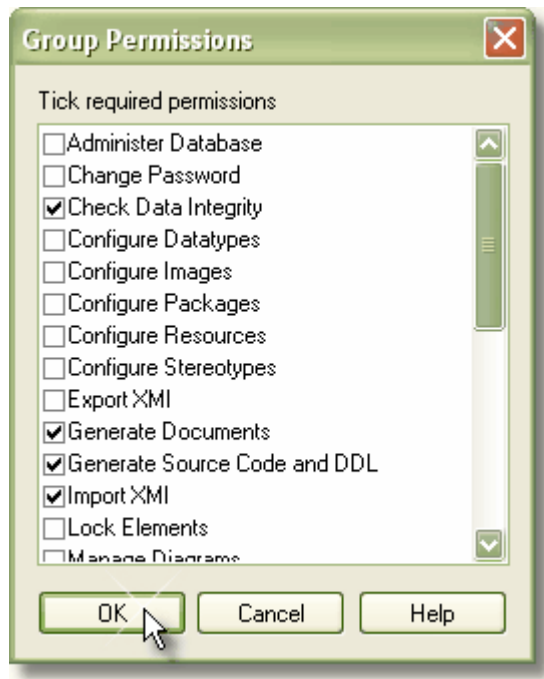


3. Specify a *Group Name* and a *Description*.
4. Press *Save*.

7.11.4.8 Set Group Permissions

To set up permissions to apply to a security group, follow the steps below:

1. Select *Maintain Groups* from the *Project | Security* submenu.
2. Press *Set Group Permissions* on the *Security Groups* dialog.
3. In the Group Permissions dialog tick the required permissions.



4. Press **OK** to assign the permissions. All of the users assigned to this group will share in this set of permissions.

7.11.4.9 List of Available Permissions

The following table lists the available permissions in the Corporate Edition of EA and their meaning:

Permission	Meaning
Security - Enable/Disable	Enable or disable user security in Enterprise Architect.
Security - Manage Users	Maintain users, groups and assigned permissions.
Security - Manage Locks	View and delete element locks.
Manage Reference Data - Update	Update and delete reference items.
Update Diagrams	Update diagram properties and layout.
Administer Database	Compact and repair database.
Manage Replicas	Create and synchronize replicas.
Lock Objects	Lock an element.
Manage Project Information	Update and manage resources, metrics, risks etc.
Configure Resources	Import and manage resources tab items: patterns, profiles, favorites etc.
Update Elements	Save changes (including delete) for elements, diagrams, packages, links, etc.
Export XMI	Export model to XMI.
Import XMI	Import model from XMI.
Manage Tests	Update and delete Test records.
Manage Issues	Update and delete Issues.
Change Password	Change password of current user.
Transfer Data	Transfer model between different repositories.
Check Data Integrity	Check and repair model integrity.
Configure Datatypes	Add/Modify and Delete Datatypes.
Configure Stereotypes	Add/Modify and Delete Stereotypes.
Configure Images	Configure alternate element images.
Generate Source Code and DDL	Generate source code and ddl from model element. Synchronize if already exists.
Reverse Engineer from DDL and Source Code	Reverse engineer from source code or ODBC. Synchronize with model elements.
Generate Documents	Generate RTF and HTML documents from model packages.
Configure Packages	Configure controlled packages and package properties.
Manage Diagrams	Create new diagrams, copy existing and delete diagrams. Also save diagram as UML Pattern.
Spell Check	Spell check package and set spell check language.
Configure Version Control	Set up Version Control options for the current model.
Use Version Control	Check files in and out using Version Control.

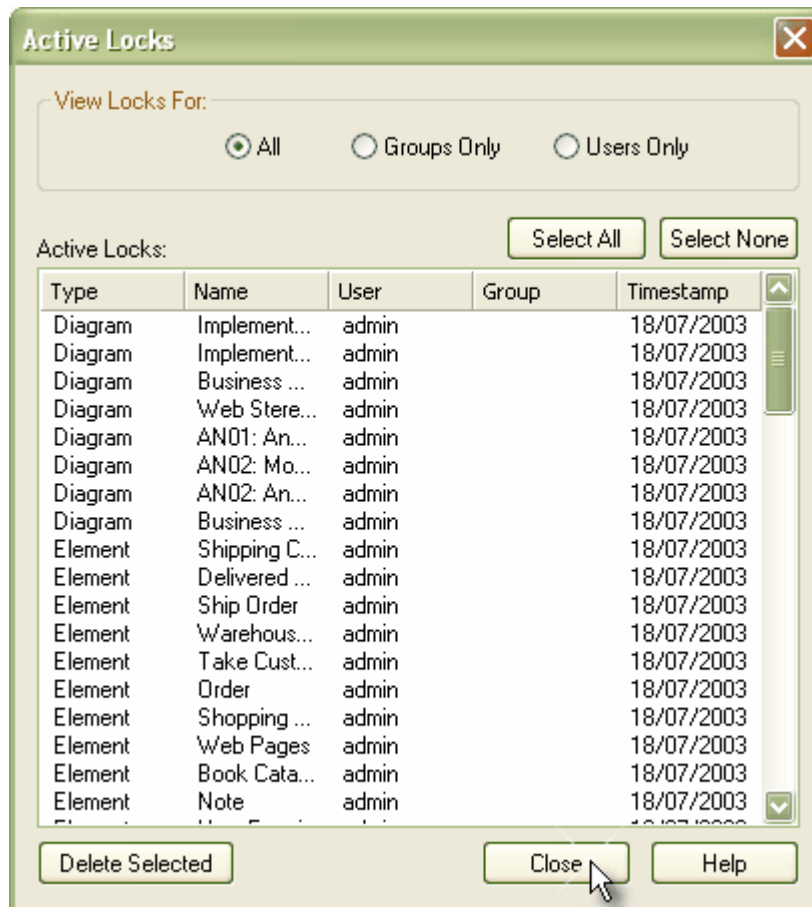
7.11.4.10 View and Manage Locks

From time to time it may be necessary to examine or delete locks placed on elements by users. To do this, a function is provided to view and manage active locks

Delete a Lock

To view locks, and delete if necessary, follow the steps below:

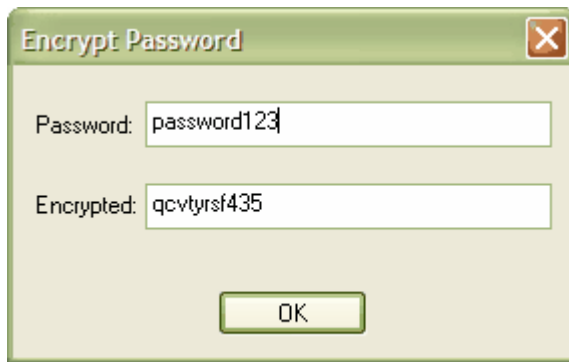
1. Select *View and Manage Locks* from the *Project | Security* submenu.
2. From the *Active Locks* dialog (shown below), select the type of lock to view - All locks, Group locks only or User locks only.
3. To delete a lock, highlight it and press *Delete Selected*.
4. *Close* the dialog when finished.



7.11.4.11 Password Encryption

Users or SQL Server or Oracle repositories have the option of encrypting the password used to set up the connection of EA to the repository. The EA user will not have the real password thereby preventing the user accessing the repository using other tools such as Query Analyzer or SQLPlus.

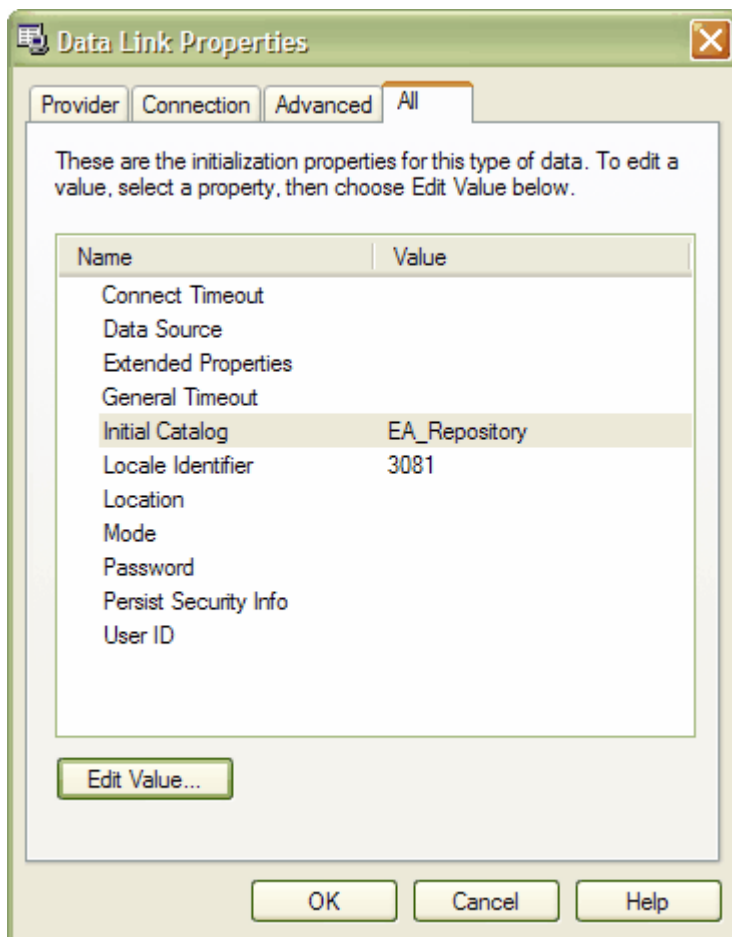
Once security is enabled, the administrator needs to log on to access the dialog to create encrypted passwords. To encrypt a password open the *Project* menu, then mouse over the *Security* menu option and choose *Encrypt Password* from the submenu. In the example below the Password "password123" is used as the password to access the repository.



To connect EA to the repository, the user enters the encrypted password prefixed with \$\$, so the encrypted password will become '\$\$qcvtysf435'. Do not use the '*Test Connection*' button as it will cause an error with encrypted passwords.

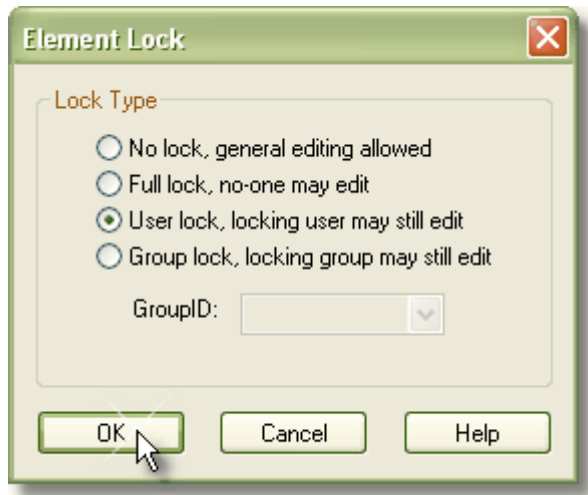
For more information relating to connecting to Oracle9i and SQL Server look at the to [Oracle9i Data Repository](#) and Connect to [SQL Server Data Repository](#) topics respectively.

Note: For SQL Server repositories, the user will need to enter the Initial Catalog details from the *All* tab of *Data Link Properties*.



7.11.4.12 Locking Model Elements

With security enabled, locking a model element is slightly different to when security is not enabled. If you have the *Project | Security | Require User Lock to Edit* option turned off, when you elect to lock a diagram or element, the *Element Lock* dialog will appear:



There are 4 lock options available:

- No Lock. Do not lock this element - clear any existing lock.
- Full lock. Lock this element so no-one can edit it.
- User lock. Lock this element so only the locking user may make further edits.
- Group lock. Lock this element so any member of the group specified may update the element - but others are excluded.

Select the appropriate lock and press *OK*.

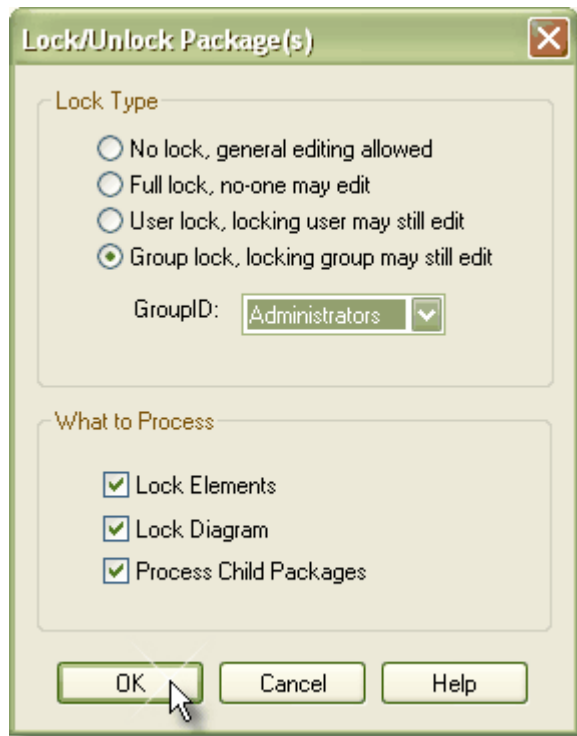
7.11.4.13 Locking Packages

You can lock all the contents of a package (and optionally all contents in child packages) in one step, using the *Lock Package* function. The locks are automatically applied to elements and to diagrams - as if they had been individually set or cleared. Lock types and details are the same as for [locking a single element](#).

Lock a Package

To lock a package, if you have the *Project | Security | Require User Lock to Edit* option turned off, follow these steps:

1. In the Project Browser, right click on the package you wish to lock.
2. Select the *Lock Package Contents* menu option.
3. In the *Lock/Unlock Package(s)* dialog, select the *Lock Type* to apply, whether to lock element and/or diagrams, and whether to process package children (ie. lock the whole branch).



4. Press **OK** to apply the lock.

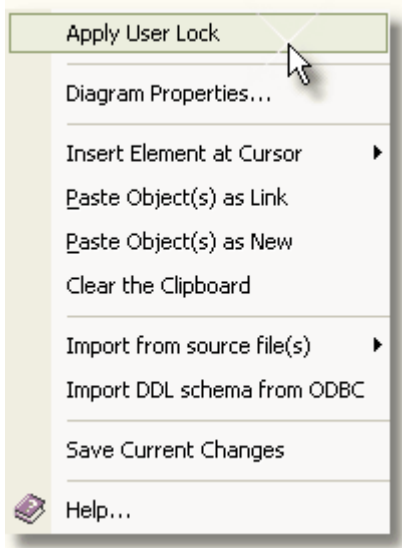
7.11.4.14 Apply a User Lock

In Mode 2 Security, where a User Lock is required before any edit can occur, you can use the following menu functions.

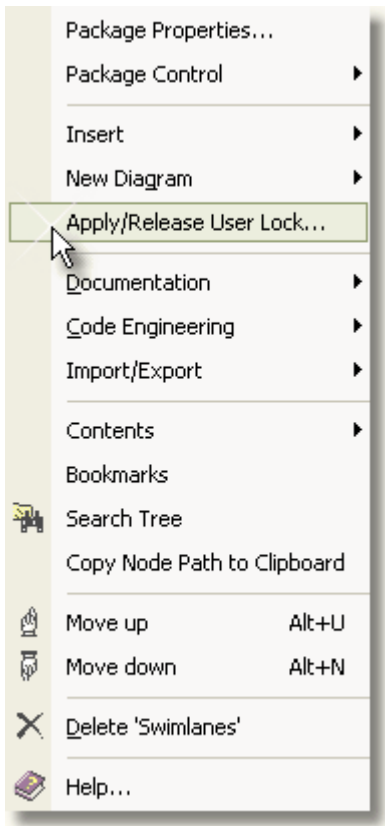
Right click on a diagram or an element to open its context menu. Select the **Apply** or **Release User Lock** option.

Note: The menu option will be **Apply** if there is no user lock currently on, it will be **Release** if there is currently a user lock applied.

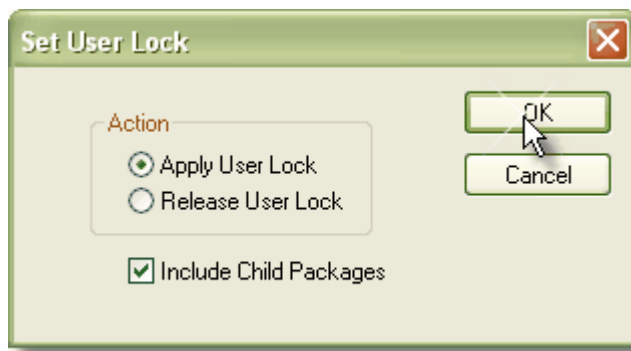
EA will apply or release the lock for the diagram and for any elements contained in the diagram.



In the Project Browser you can right click on Packages, Diagrams and Elements and apply a User Lock to the selected item.



When selecting items from the Project Browser, you will get the following dialog. Tick whether you wish to apply or release a user lock on the selected item.

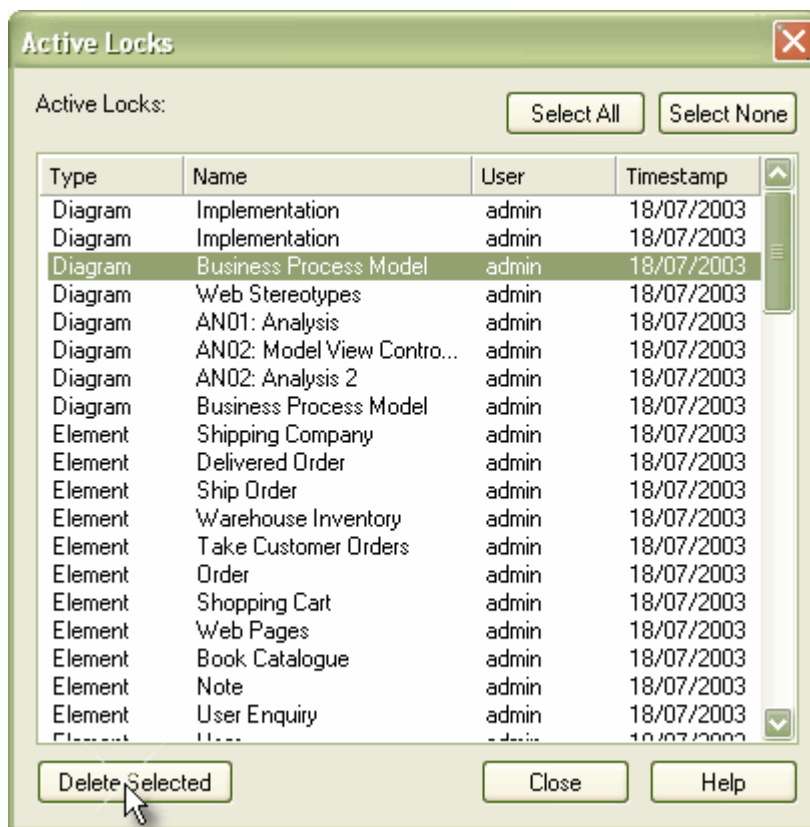


Note that for a package, you can elect to also lock all child packages at the same time. If any elements in the package tree are locked by other users, you will receive a list of elements that couldn't be locked at the end of the process.

7.11.4.15 Managing Your Own Locks

It is possible to view and delete your own user level locks in EA. This is especially useful when working in Mode 2 security (user locks required to edit).

To manage your locks use the *Project* | *Security* | *Manage My Locks* menu item. This will open the *Active Locks* dialog.



In the *Active Locks* dialog you can select one or more locks and delete it by pressing *Unlock Selected*.

7.11.5 Replication

In addition to sharing Enterprise Architect projects in real time over a network, projects may also be shared using *Replication* - options for which are available through the *Tools | Manage .EAP File* submenu.

Replication allows different users to work independently of one another, and to merge their changes at a later time. To avoid difficulties in this inevitably hazardous process, please read all topics in this section carefully.

Enterprise Architect Merge Rules

Enterprise Architect uses follows these rules in merging:

- Additions are cumulative - i.e. Two replicas creating 3 new classes each will result in 6 new classes after merging.
- Deletions prevail over modifications, if one replica changes a class name and other deletes the class, performing a merge will result in both files losing the class.
- Conflicting modifications appear in the "Resolve Replication Conflicts" dialog under *Tools | Manage EAP File*.

See [Resolving Conflicts](#) for details on how to deal with conflicting modifications.

Using Replication

To use replication, follow these steps:

1. Convert the base project to a [Design Master](#) using the *Make Design Master* option in the *Tools | Manage .EAP File* submenu).
2. Create replicas from the design master using the *Create New Replica* option in the *Tools | Manage .EAP File* submenu.
3. Take the replica away and work on it as required, then bring it back for synchronization with the design master.
4. [Synchronize the replicas](#). During synchronization, all changes to both the master and the replica are propagated in both directions, so at the end they both contain the same information.

Avoid Change Collisions

If two or more people make changes to the same element, eg. a class, Enterprise Architect will arbitrarily overwrite one person's change with other other's. To avoid this, different users should work on different packages.

However, since Enterprise Architect does not enforce this rule, it is possible for users' work to conflict. To minimize the difficulties this causes, please note the following guidelines:

- If users are likely to have worked in the same area of the model - they should both witness the synchronization and confirm that they are happy with the net result.
- If small pieces of information have been lost, they should be typed into one of the merged models after synchronization.
- If a large piece of information has been lost - for example, a large class note that was overwritten by another user who had made a minor change to the same class use the [Resolve Replication Conflicts](#) dialog.

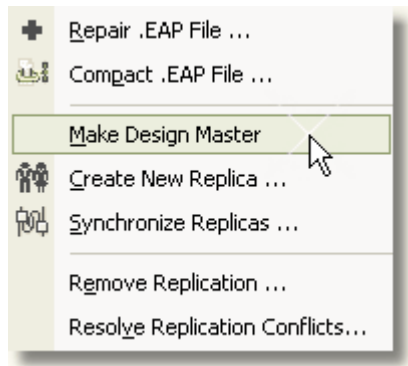
Disable or Remove Replication Features

If you have converted a project to a [Design Master](#) but now wish to disable the replication features, use the [Remove Replication](#) option in the [Tools | Manage .EAP File](#) submenu. Make sure you back up all your files first!

7.11.5.1 Creating Replicas

To create a replica, follow these steps:

1. First create a [Design Master](#), then select the [Create New Replica](#) option in the [Tools | Manage .EAP File](#) submenu and follow the instructions.



2. This process will create a replica of the current project which may then be modified independently, and afterwards re-merged with the main project.

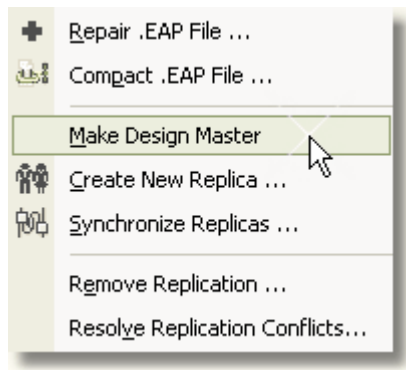
7.11.5.2 Design Masters

A *design master* is the first converted Enterprise Architect project that supports replication. From the design master you may create replicas, which may be modified independently of the master project, and re-merged at a later date.

Create a Design Master

To create a design master, follow these steps:

1. Load an Enterprise Architect project (always back up the original first!).
2. Select [Make Design Master](#) on the [Tools | Manage .EAP File](#) submenu and follow the instructions.



Tip: Replication is a powerful means of using Enterprise Architect in a work group or multi-user scenario.

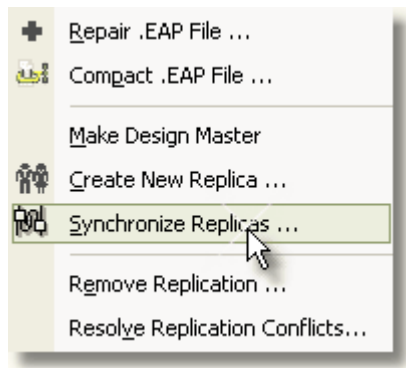
7.11.5.3 Synchronizing Replicas

To copy changes from one member of the replica set to another, you use the Synchronize function. Note that information will be copied both ways - including deletes, updates and inserts.

Synchronize a Replica

To synchronize a replica and a design master, follow these steps:

1. Open the Master project file.
2. Select *Synchronize Replicas* from the *Tools | Manage .EAP File* submenu to merge the open project and the selected replica back together.



Note: When you synchronize, both projects will end up containing identical information.

Change Collisions

Note that if two or more people work on the same element (package, diagram, etc.) then the replication engine has problems in resolving which change is the master. To avoid this, always work on separate areas in the model when you are using replicas. You may also use the *Resolve Replication Conflicts* option in the *Tools | Manage .EAP File* submenu.

7.11.5.4 Remove Replication

Replication makes many changes to the database structure of your model. As a consequence the model file becomes considerably larger with additional information. If you no longer require a model to be replicable, you can elect to remove all replication features.

Remove Replication

To remove replication, follow these steps:

1. If a Repository is not opened - it does not allow the menu option for removal of the replication (a temporary repository - not the one having replication removed) needs to open at the time. Ensure you have a repository opened at the time of creation.
2. From the *Tools | Manage .EAP File* submenu, select *Remove Replication*, to open the *Remove Replication Wizard*.
3. Enter the full path and file name of the project to have replication removed. Press *Next*.
4. Enter the full path and file name of the base EA model (with no replication) to act as template. Press *Next*.
5. Enter the full path and desired file name for the output file. Press *Next*.
6. Select whether you wish to have a log file created, and enter a file name for the log file.
7. Press *Run* to begin removing replication.
8. Enterprise Architect will create a new project containing all the model information.
9. Your model has now had replication removed, and should be considerably smaller.

7.11.5.5 Upgrading Replicas

With new releases of Enterprise Architect there may be changes to the underlying project structure - eg. more tables, changed queries, etc. If you are using replicas to share and work with EA projects, it is very important that you open the Design Master before opening any of the replicas with an updated version of EA.

Warning: Upgrading Replicas takes special care!

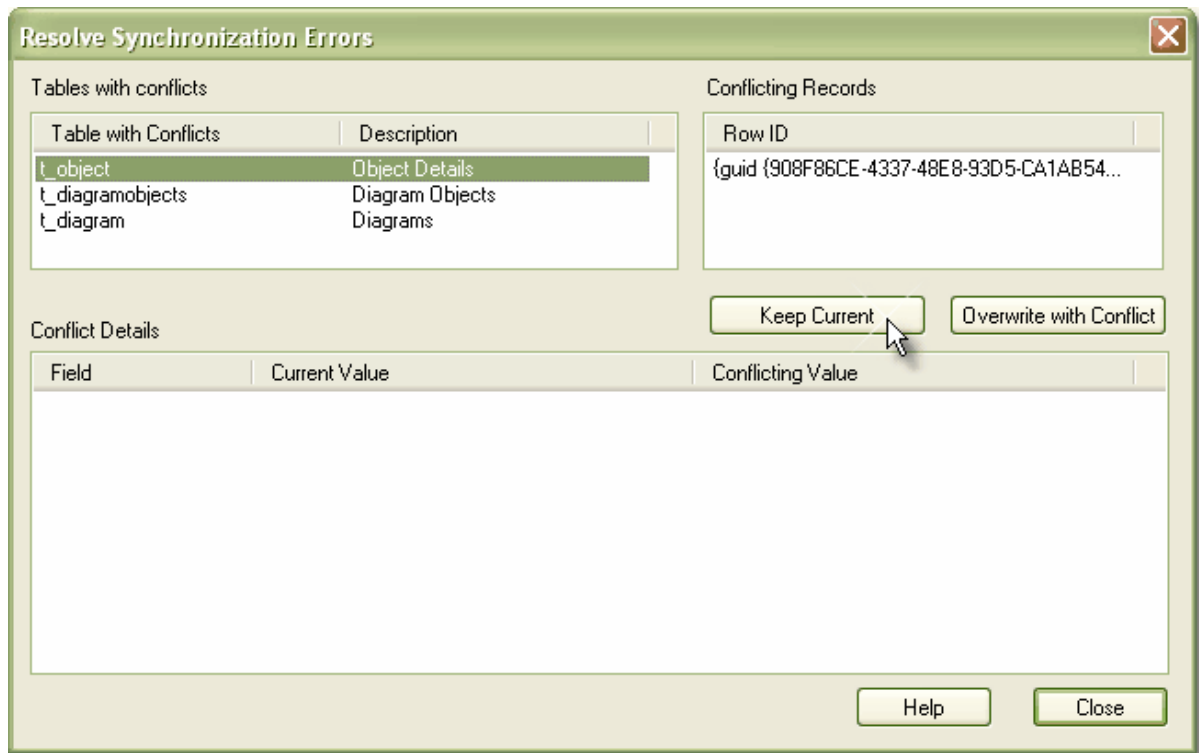
Changes to the database design in a replica set may ONLY be done to the Design Master. Next time the replicas are synchronized with the Master, the design changes are propagated through to the replicas. Trying to update a replica first will at best do nothing, and at worst cause the update of the Master to fail.

One other strategy is to remove replication from a copy of the replica set, upgrade that project and convert it into a new Design Master from which new replicas are created.

7.11.5.6 Resolving Conflicts

When two or more people have changed the same element between synchronization points, EA will have trouble resolving which change to accept and which to discard. A choice is made based on some rules within the JET replication manager, but the discarded changes are also stored so you may manually override that choice.

After synchronizing replicas, open the *Resolve Conflicts* dialog and determine if there were any conflicts. Select whether to accept each change or use one of the discarded changes instead.



Recommendations for Resolving Conflicts

Enterprise Architect stores model information in database records. When two records have been modified in different ways by different users, they appear in this dialog.

Normally it is not necessary or desirable to examine conflicts - since they represent relatively inconsequential pieces of information that may very easily be modified through the normal Enterprise Architect interface eg by moving a diagram element.

The only case in which this dialog box should be used is where there is substantial piece of information has been overridden by another user - and you want to retrieve

1. Click on an entry in the *Table with Conflicts* list which is likely to contain the lost information.
2. Click on each entry in *Conflicting Records* list.
3. When the lost information appears in the *Conflict Details* list, click on *Overwrite with Conflict* button.

7.12 Spell Checking

EA provides a powerful spell checking facility. This operates at the project level and allows you to quickly spell check an entire project.

See:

- [Using the Spell Checker](#)
- [Correcting Words](#)
- [User Dictionaries](#)
- [Select Language](#)

7.12.1 Using the Spell Checker

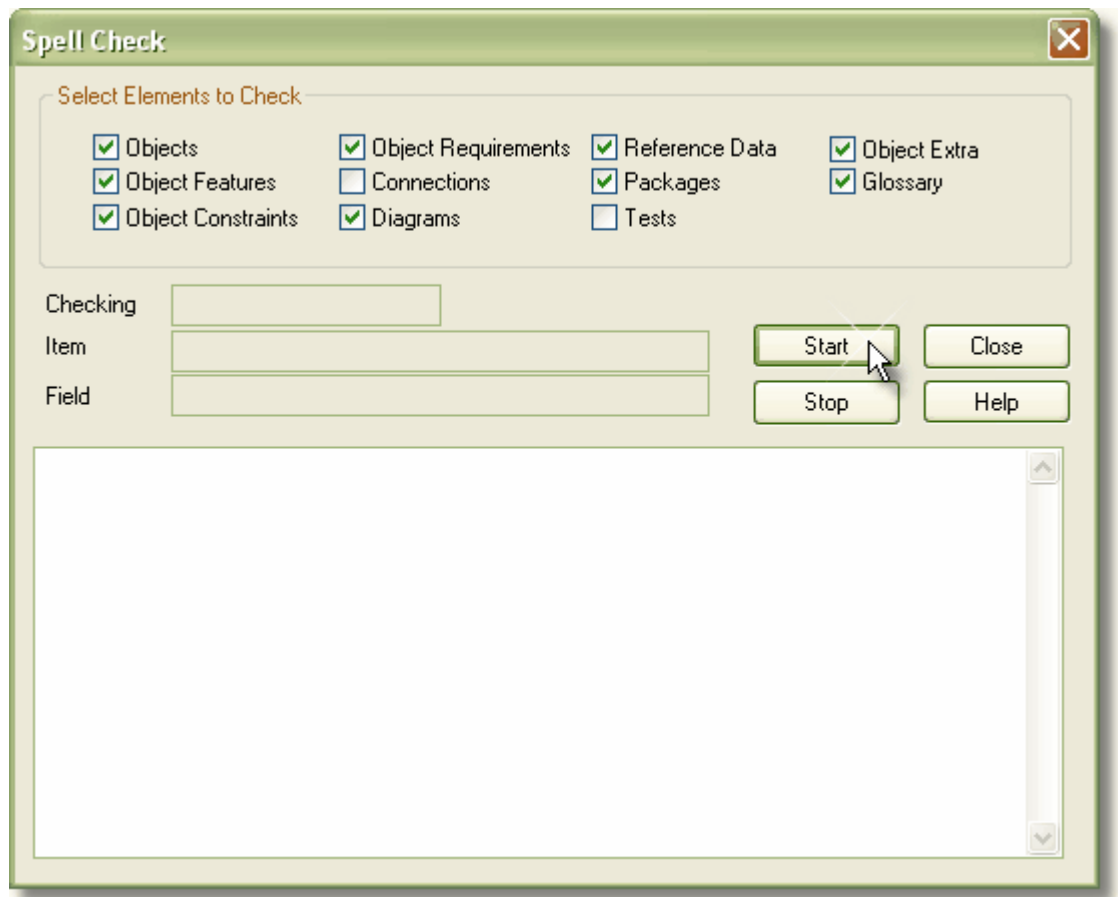
Enterprise Architect comes with an inbuilt spell checker.

Note: EA currently supports checking an entire model, and spell checking by single package. A future release will support more detailed spell checking at the element and diagram level.

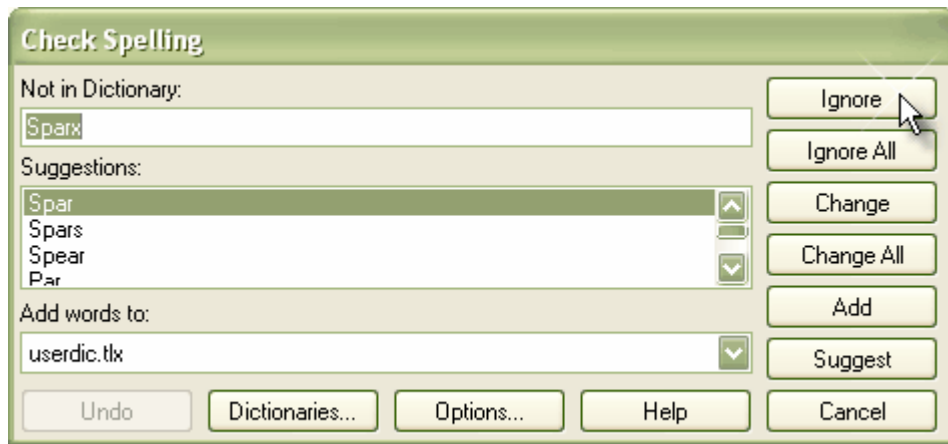
To perform a spell check, follow these steps:

1. From the **Tools** menu, select **Spell Check Model** or **Spell Check Current Package**, depending on which level of spell check you require
2. The **Spell Check** dialog will open. Select all the items that you wish to spell check within your model.

Note: The **Spell Check Model** option allows you to check spelling for the entire model, whereas **Spell Check Current Package** only checks the package currently open, and does not allow you to check the options shown below.

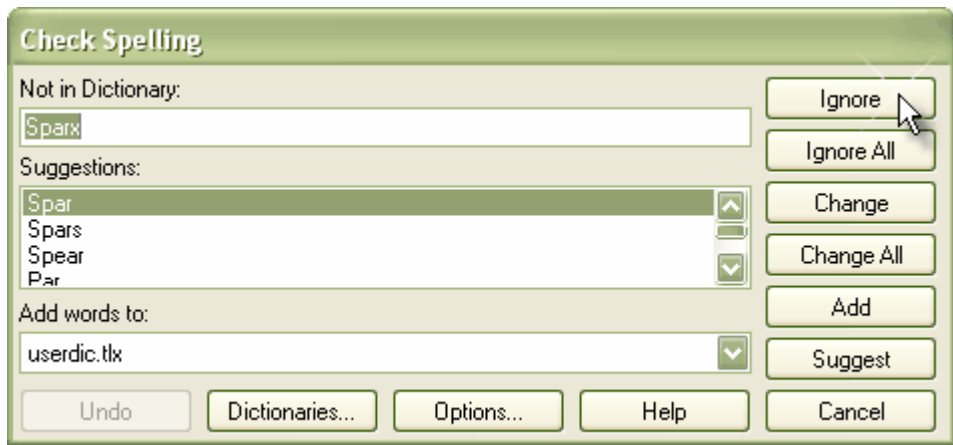


3. Press **Start** to begin the spell check.
4. As the spell check proceeds, you will see the text being checked in the visible edit area. If an error is detected, the word correction dialog will appear and offer you several **options** to correct the error.



7.12.2 Correcting Words

As the spell checking progresses, any errors or unknown words will be highlighted in the *Check Spelling* dialog. This allows you to correct the spelling of a word, ignore the error, add the word to a user dictionary, suggest alternatives - and otherwise assist in the spelling correction process.



To correct the current word you can:

- Modify the spelling by hand and press *Change* or *Change All*
- Accept the suggested alternative and press *Change* or *Change All*
- Press *Ignore* or *Ignore All* to ignore the word
- Press *Add* to add to the current user dictionary
- Press *Suggest* for alternatives
- Press *Cancel* to abort the spell check entirely

7.12.3 User Dictionaries

The inbuilt spell check stores user defined words in the *User Dictionary* (userdict.tlx) stored in the EA installation directory. During the spell check process, if you elect to add a word, it will be written into this file for later reference.

7.12.4 Select Language

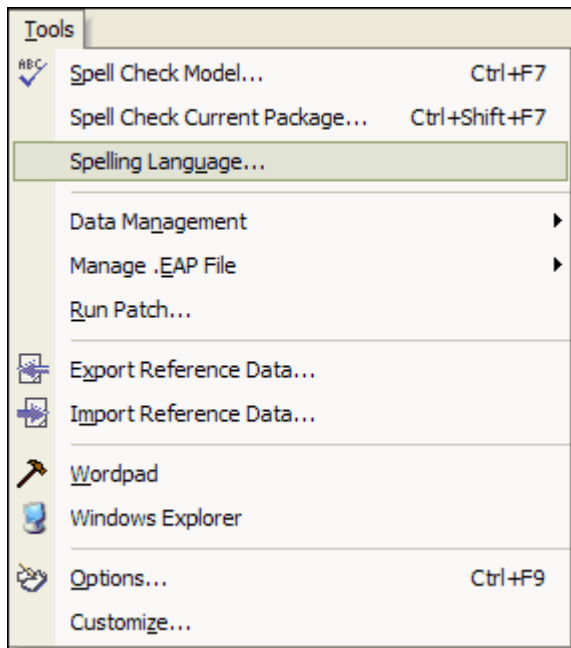
Enterprise Architect comes supplied with two dictionaries - US and British English. Additional dictionaries are available for download from the registered area of Sparx Systems. Once these have been downloaded and installed, you may select another language to perform the spell check in.

Before selecting a language as described below, ensure you have downloaded the additional language pack and installed it in the EA install directory.

Select a Different Language

To select another language:

1. From the *Tools* menu, select *Spelling Language*.

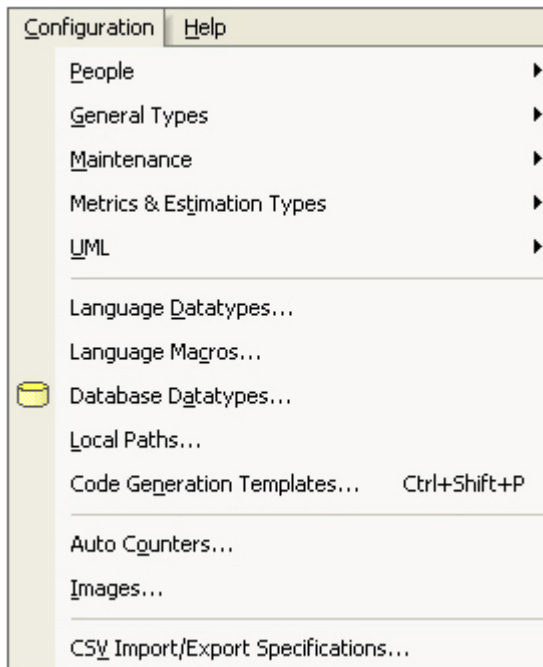


2. In the *Spell Check Language* dialog, select the required dictionary by checking the appropriate field.
3. Press *OK*. The selected language will remain the current language until changed.



7.13 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 *Configuration* menu.



See:

- [People](#)

- [General Types](#)
- [Maintenance](#)
- [Metrics and Estimation](#)
- [UML](#)
- [Data Types](#)
- [Import and Export Reference Data](#)

7.13.1 People

You can control the following for your project from the *Configuration | People* submenu:

- [Model Authors](#)
- [Clients](#)
- [Resources](#)
- [Roles](#)

7.13.1.1 Model Authors

The *Project Author(s)* dialog allows you to enter a list of project authors. This information is used to denote the author of specific elements.

To access this dialog, select *Model Authors* from the *Configuration | People* submenu.

Project Author(s)

Set up one or more project authors

Name:

Role:

Notes:

New Save Delete

Defined Authors

Name	Description
Geoffrey Sparks	Analyst, Designer Architect
Paul Mathers	C++ Programmer

Close Help

The options that are available for this dialog includes:

Element	Description
Name	The name of the person registered as a Project Author.
Role	The role(s) they play in the project (eg. Designer, Analyst, Architect) This role may be defined by using the Roles function or created in this dialog
Notes	Additional notes.
Defined Authors	List of authors already defined.

7.13.1.2 Clients

Clients are the eventual owners of the software system. Capture client details associated with the current model using the *Project Clients* dialog.

To access this dialog, select *Clients* from the *Configuration* | *People* submenu.

Project Clients

Information on clients

Name: Organisation:

Role(s):

Phone 1: Phone 2:

Mobile: Fax:

Email:

Notes:

Defined Clients

Name	Role(s)
Arthur Stoa	Project Manager
Joanna Stoa	Business Analyst
Margery Stoa	Assistant PM

The options that are available for this dialog includes:

Element	Description
Name	The name of the person listed as a resource. The resource Name is available for use in the Maintenance Workspace .
Organisation	The name of the organisation that the resource is employed by
Role(s)	The role(s) they play in the project (eg. Designer, Analyst, Architect).
Phone 1, Phone 2, Mobile, Fax	Contact Phone Numbers for the resource
Email	Email Address for the resource.
Notes	Additional notes.
Defined Clients	Clients that have already been defined.

7.13.1.3 Resources

Resources are model authors, analysts, programmers, architects etc. That is, anyone who may work on the system over time - either adding to the model -OR- programming and designing elements of the system outside EA. A variety of information about resources may be recorded using the **Resources** dialog.

To access this dialog, select **Resources** from the **Configuration | People** submenu.

Resources

Name: Organisation:

Role(s):

Phone 1: Phone 2:

Mobile: Fax:

Email:

Notes:

Available Resources

Name	Role(s)
Elosie Norman	Developer
Geoffrey Sparks	Solution Architect, Lead Designer, Project Manager
Paul Mathers	Lead Developer, Technical Architect

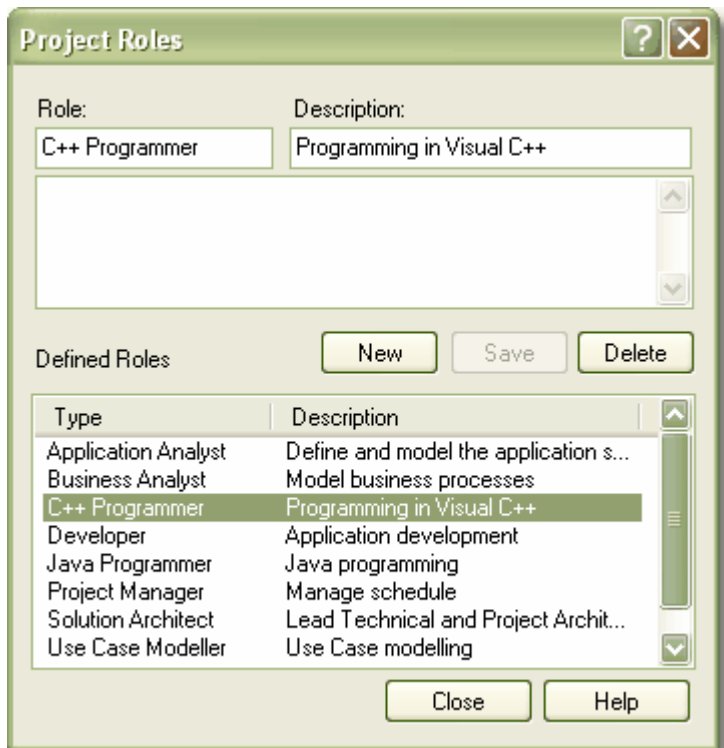
The options that are available for this dialog includes:

Element	Description
Name	The name of the person listed as a resource. The resource Name is available for use in Resource Management .
Organisation	The name of the organisation that the resource is employed by
Role	The role(s) they play in the project (eg. Designer, Analyst, Architect).
Phone 1, Phone 2, Mobile, Fax	Contact Phone Numbers for the resource
Email	Email Address for the resource.
Notes	Additional notes.
Available Resources	Resources that have already been defined.

7.13.1.4 Roles

Resources (People) associated with a project may play one or more *Roles* in the analysis, design or implementation. Use the *Project Roles* dialog to set the role types that are captured within EA. Examples of project roles include: application analyst, architect, developer and project manager.

To access this dialog, select *Roles* from the *Configuration* | *People* submenu.



The options that are available for this dialog includes:

Element	Description
Role	The name of the role. The role is available for use with the Model Author .
Description	A short description of the role
Notes	Additional information related to the role.
Defined Roles	Roles that have been previously defined including predefined roles.

7.13.2 General Types

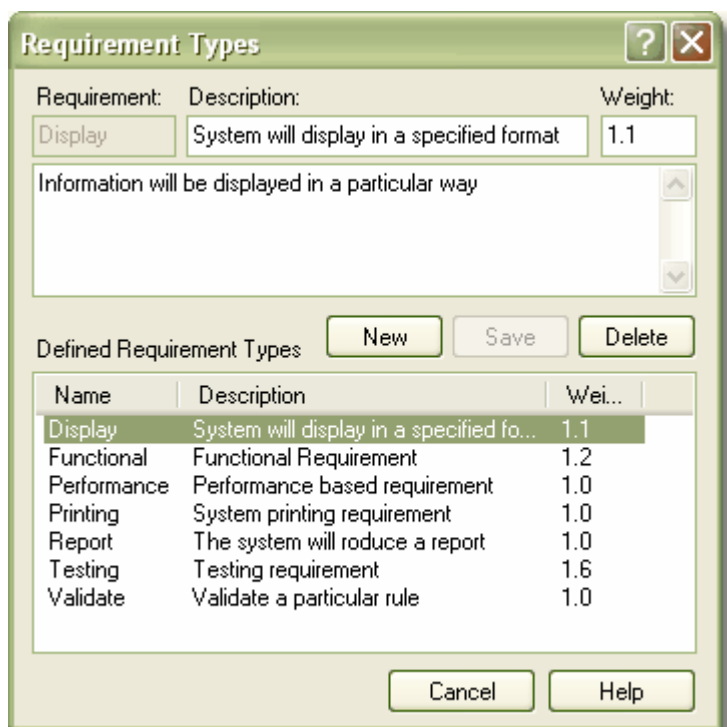
You can control the following for your project from the *Configuration | General Types* submenu:

- [Requirement types](#)
- [Status types](#)
- [Constraint types](#)
- [Scenario types](#)

7.13.2.1 Requirement Types

The *Requirement Types* dialog allows you to specify the generic set of requirement types that can be entered into the requirements sections of dialogs. This helps to maintain a single set of typed requirements.

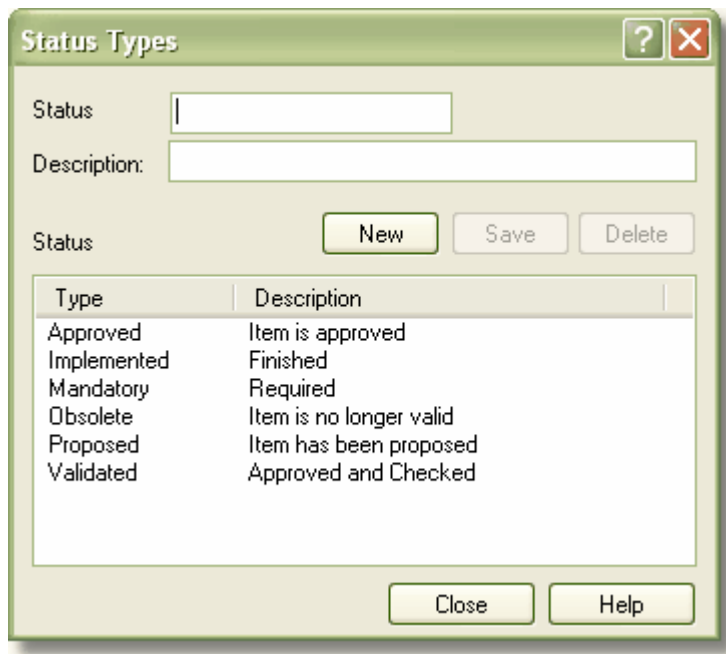
To access this dialog, select *Requirements* from the *Configuration | General Types* submenu.



7.13.2.2 Status Types

You can configure the basic list of *Status Types* used in EA. Note that not all dialogs use this list ... but most do. Set the status type and an optional description.

To access this dialog, select *Status Types* from the *Configuration* | *General Types* submenu.



7.13.2.3 Constraint Status Types

You can configure the basic list of *Constraint Status Types* used in EA. Set the status type.

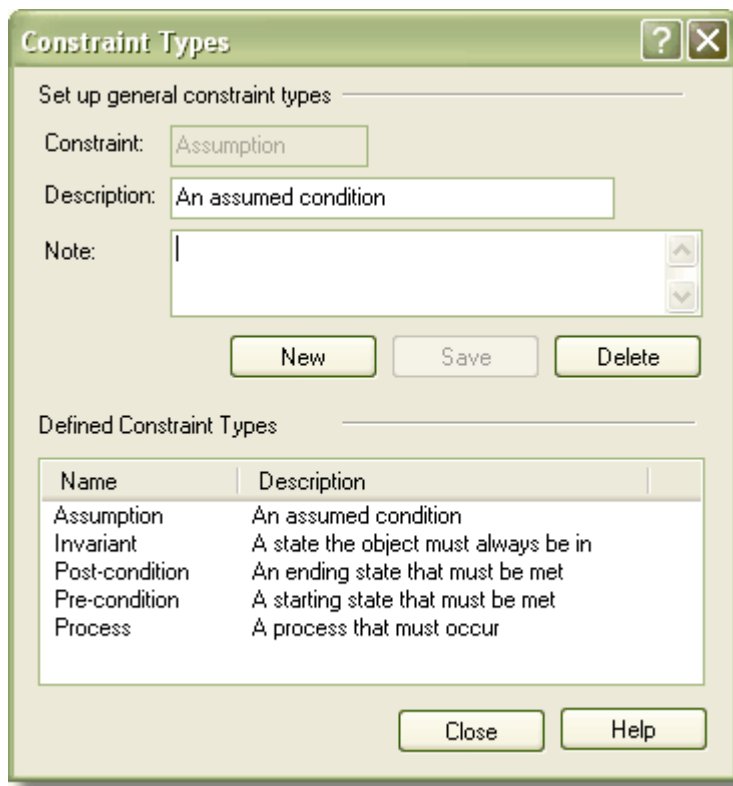
To access this dialog, select *Constraint Status Types* from the *Configuration* | *General Types* submenu.



7.13.2.4 Constraint Types

The *Constraint Types* dialog can be used for defining constraints. This will be picked up in a variety of places where constraints may fall into more categories than the basic Pre- Post- and Invariant conditions.

To access this dialog, select *Constraint Types* from the *Configuration | General Types* submenu.

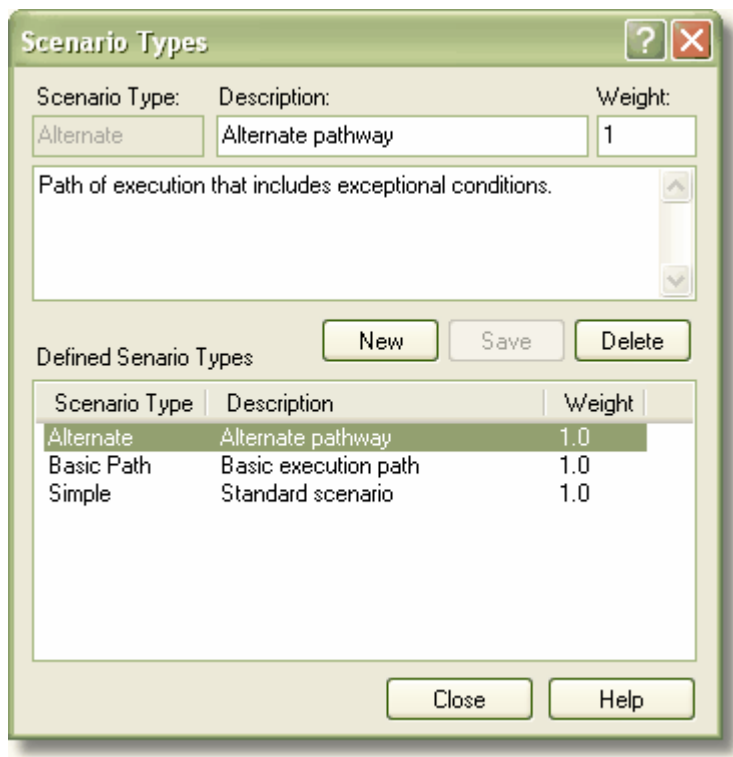


An example of an additional constraint type might be an *Assumption*.

7.13.2.5 Scenario Types

A drop down list of scenario types is available in the [scenario dialog](#). The standard types are Basic Path and Alternate Flow. Set additional scenario types using the [Scenario Types](#) dialog.

To access this dialog, select [Scenarios](#) from the [Configuration | General Types](#) submenu.



7.13.3 Maintenance

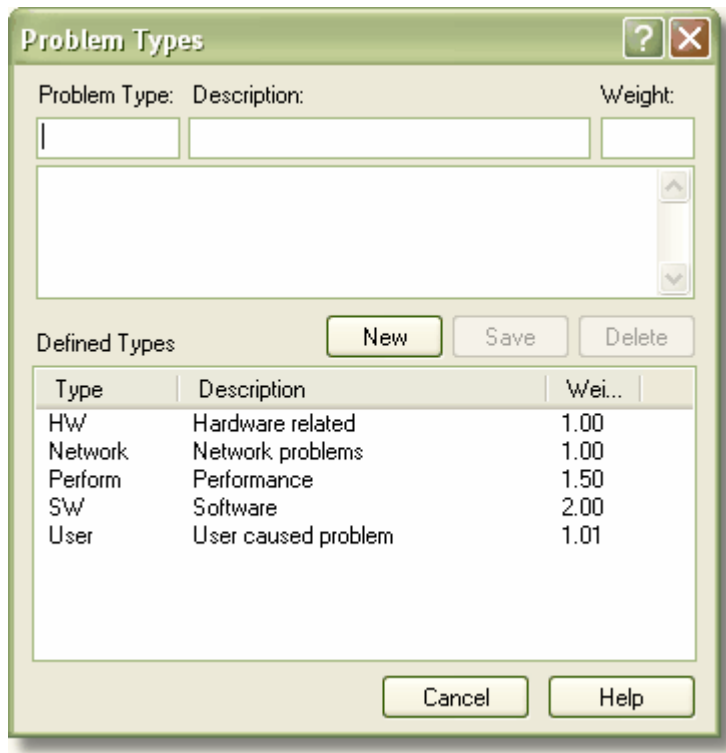
You can control the following for your project from the *Configuration | Maintenance* submenu:

- [Problem types](#)
- [Testing types](#)

7.13.3.1 Problem Types

For the maintenance and change control screens, you can set the base *Problem Types* that are handled. Examples are hardware related issues, performance problems, software bugs and network problems.

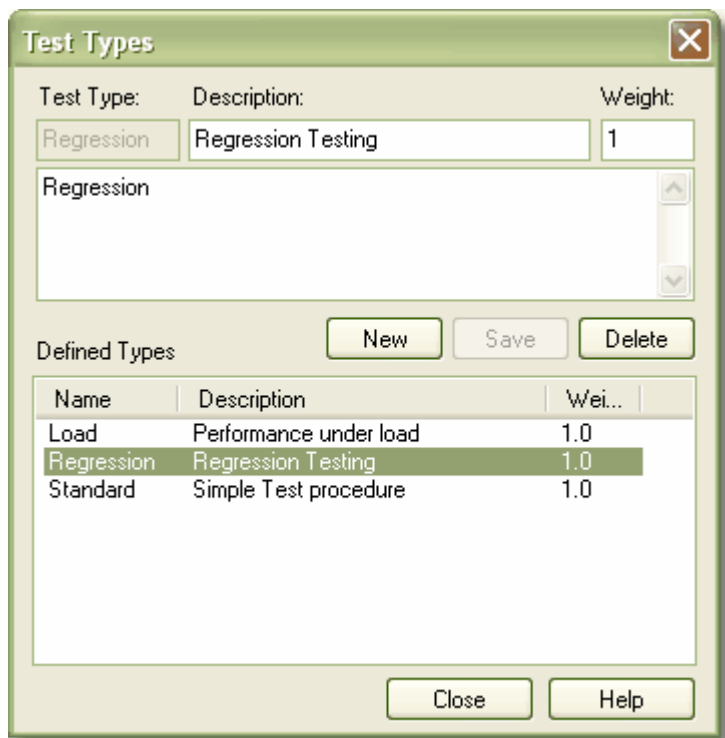
To access this dialog, select *Problems* from the *Configuration | Maintenance* submenu.



7.13.3.2 Testing Types

Use the *Test Types* dialog to add testing types to the basic set that comes with EA. Typical test types are load tests, performance tests, function tests, etc.

To access this dialog, select *Testing* from the *Configuration* | *Maintenance* submenu.



7.13.4 Metrics and Estimation

Risks, metrics, effort, TCF values, EFC values and Default Hour Rate for the project are controlled from the [Configuration | Metrics and Estimation](#) submenu. For further information on these, see the [Project Management](#) section.

7.13.5 UML

You can control the following for your project from the [Configuration | UML](#) submenu:

- [Stereotypes](#) - also see a list of [Standard Element Stereotypes](#)
- [Tagged Values](#)
- Cardinality

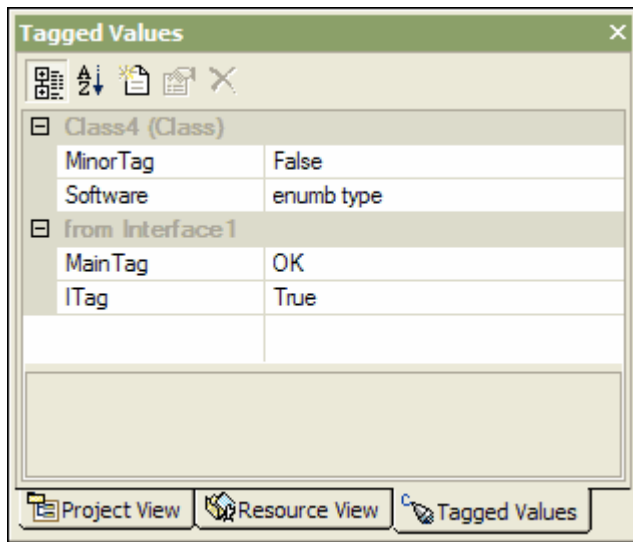
7.13.5.1 Standard Element Stereotypes

Stereotype	Base Class	Type
«access»	Permission	Stereotype
«appliedProfile»	Package	Stereotype
association	Association	Constraint
«association»	AssociationEnd	Stereotype
«auxiliary»	Class	Stereotype
«become»	Flow	Stereotype
«call»	Usage	Stereotype
complete	Generalization	Constraint
«copy»	Flow	Stereotype
«create»	BehavioralFeature	Stereotype
«create»	CallEvent	Stereotype
«create»	Usage	Stereotype
«derive»	Abstraction	Stereotype
derived	ModelElement	Tag
«destroy»	BehavioralFeature	Stereotype
«destroy»	CallEvent	Stereotype
destroyed	Association	Constraint
destroyed	Association	Constraint
disjoint	Generalization	Constraint
«document»	Abstraction	Stereotype
documentation	Element	Tag
«executable»	Abstraction	Stereotype
«facade»	Package	Stereotype
«file»	Abstraction	Stereotype
«focus»	Class	Stereotype
«framework»	Package	Stereotype
«friend»	Permission	Stereotype
global	Association	Constraint
«global»	AssociationEnd	Stereotype

7.13.5.2 Tagged Values

Tagged Values are used in a variety of places within EA to specify additional information about an element. Add default Tagged Value names using the *Tagged Values* dialog. These will appear in the drop list of Tagged Value names in the Tagged Value dialogs for elements, operations and attributes etc. For more information regarding the use of Tagged Values see the [Tagged Values Window](#) section.

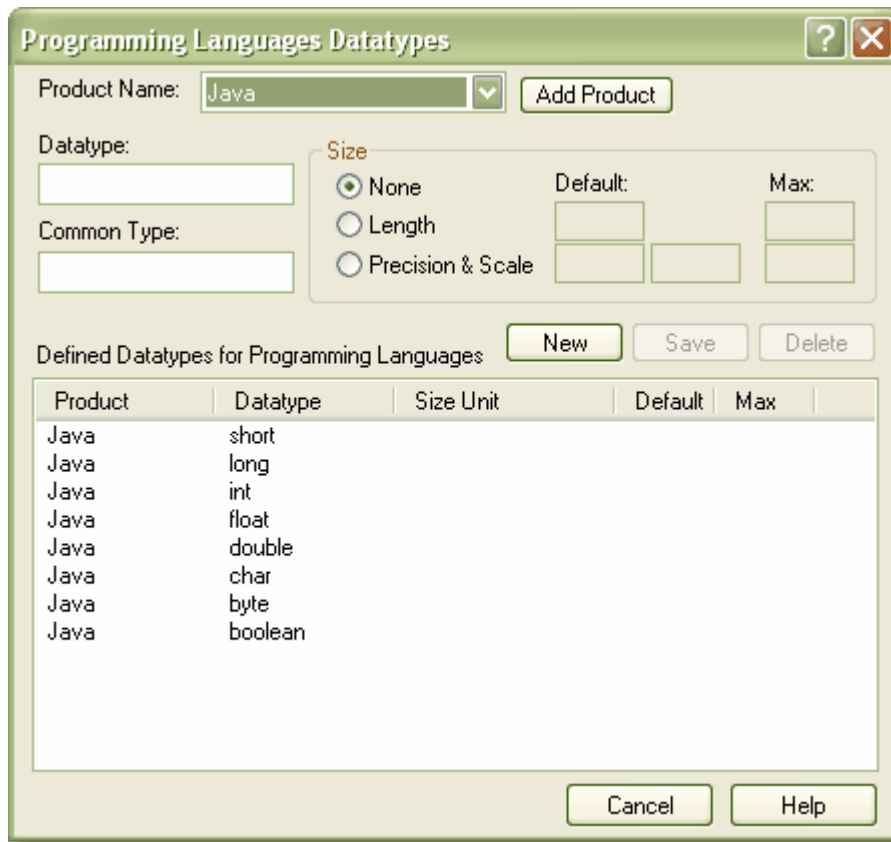
To access this dialog, select *Tagged Values* from the *Configuration | UML* submenu.



7.13.6 Data Types

Different programming languages support different inbuilt data types. The *Programming Languages Datatypes* dialog lets you extend and manage the set of inbuilt data types associated with a language as well as create new programming languages for use within EA.

To access this dialog, select *Language Datatypes* from the *Configuration* menu.



The options that are available for the Programming Languages Datatypes dialog includes:

Element	Description
Product Name	The name of the programming language.
Add Product	The Add Product button allows the user to add a new programming language to the dropdown fields for class elements within the EA model and allows the new language to be made available to the Code template editor once at least one datatype has been added to the language.
Datatype	The name for the datatype, this will be the language specific name of the datatype.
Common Type	The common type is the generic name of the datatype , for example the Java boolean datatype has a common datatype Boolean.
New	Create a new data type.
Save	Saves the newly created datatype.
Delete	Deletes custom datatypes, the predefined datatypes may not be deleted.

See:

- [Code Template](#)
- [Class Attributes](#)
- [Class Operations](#)

7.13.7 Import and Export Reference Data

Reference data (including Glossary and Issue information) may be imported and exported to XML files for convenient updating of models.

Examples of where this may be useful are:

- Copying glossaries from one model to another
- Adding additional stereotype profiles by merging new stereotypes into the model
- Updating reference data from files supplied by Sparx Systems as a maintenance release
- Copying resources, clients, etc. from one model to another

See:

- [Export Reference Data](#)
- [Import Reference Data](#)

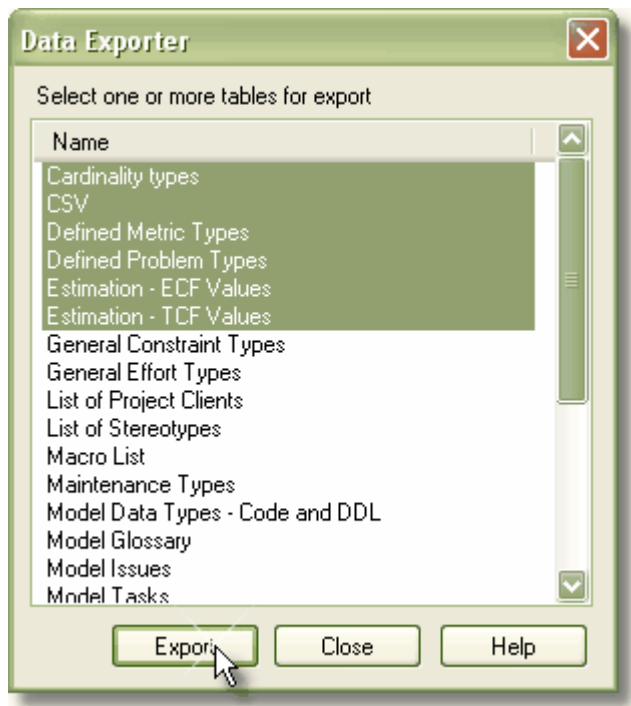
7.13.7.1 Export Reference Data

When reference and project data is exported, Enterprise Architect writes it out to a custom XML file. This includes table information, filter information, rows and columns.

Export Data

To export data, follow these steps:

1. From the **Tools** menu, select **Export Reference Data**.
2. In the **Data Exporter** dialog, select the table(s) you wish to export - you can select one or more tables for a single file.



3. Press **Export**.
4. Enter a valid file name with a .XML extension when prompted to do so.
5. This will export the data to the file. You can use any text or XML viewer to examine this file.

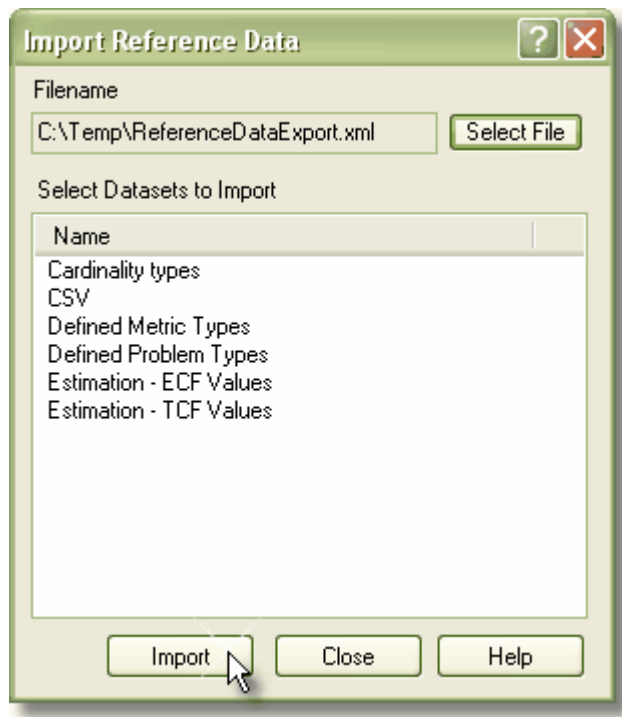
7.13.7.2 Import Reference Data

When you import data into Enterprise Architect, the system will merge the incoming data with the existing. If a record already exists it is updated to the new values - if not, a new record is added. Enterprise Architect never deletes records.

Import Data

To import data, follow these steps:

1. From the **Tools** menu, select **Import Reference Data**.
2. The **Import Reference** data dialog will appear.



3. Press **Select File** and select the filename to import data from - this must be an XML file produced by the EA [Data Exporter](#).
4. If you have entered the name of a valid file, a list of available tables to import will be displayed.
5. Select one or more of the tables to import.
6. Press **Import** to start the process - a message will pop up when the import is complete. Generally the process is quite fast.

Part



8 Project Management

Enterprise Architect provides some support for managing your project. Project managers may use Enterprise Architect to assign resources to elements, measure risk and effort and to estimate project size. Change control and maintenance are also supported (see [Maintenance Support](#)).

See:

- [Estimation](#)
- [Resources](#)
- [Testing](#)
- [Life Cycle](#)
- [Changes and Defects](#)
- [Model Tasks List](#)
- [Project and Model Issues](#)
- [Model 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 EA provides a starting point for estimating project effort. Using this facility you can get a rough measure of the complexity of a system and some indication of the effort required to implement the model. Like all estimation techniques, this one requires some experience with previous projects to 'calibrate' the process.

There is additional information available on use case metrics at www.sparxsystems.com.au/UCMetrics.htm.

Calibrating

The following values will need careful calibration in order to gain the best possible estimates:

- [Technical Complexity Factors](#) are values that attempt to quantify the difficulty and complexity of the work in hand.
- [Environment Complexity Factors](#) are values that attempt to quantify non-technical complexities such as team experience and knowledge.
- [Default Hour Rate](#) sets the number of hours per **use case point**.

Estimating

Once all of the calibration values have been entered, the project timescale can be estimated. This is done through the [Use Case Metrics dialog](#).

8.1.1 TCF Dialog

Technical Complexity Factors are used in determining the Use Case Metrics estimation technique. You may add or modify these using the TCF Values dialog.

Open this dialog by selecting **TCF Values** from the **Configuration | Metrics & Estimation Types** submenu.

Technical Complexity Factors

Set up Technical factors for estimation

Factor Number: Description: Weight: Assigned Value:

TCF04 Complex internal processing 1.00 4.00

New Delete Save

Defined Technical Types

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

Close Help Unadjusted TCF: 47.00

Defined Technical Types

The editable list should contain all factors that could affect the technical complexity of the project environment.

These configured factors, whose summed Ex Values yield the Unadjusted TCF, works together with the ECF, to skew the overall complexity up or down depending on the level of technical complexity and the corresponding level of environmental support.

Weight

The TCF weight factor indicates how much technical complexity you assign to a factor - eg. "the system will be developed in ADA" may warrant a higher TWF than "the system will be a shell script". A weight evaluates its respective factor, but is irrelative to a project; the value field assesses each factor's role within a project. The supplied factors and their associated weights are defined by the Use Case Points Method, although they may be adjusted to suit a project's specific needs.

Value

For most purposes, the only table column needing adjustment will be 'Value', which indicates the degree of influence a particular factor holds over the project. As a suggested gage, a value of '0' indicates no influence, a '3' indicates average influence, and a '5' indicates strong influence.

8.1.2 ECF Dialog

Environment Complexity Factors are used in determining the Use Case Metrics estimation technique. You may add or modify these using the ECF Values dialog.

Open this dialog by selecting *ECF Values* from the *Configuration | Metrics & Estimation Types* submenu.

Environment Complexity Factors

Set up Environment factors for estimation

Factor Number: Description: Weight: Value:

ECF02 Application experience 0.50 3.00

New Delete Save

Defined Environment Types

Type	Description	Wei...	Value	Ex V...
ECF01	Familiar with Rational Unified Process	1.50	4.00	6.00
ECF02	Application experience	0.50	3.00	1.50
ECF03	Object-oriented experience	1.00	4.00	4.00
ECF04	Lead analyst capability	0.50	4.00	2.00
ECF05	Motivation	1.00	3.00	3.00
ECF06	Stable requirements	2.00	4.00	8.00
ECF07	Part-time workers	-1.00	0.00	0.00
ECF08	Difficult programming language	-1.00	3.00	-3.00

Close Help Unadjusted ECF: 21.50

Defined Environment Types

The editable list should contain all factors affecting the general design and development environment, including team experience and knowledge, team size, expertise and other non-functional environmental factors.

These configured factors, whose summed Ex Values yield the Unadjusted ECF, works together with the TCF, to skew the overall complexity up or down depending on the level of technical complexity and the corresponding level of environmental support.

Weight

A weight evaluates its respective factor's complexity in comparison to other factors, but is irrelative to a project; the value field assesses each factor's role within a project. The supplied factors and their associated weights are defined by the Use Case Points Method, although they may be adjusted to suit a project's specific needs.

Value

For most purposes, the only table column needing adjustment will be 'Value', which indicates the degree of influence a particular factor holds over the project. As a suggested gage, a value of '0' indicates no influence, a '3' indicates average influence, and a '5' indicates strong influence.

8.1.3 Estimating Project Size

Enterprise Architect uses a simple estimation technique based on the number of Use Cases to be built, the difficulty level of those Use Cases, some project environment factors and some build parameters. Once the parameters are set up and the Use Cases defined, open the *Use Case Metrics* dialog by:

- Navigating to the package of interest and selecting *Use Case Metrics* from the *Project* menu

-OR-

- Right clicking on the package of interest in the Project Browser and selecting *Package Metrics* from the *Documentation* submenu

Use Case Metrics

Use Cases

Root Package: Use Case Model

Bookmarked: Phase like Keyword like

Reload All * Include Actors

Package	Name	Type	Com...	Pha
UC01-1: User Manage...	Register with Book Shop	UseCase	5.000...	1.0
UC01-1: User Manage...	Login	UseCase	5.000...	1.0
UC01-2: Select Books	Request UnListed Book	UseCase	5.000...	1.0
UC01-2: Select Books	Locate Book by Title o...	UseCase	5.000...	1.0
UC01-2: Select Books	Browse Book Catalogue	UseCase	10.00...	1.0
UC01-3: Manage Order	Pay for Order	UseCase	5.000...	1.0
UC01-3: Manage Order	View Cart	UseCase	5.000...	1.0
UC01-3: Manage Order	Remove Item from Cart	UseCase	5.000...	1.0
UC01-3: Manage Order	Add title to Cart	UseCase	5.000...	1.0

Unadjusted Use Case Points (UUCP) = Sum of Complexity 60

Use Cases 11 Ave Hours per Use Case Easy: 42 Med: 85 Diff: 127

Technical Complexity Factor

Unadjusted TCF Value (UTV): 47.1

TCF Weight Factor (TWF): 0.01

TCF Constant (TC): 0.6

TCF = TC + (TWF x UTV): 1.071

Environment Complexity Factor

Unadjusted ECF Value (UEV): 19.97

ECF Weight Factor (EWF): -0.03

ECF Constant (EC): 1.4

ECF = EC + (EWF x UEV): 0.8009

Package Estimated

Use Case Points (UCP) = UUCP * TCF * ECF = 60 * 1.071 * 0.8009 = 51 UCP

Default Hours Estimated work effort (hours) = 10 * 51 = 510 hours

Buttons: Recalculate, Report, Help, Close

Note: This technique is of value only once you have a had couple of known projects to use as a baseline. Please, **DO NOT** use the provided "guesstimates" as a real world measure until you have some real world base lines to measure against.

Element	Description
Root Package	The root package in the hierarchy. All use cases under here are included in the report (potentially) .
Include	Include tagged, non-tagged or all use cases.
Phase like	Include use cases with phase like (* = all). Use 1.* for 1.1 and 1.2 etc.
Tag contains	Place additional strings in the use case tag field and use this filter to only select use cases that have some specific text in the tag (eg. CORE;).
Technical Complexity Factor	This area lists parameters that describe the degree of technical complexity the project has. While the unadjusted TCF value comes from the TCF dialog, the other values compose the Use Case Points Method formula. These fields should be modified with caution. The final project estimate is directly proportional to the TCF.
Environmental Complexity Factor	This area lists parameters that calculate the degree of environmental complexity the project has, from factors such as programmer motivation or experience. The listed parameters compose the formula calculating the ECF, defined by the Use Case Points Method; the only parameter affected by the project is the unadjusted ECF value, derived from the ECF dialog. The final project estimate is directly proportional to the ECF.
UUCP	Unadjusted use case points = sum of use case complexity numbers.
Number of Use Cases	Total use case count in estimate.
Ave hours per use case	Averages the number of hours assigned to easy, medium and hard use cases - for information purposes only.
Package Estimate	Detailed breakdown of final figure - note that you will need to tailor the hours per use case point figure to the level that matches your type of project and capability based on known previous project outcomes.
Default Hours	Set the default hours that is fed into the final calculation.
Re-calculate	Re-run the estimate - usually after you change the hours/use case point number.
Reload	Re-run the search - usually after you change the filter criteria.
Report	Produce a rich text formatted report from the current estimate.

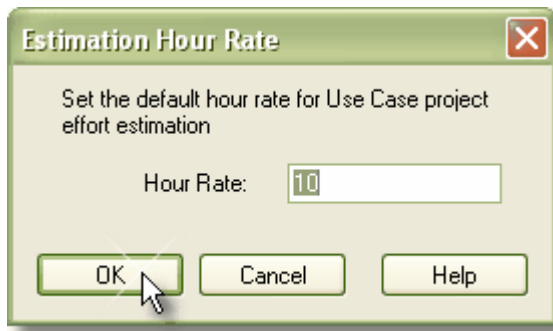
8.1.4 Default Hours

Set the default hour rate per adjusted use case point by using the *Estimation Hour Rate* dialog. Open this dialog by:

- Pressing *Default Hours* on the *Use Case Metrics* dialog.

-OR-

- Selecting *Default Hour Rate* from the *Configuration | Metrics & Estimation Types* submenu.



Enter the hour rate and press **OK** to change current value.

Note: The value you enter will be stored as a local setting on your computer only.

Setting an hour rate is the most difficult factor in an accurate estimation. Typical ranges can vary from 10-30 hours per use case point. Studying the Use Case Points Method, from which this variable is defined, can help to understand its role in the estimation and facilitate selection of a suitable initial value. The best way to estimate this value is through the analysis of previous, completed projects. By calculating the project estimation on a completed project whose use cases and environment are configured within EA, the hour rate can be adjusted to render an appropriate value for your unique work environment.

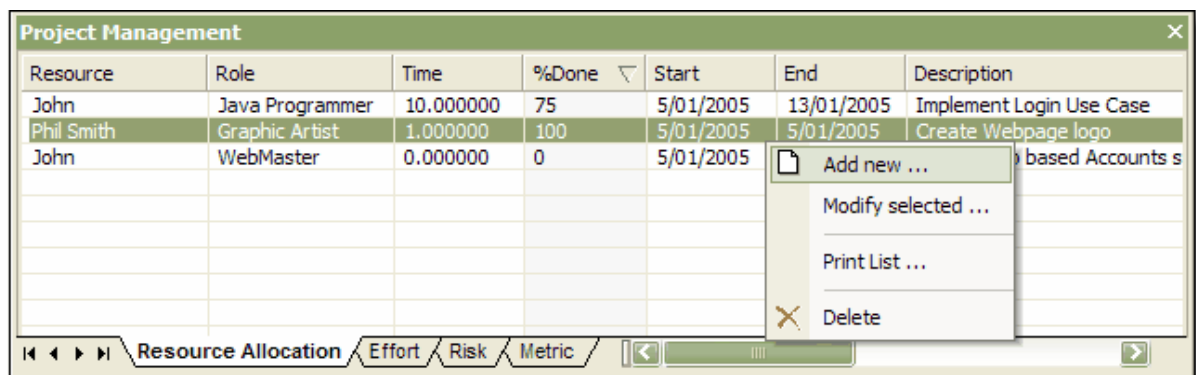
8.2 Resources

What is a Resource?

Resources are the people who work on a project. They can be assigned roles and allocated tasks, which allows for tracking of effort and estimation of time to complete.

Project Management Window

Resources are added, modified and deleted from the Project Management window. Open this window by either: selecting **Element | Resourcing Metrics & Risk** from the main menu; selecting **View | Other Windows | Project Management** from the main menu; or pressing **Ctrl + Shift +7**.



What to Do?

To find out more information about Project Management tasks, especially those involving :

- To allocate a resource to an element, see [Resource Management](#).
- To record additional project management information for an element, see:

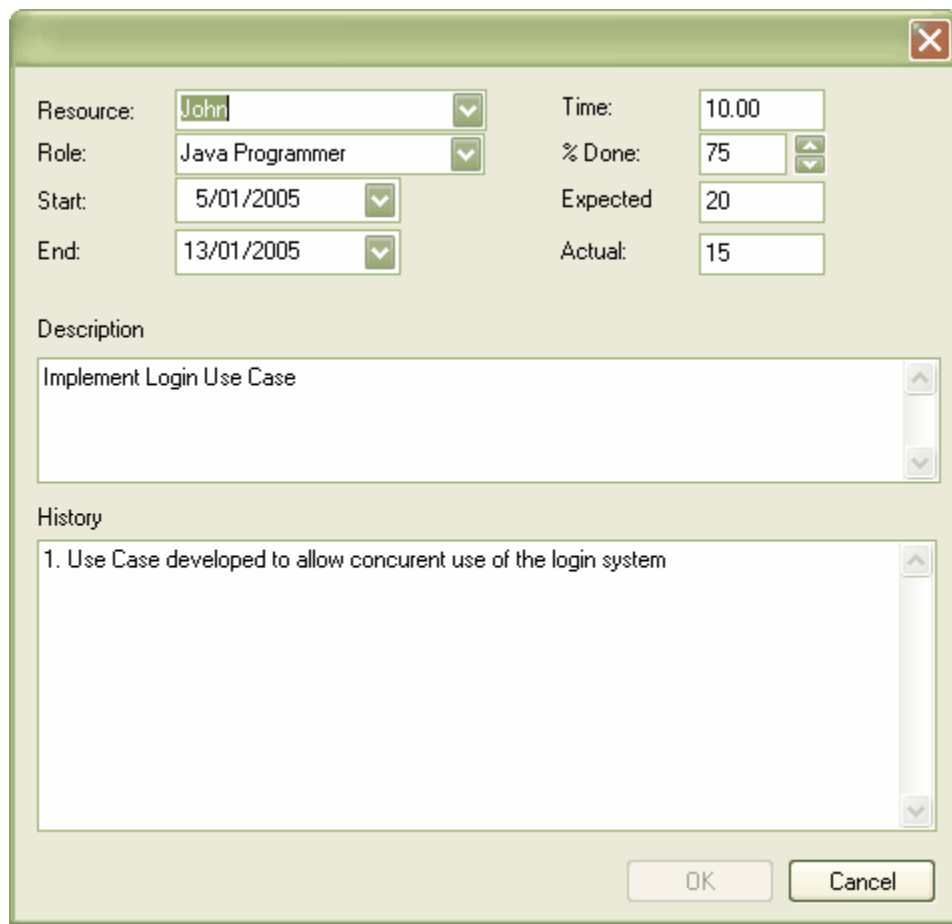
- [Effort Management](#) to record effort expended on the element.
- [Risk Management](#) to record risk associated with the element.
- [Metrics](#) to record metrics measured for an element.
- To obtain a report of resource allocation details, see [Resource Report](#).
- To configure Project Management data and populate the drop-down lists used on the Project Management dialogs, see:
 - [Roles](#).
 - [System Clients](#).
 - [Effort Types](#).
 - [Metric Types](#).
 - [Risk Types](#).

8.2.1 Resource Management

Enterprise Architect provides the ability to link a named resource in a named role to a given model element. This enables the project manager to track how far development of required components and classes has progressed (provided the programmers and others keep their figures up to date).

To enter Resource Allocation details for an element select the element and then choose the *Add new* or *Modify selected* options by right clicking on the *Project Management* window with the tab set to *Resource Allocation*. This will present the following dialog which allows the user to enter the following data:

- The name of the *Resource*
- The *Role* of the resource
- The *Start* and *End* date for the resource
- The *Time* allocated to the resource
- The *% Done* the resource has completed
- The *Expected* time allocated to the resource
- The *Actual* time expended on the resource
- The *Description*
- The *History* the resource



The dialog box displays the following information:

Resource:	John	Time:	10.00
Role:	Java Programmer	% Done:	75
Start:	5/01/2005	Expected:	20
End:	13/01/2005	Actual:	15

Description

Implement Login Use Case

History

1. Use Case developed to allow concurrent use of the login system

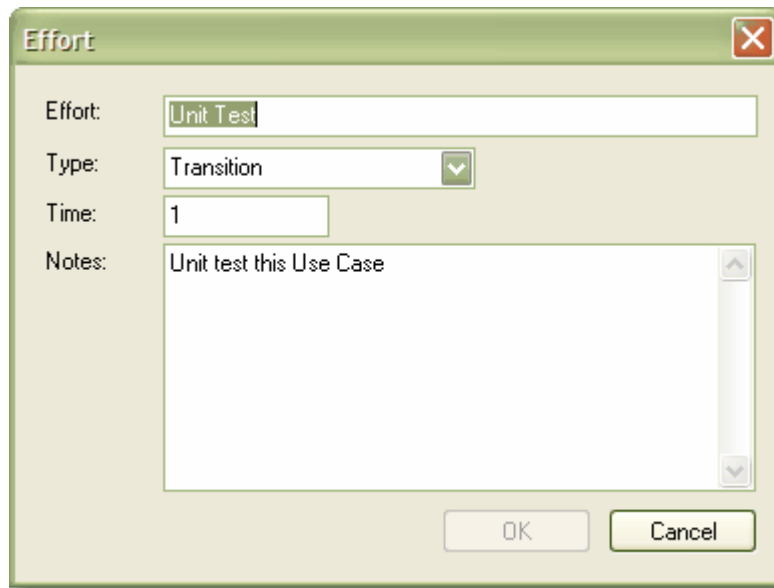
Buttons: OK, Cancel

8.2.2 Effort Management

To enter Effort details for an element select the element and then choose the *Add new* or *Modify selected* options by right clicking on the *Project Management* window with the tab set to *Effort*. This will present the following dialog which allows the user to enter the following data:

On the *Effort* tab you can enter the following information:

- A name for the *Effort* (short description)
- The *Type* of effort
- The *Time* the effort will expend
- Some *Notes* on the effort



Effort

Effort: Unit Test

Type: Transition

Time: 1

Notes: Unit test this Use Case

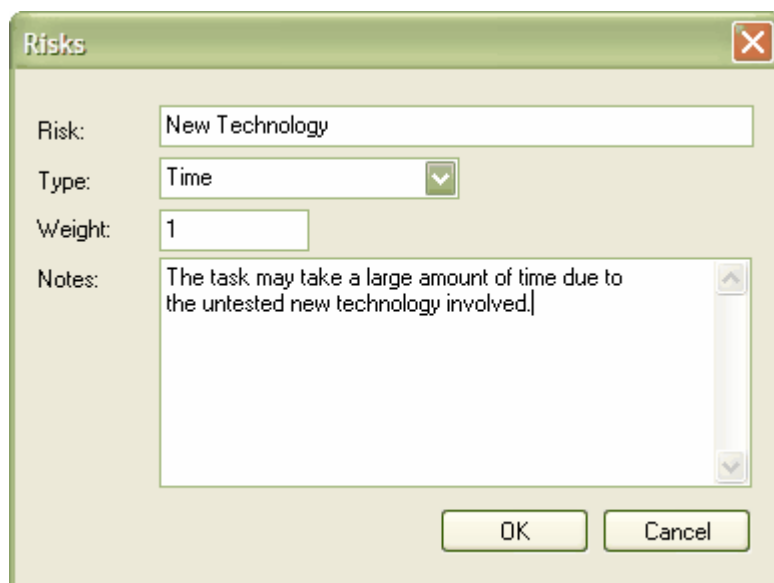
OK Cancel

8.2.3 Risk Management

To enter Risk details for an element select the element and then choose the *Add new* or *Modify selected* options by right clicking on the *Project Management* window with the tab set to *Risk*. This will present the following dialog which allows the user to enter the following data:

On the *Risk* tab you can enter the following information:

- A name for the *Risk* (short description)
- The *Type* of risk
- A *Weight* for the risk
- Some *Notes* on the risk



Risks

Risk: New Technology

Type: Time

Weight: 1

Notes: The task may take a large amount of time due to the untested new technology involved.

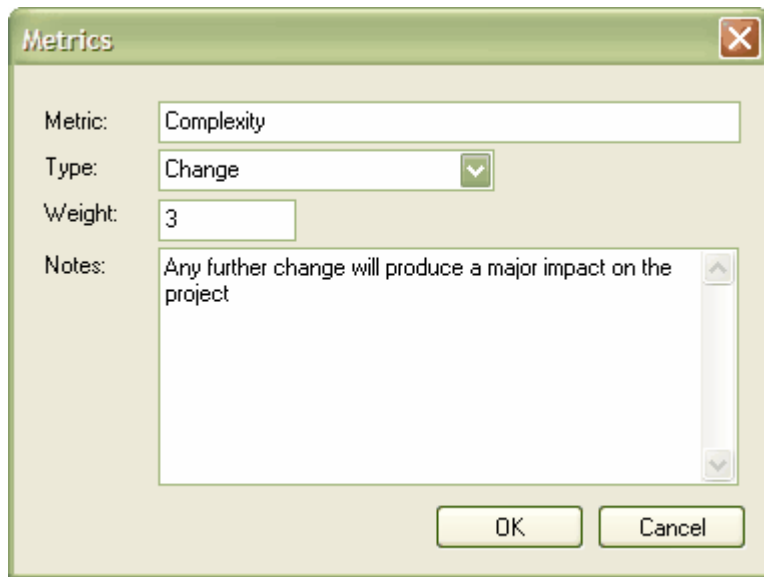
OK Cancel

8.2.4 Metrics

To enter Metrics for an element select the element and then choose the *Add new* or *Modify selected* options by right clicking on the *Project Management* window with the tab set to *Metrics*. This will present the following dialog which allows the user to enter the following data:.

On the *Metrics* tab you can enter the following information:

- A name for the *Metric* (short description)
- The *Type* of metric
- A *Weight* for the metric
- Some *Notes* on the metric



The screenshot shows a dialog box titled "Metrics". It has a close button in the top right corner. The dialog contains the following fields:

- Metric:** A text box containing the word "Complexity".
- Type:** A dropdown menu with "Change" selected.
- Weight:** A text box containing the number "3".
- Notes:** A text area containing the text "Any further change will produce a major impact on the project".

At the bottom of the dialog are two buttons: "OK" and "Cancel".

8.2.5 Resource Report

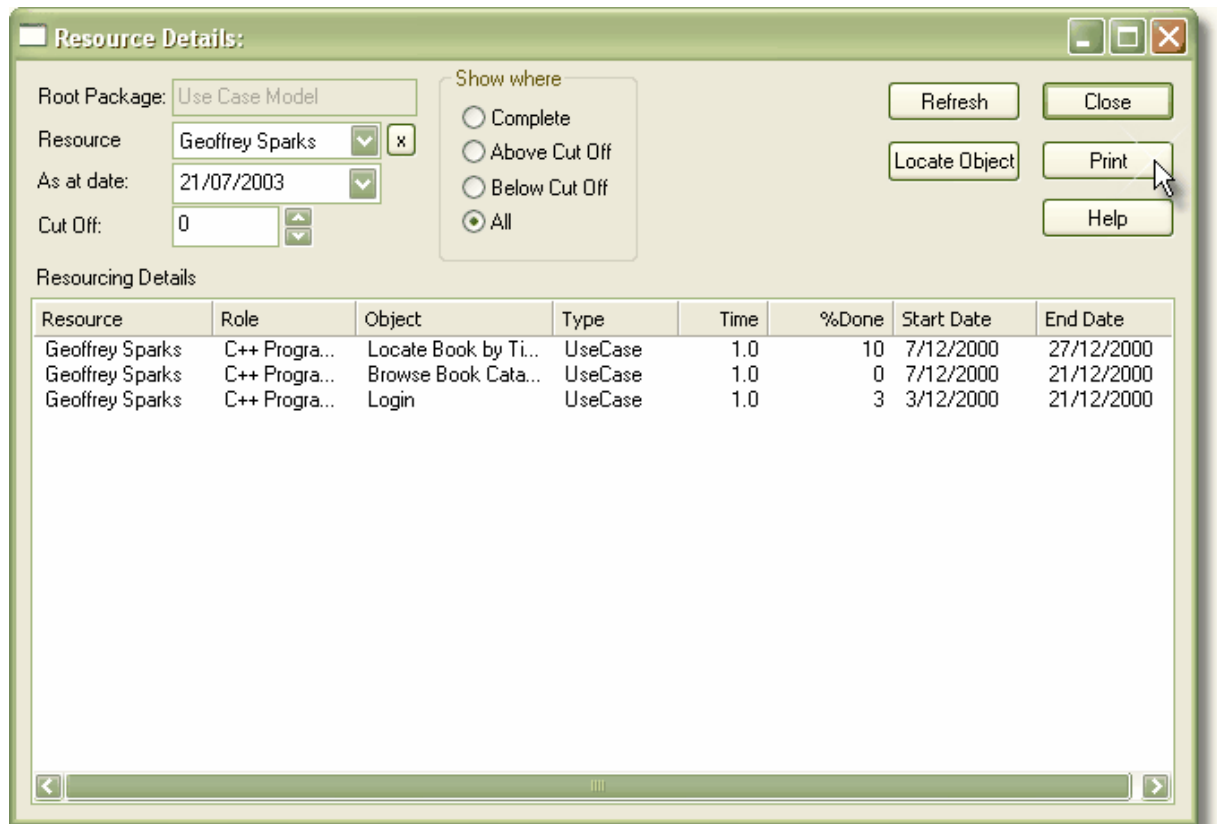
For a resource report on a package, do one of the following:

- In the Project Browser, right click on the package you require a report for. In the context menu, from the *Documentation* submenu, select *Resource Allocation*.

-OR-

- If the diagram currently active belongs to the package you require a report for, select *Resource and Tasking Details* from the *Project | Documentation* submenu.

The Resource Report dialog displays a list of all elements that have resources allocated to them. The result list includes the resource allocated, start and end dates, % complete and other relevant information. You may print out the results if required.



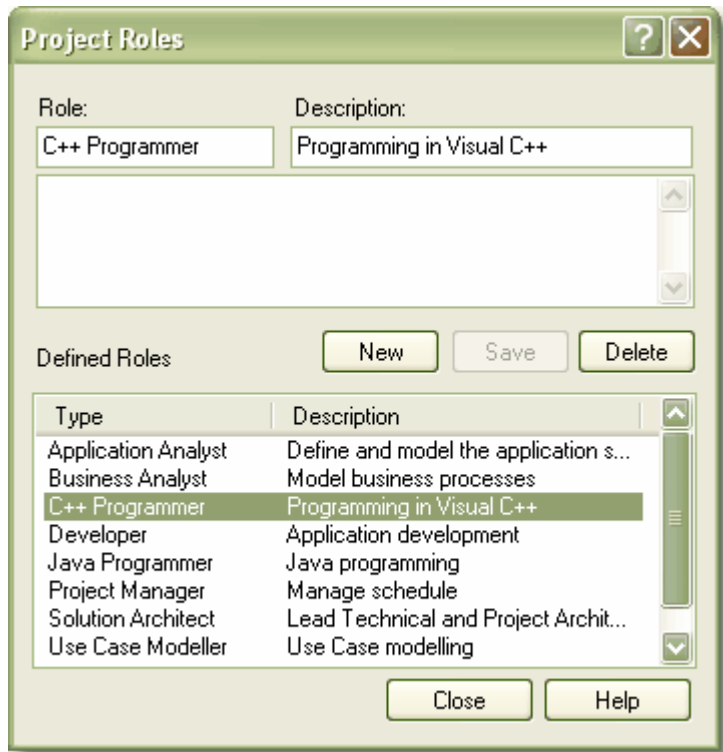
Control	Description
Root Package	The name of the root package for which resourcing is being determined.
Resource	The (optional) name of a specific resource assigned to the project.
As At Date	Date to run the resource report for.
Show Where	Show resourcing where % complete is - Complete, Above the cut-off, below the cut-off, all.
Refresh	Refresh the form.
Locate Object	Find the selected element in the results list in the Project Browser (tree).
Print	Print a report.
%Complete	Set the limit to include or exclude resource details for - see "Show Where".
Resource Details	List of resources that meet search criteria.
Cancel	Close the New Project dialog without creating a new project.

8.2.6 Roles

Project roles define the activities that resources may undertake with may be used with the Resource Management dialog in the *Project Management* window. Creating a role using this dialog creates a global list of roles which may be added to any element in the model.

To add a new role, follow the steps below:

1. From the *Configuration* | *People* submenu, select *Roles*, to open the *Project Roles* dialog.



2. Enter the *Role* title, *Description* and optional *Note*.
3. Press *Save*.

8.2.7 System Clients

Enter names and additional information for system clients using the Project Clients dialog. This information is used in such places as the maintenance screens to indicate who reported a fault.

To add a new client, follow these steps:

1. From the *Configuration* | *People* submenu, select *Clients*. This will open the *Project Clients* dialog.

Project Clients

Information on clients

Name: Organisation:

Role(s):

Phone 1: Phone 2:

Mobile: Fax:

Email:

Notes:

New Save Delete

Defined Clients

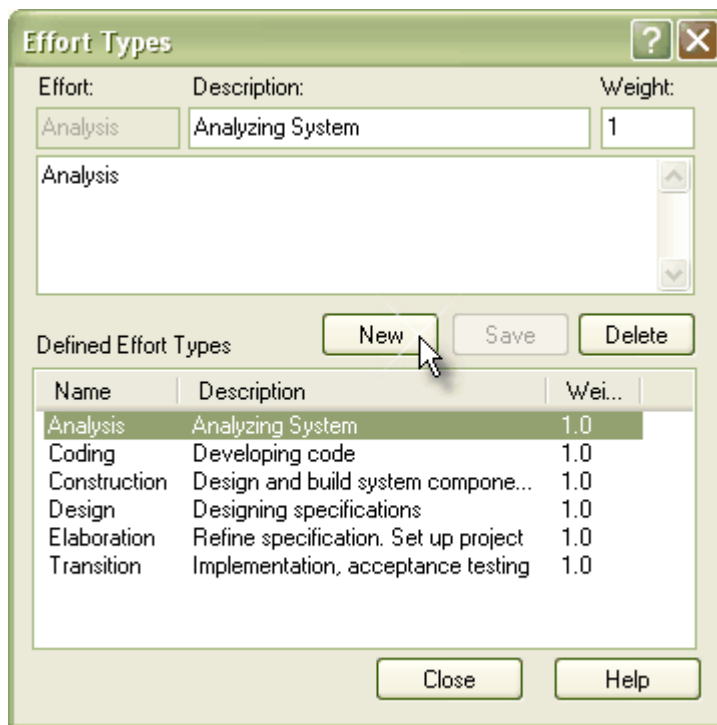
Name	Role(s)
Arthur Stoaat	Project Manager
Joanna Stoaat	Business Analyst
Margery Stoaat	Assistant PM

Close Help

2. Fill in the appropriate client details.
3. Press **Save**.

8.2.8 Effort Types

You can specify the **Effort Types** used when assigning effort to an element in EA which may be used with the Effort dialog accessed from the **Project Management** window. Creating a Effort using this dialog creates a global list of Effort types which may be added to any element in the model. From the **Configuration | Metrics & Estimation Types** submenu, select **Effort**, to open the **Effort Types** dialog.

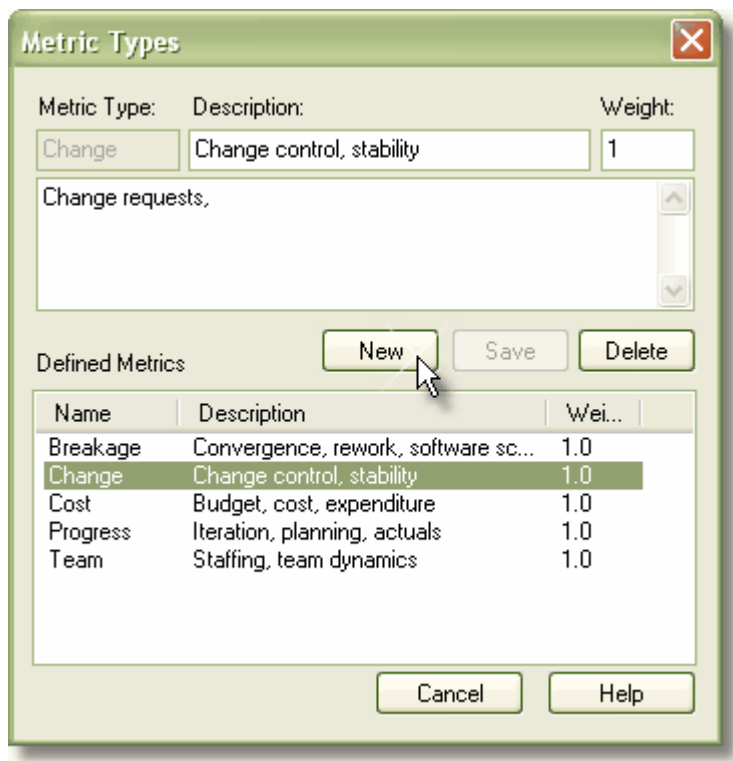


The list of types will appear in the drop list for Effort Type. The weighting is a numeric value you may assign to this type for your own metrics.

Note: Although EA does not currently provide detailed reports on 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.

8.2.9 Metric Types

You can specify the **Metric Types** used when assigning Metrics to an element in EA which may be used with the Metric dialog accessed from the **Project Management** window. Creating a Metric using this dialog creates a global list of Metric which may be added to any element in the model. From the **Configuration | Metrics & Estimation Types** submenu, select **Metrics** to open the **Metric Types** dialog.

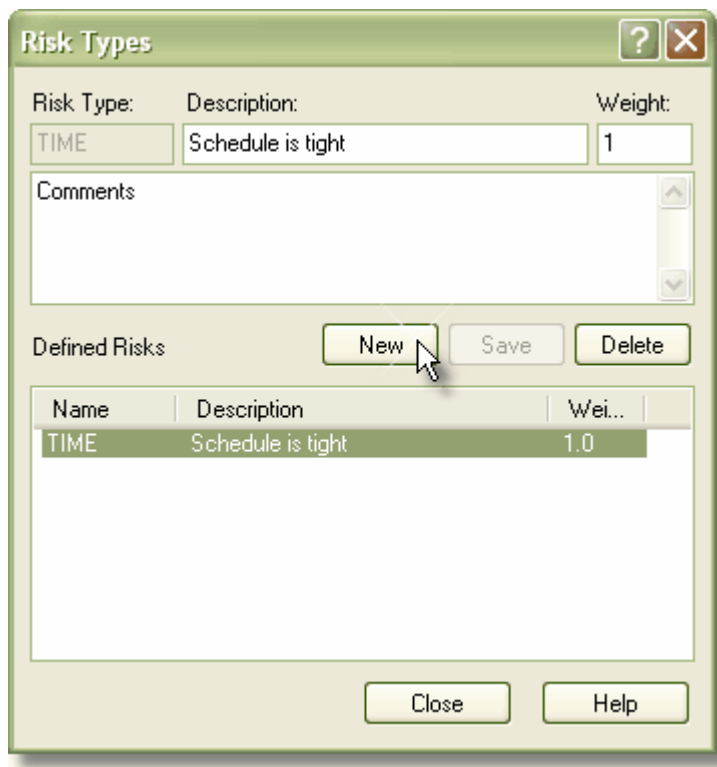


The list of types will appear in the drop list for Metric Type. The weighting is a numeric value you may assign to this type for your own metrics.

Note: Although EA does not currently provide detailed reports on Metrics within a model, you can use Automation Interface or similar tools to create your own custom reports based on Metric information you enter.

8.2.10 Risk Types

You can specify the *Risk Types* used when assigning Risk to an element in EA which may be used with the Risk dialog accessed from the *Project Management* window. Creating a risk type using this dialog creates a global list of risk types which may be added to any element in the model. From the *Configuration | Metrics & Estimation Types* submenu, select *Risks*, to open the *Risk Types* dialog.



The list of types will appear in the drop list for Risk Type. The weighting is a numeric value you may assign to this type for your own Risk.

Note: Although EA does not currently provide detailed reports on Risks within a model, you can use Automation Interface or similar tools to create your own custom reports based on Risk information you enter.

8.3 Testing

Introduction to Testing

Enterprise Architect allows you to create test scripts for model elements. Typically you would create unit tests for things that are being built, such as classes and components; integration tests to test how components work together; system tests to ensure the system meets business needs; acceptance tests to test user satisfaction; and scenario tests to test the end-to-end suitability and functionality of the application.

Basic Tasks

Simple tasks that you will want to do include:

- [Open the Testing Workspace](#)
- [Use the Test Details Dialog](#)

Categories

Tests come in the following categories:

- [Unit tests](#)
- [Integration tests](#)
- [System tests](#)
- [Acceptance tests](#)
- [Scenario tests](#)

Using Tests

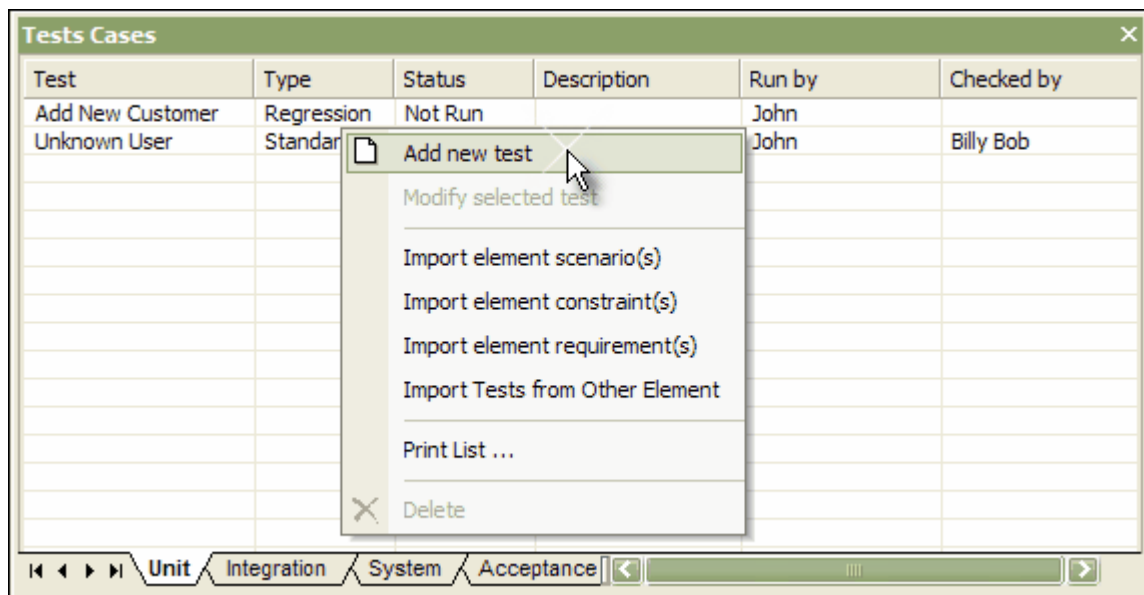
Other things that you might want to do when working with tests include:

- [Import Scenario as Test](#)
- [Import Test from other elements](#)
- [Test Details Report](#)
- [Show Test Scripts in Compartments](#)
- [Create Test Documentation](#)

8.3.1 The Testing Workspace

The **Test Cases** window provides a quick and convenient method of working with element tests. When you select an element in a diagram or in the Project Browser, if the Testing Workspace is visible, the lists of tests for that element are loaded ready for modification or addition.

To open the Testing Workspace, select **Testing** from the **View | Other Windows** submenu. Alternatively, you can use the keyboard shortcut **Alt+3**. This window may be docked to the application workspace.



Right click on the Test Cases window to access the context menu, to add or delete tests, or modify test details. Double clicking on a line item will also bring up the [Test Details dialog](#), allowing you to modify test details.

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 Test Cases window when you opt to add or modify test cases.

Tip: Add multiple test cases in one batch by using the **Apply**, **Save** and **New** options.

Test details for Class: Customer

Test Details and Execution Status

Test: Add new customer Type: Standard

Description: Add a new customer

Input: Customer name

Acceptance Criteria: Customer created

Execution

Status: Not Run Last Run Date: 7/07/2003

Run By: Checked By:

Results:

8.3.3 Unit Testing

Use Unit Testing to test classes, components and other elements as programmers build them.

The Unit Testing tab is displayed in the Test Cases window by default. Open the Test Cases window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window.

Test details for Class: Customer ✕

Test Details and Execution Status

Test: Type: ▾

Description: ▾

Input: ▾

Acceptance Criteria: ▾

Execution

Status: ▾ Last Run Date: ▾

Run By: ▾ Checked By: ▾

Results: ▾

Control	Description
Unit Test	Name of test
Type	The type of test
Status	The current status of test (passed, failed...)
Last Run	The date test last run
Description	A description of the test
Run By	Test run by (person)
Checked By	Test run checked by (person)
Input	Input data
Acceptance Criteria	Acceptance conditions
Results	Results of last test
Defined Tests	List of defined tests associated with this element

8.3.4 Integration Testing

Use Integration Testing to test how the constructed components work together.

To display the Integration Testing screen, open the Test Cases window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Integration* tab.

Test details for Class: Customer

Test Details and Execution Status

Test: Add new customer Type: Standard

Description: Add a new customer

Input: Customer name

Acceptance Criteria: Customer created

Execution

Status: Not Run Last Run Date: 7/07/2003

Run By: Checked By:

Results:

Control	Description
Integration Test	Name of test.
Type	The type of test.
Status	The current status of test (passed, failed...).
Last Run	The date test last run.
Description	A description of the test.
Run By	Test run by (person).
Checked By	Test run checked by (person).
Input	Input data.
Acceptance Criteria	Acceptance conditions.
Results	Results of last test.
Defined Tests	List of defined tests associated with this element.

8.3.5 System Testing

Use System Testing to test the system performs the right business functions correctly.

To display the System Testing screen, open the Test Cases window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *System* tab.

Test details for Class: Customer

Test Details and Execution Status

Test: Add new customer Type: Standard

Description: Add a new customer

Input: Customer name

Acceptance Criteria: Customer created

Execution

Status: Not Run Last Run Date: 7/07/2003

Run By: Checked By:

Results:

Control	Description
System Test	Name of test
Type	The type of test
Status	The current status of test (passed, failed...)
Last Run	The date test last run
Description	A description of the test
Run By	Test run by (person)
Checked By	Test run checked by (person)
Input	Input data
Acceptance Criteria	Acceptance conditions
Results	Results of last test
Defined Tests	List of defined tests associated with this element

8.3.6 Acceptance Testing

Use Acceptance Testing to ensure users are satisfied with the system.

To display the Acceptance Testing screen, open the Test Cases window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Acceptance* tab.

Test details for Class: Customer ✕

Test Details and Execution Status

Test: Type: ▾

Description: ▴ ▾

Input: ▴ ▾

Acceptance Criteria: ▴ ▾

Execution

Status: ▾ Last Run Date: ▾

Run By: ▾ Checked By: ▾

Results: ▴ ▾

Control	Description
Unit Test	Name of test
Type	The type of test
Status	The current status of test (passed, failed...)
Last Run	The date test last run
Description	A description of the test
Run By	Test run by (person)
Checked By	Test run checked by (person)
Input	Input data
Acceptance Criteria	Acceptance conditions
Results	Results of last test
Defined Tests	List of defined tests associated with this element

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 the Scenario Testing screen, open the Test Cases window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Scenario* tab.

Control	Description
Unit Test	Name of test.
Type	The type of test.
Status	The current status of test (passed, failed...).
Last Run	The date test last run.
Description	A description of the test.
Run By	Test run by (person).
Checked By	Test run checked by (person).
Results	Results of last test.
Defined Tests	List of defined tests associated with this element.

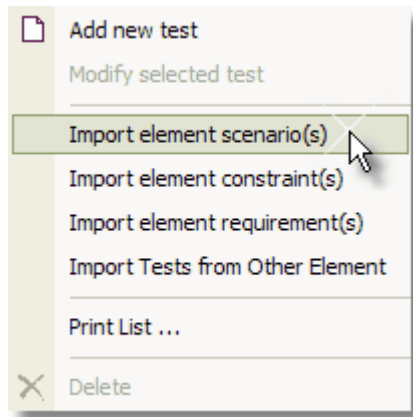
8.3.8 Import Scenario as Test

It is possible to import a Scenario from a Use Case or other element into the Test Scenarios list. This avoids having to duplicate the scenario information manually.

Import a Scenario

To import a scenario, follow the steps below:

1. Open the Test Scenario list - first open the *Test Cases* window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window. Select the *Scenario* tab.
2. Right click and select *Import Element Scenario(s)*. This will open the *Import Scenarios* dialog.



3. Select the scenarios to import in the list provided, from this dialog it is possible to import scenarios from any element in the model by using the Select Element drop down menu.



4. Press **OK** to import the selected scenario(s).

The Import Scenario Dialog has the following options:

Control	Description
Select items to import	Selects the Scenario/s to import.
Select Element	Select any element from the model.
Limit Selection	Filter out specific element types.
Refresh	Refresh available options.
Show related objects only	Filters selection to apply only to related objects.

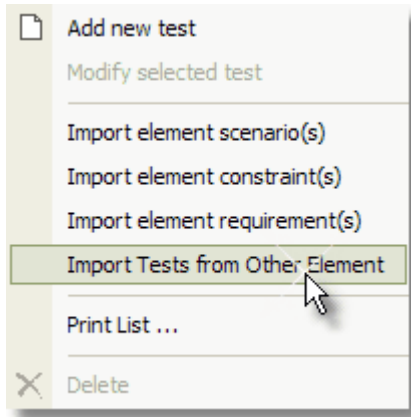
8.3.9 Import Test from other elements

It is possible to import any test from a Use Case or other element into the Test Case. This avoids having to duplicate the test information manually.

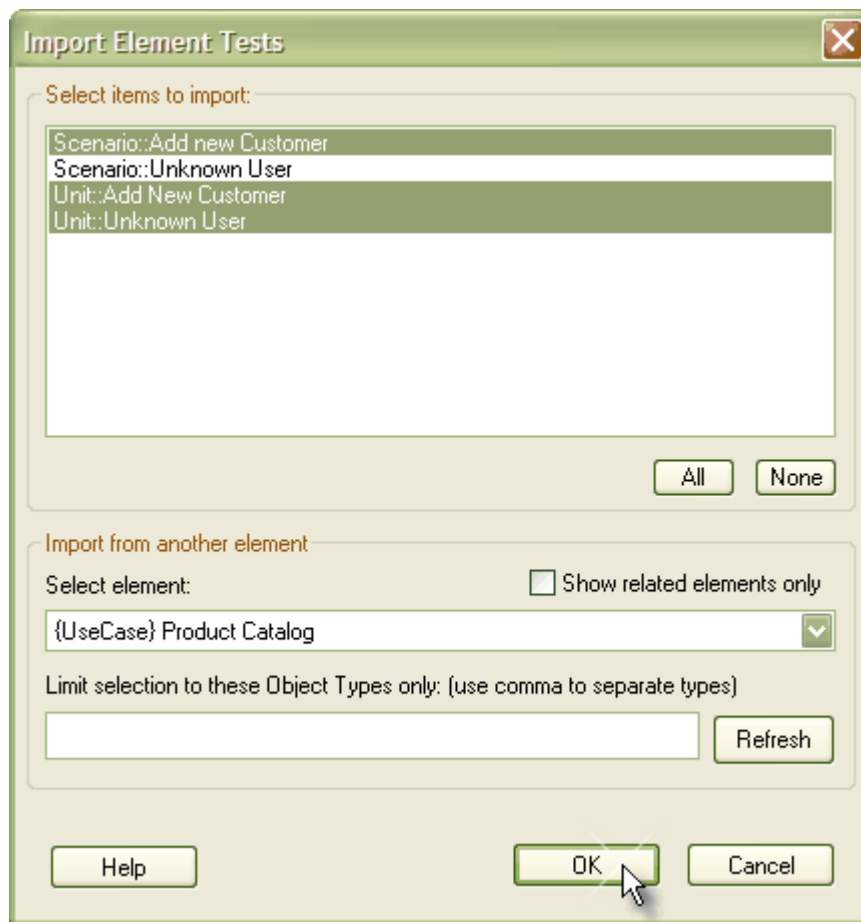
Import a Test

To import a test, follow the steps below:

1. Open the Test list - first open the *Test Cases* window by selecting *Testing* from the *View | Other Windows* submenu. Then open a diagram and select an element - all of the test scripts for that element will appear in the *Test Cases* window.
2. Right click and select *Import Tests from Other Element*. This will open the *Import Element Tests* dialog.



3. Select the scenarios to import in the list provided, from this dialog it is possible to import scenarios from any element in the model by using the Select Element drop down menu.



4. Press **OK** to import the selected test(s).

The Import Element Tests Dialog has the following options:

Control	Description
Select items to import	Selects the Scenario/s to import.
Select Element	Select any element from the model.
Limit Selection	Filter out specific element types.
Refresh	Refresh available options.
Show related objects only	Filters selection to apply only to related objects.

8.3.10 Test Details Report

You can view the **Test Details** report window for a package, which allows you to run filtered reports on all elements in the package hierarchy under your selection. You can also print the report details.

To access the **Test Details** window, select a package in the Project Browser, right click and select **Testing Details** from the **Documentation** submenu.

Test	Type	Status	Run by	Checked by	Date Run
Add new customer	Unit	Not Run			23/07/2003
Unknown User	Unit	Fail	Elosie Norman	Geoffrey Sparks	23/07/2003

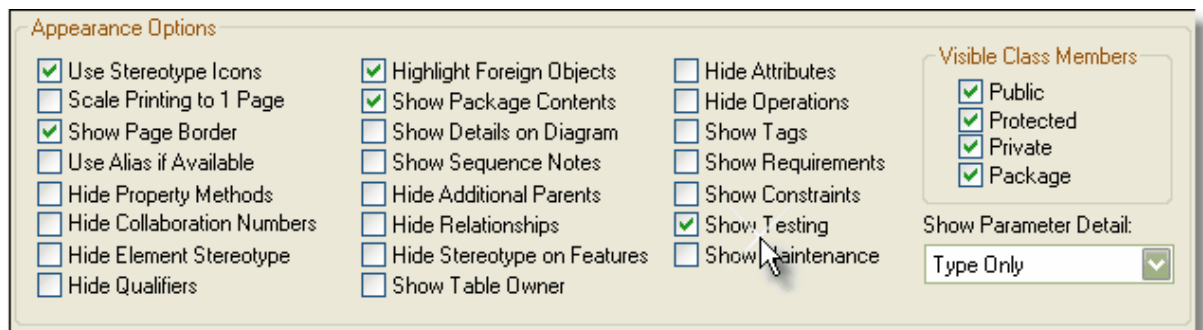
The test report includes the following fields:

Control	Description
Run By	Filter for tests run by this person.
Checked By	Filter for tests checked by this person.
Test type	Select the test type desired.
Test result	Select the result to filter for.
Locate object	Locate an element in the result list in the Project Browser.
Print	Print a summary of the test results.
Refresh	Re-run the report query.

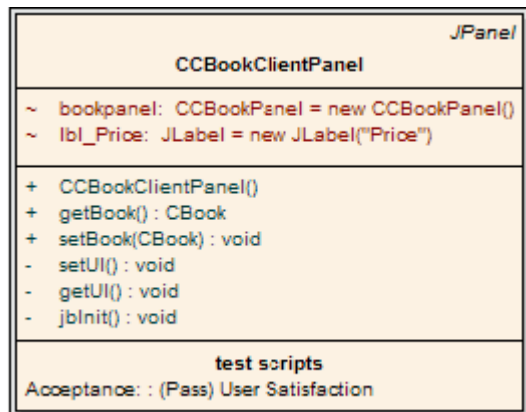
8.3.11 Show Test Scripts in Compartments

Any element that is capable of displaying a compartment can be used to show test scripts in a diagram. To make use of feature the element must have an attached test, to use this feature use the following instructions:

1. Open up a diagram with the element with the attached test item/s.
2. Double click the diagram background to bring up the Diagram Properties Dialog.



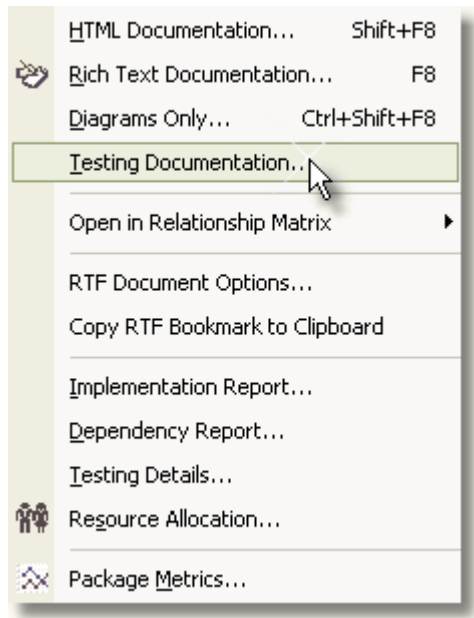
3. Check the Show Testing checkbox in the Appearance Options panel.
4. The test/s will now appear as an item in the test scripts compartment of the diagram element.



8.3.12 Test Documentation

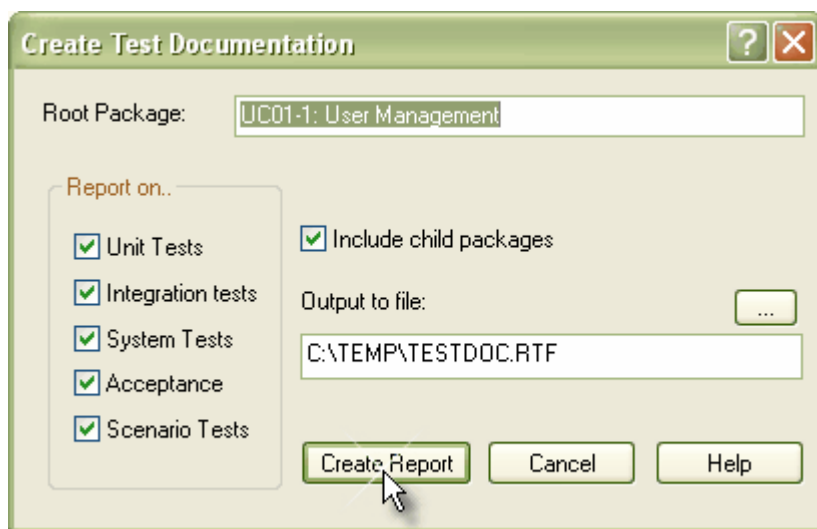
Enterprise Architect allows you to output in Rich Text format, the test scripts and results you have entered against elements in the model. For more information on entering test scripts and details see the rest of the [Testing](#) topic.

To create the documentation, access the *Create Test Documentation* dialog by right clicking on a package in the Project Browser window and selecting *Testing Documentation* from the *Documentation* submenu.



Note: You can also access the *Create Test Documentation* dialog by selecting *Testing Report* from the *Project | Documentation* submenu.

The *Create Test Documentation* dialog allows you to set up your report.



You can configure which tests to include or exclude in the report, whether to include child packages and what file to output to.

8.4 Maintenance

Maintenance Elements

The maintenance elements are defects, changes, issues and tasks. They all apply to individual model elements and may be used to record and capture problems, changes, issues and tasks as they arise and document the solution and associated details. They are defined as follows:

- A **defect** can be considered as a failure to meet a requirement for the current model element.
- A **change** can be considered as a change in requirement for the current model element.
- An **issue** is a record of a risk or other factor that might affect the project being recorded for the current model element.
- A **task** is a means of recording work in progress and work outstanding for the current model element.

Note that each of these maintenance elements applies at the model element level. For changes and issues that apply to the whole system, see the section [Changes and Defects](#); for tasks that apply to the whole system, see the section [Model Tasks](#).

More Information

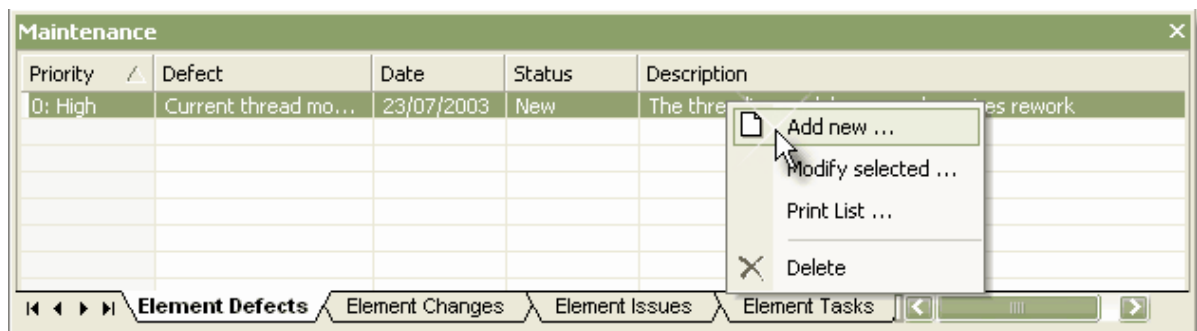
See also:

- [The Maintenance Workspace](#) shows how to create, modify, print and delete maintenance elements.
- [Show Maintenance Scripts in Compartments](#) shows how to display maintenance elements on diagrams.
- [Maintenance Element Properties](#) shows how to fill in the Properties dialog for the various maintenance elements.

8.4.1 The Maintenance Workspace

EA makes it easy to record and capture problems and issues as they arise and document the solution and associated details. The *Maintenance* window provides a quick method of viewing and modifying the list of defects, changes, issues and to do items associated with a particular model element. Access this window by selecting *Maintenance* from the *View | Other Windows* submenu, or by pressing *Alt+4*.

Four tabs provide access to *Element Defects*, *Element Changes*, *Element Issues* and *Element Tasks* - click on the tab of interest then select model elements in diagrams or in the Project Browser to see the associated maintenance items. You can include defects, changes, issues and tasks in the main RTF documentation and HTML produced by EA. The RTF setup dialog has check boxes to show or hide element defects, changes, issues and tasks.

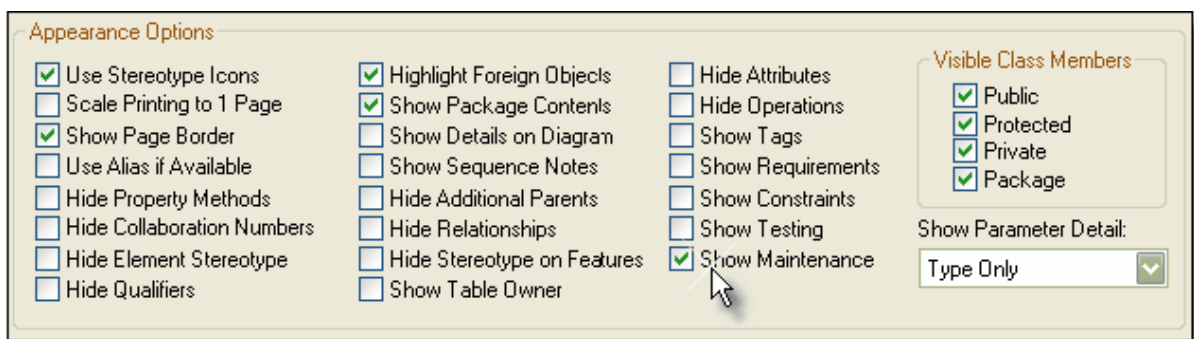


Right click on the Maintenance window to access the context menus for each tab, to add or delete items, or modify details.

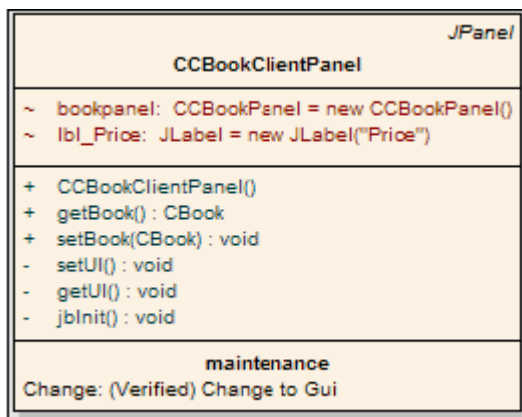
8.4.2 Show Maintenance Scripts in Compartments

Any element that is capable of displaying a compartment can be used to show maintenance scripts in a diagram. To make use of feature the element must have an attached maintenance item, to use this feature use the following instructions:

1. Open up a diagram with the element with the attached maintenance item/s.
2. Double click the diagram background to bring up the Diagram Properties Dialog.



3. Check the Show Maintenance checkbox in the Appearance Options panel.
4. The Maintenance Items will now appear as an item in the maintenance scripts compartment of the diagram element.



8.4.3 Maintenance Element Properties

Note: This page shows the properties dialog for defects. The dialogs for changes, issues and tasks differ only in minor details such as field names.

Element Defect details are recorded via the *Defect Details* dialog. Follow these steps to access this dialog:

1. Open the *Maintenance* window by selecting *Maintenance* from the *View | Other Windows* submenu.
2. Open a diagram and select an element - all of the maintenance entries for that element will appear in the *Maintenance* window, under the various tabbed sections.
3. Ensure the *Element Defects* tab is selected.
4. Double click an item in the window, or opt to add a new item (select *Add new* from the right click context menu).

Defect details for Class: IPI_ActiveObject

Details

Name:

Date: Version:

Reported by: Priority:

Description:

History

Status: Resolved by:

Date Resolved:

History:

The following fields are available:

Control	Description
Name	Name of the defect.
Date	Date of maintenance.
Status	Indicate whether complete, approved etc.
Problem	Short description.
Notes	Long description.
Reported by	Who reported the fault.
Resolved by	Who fixed the fault.
Version number	The version number associated with this fix.
Date resolved	When fixed.
Resolver comment	Notes on the fix.
Problem List	List of problems and changes associated with element.

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 Issue elements may be created in UML diagrams and linked using Realization, Dependency, Aggregation and other relationships to show what model element each affects and how each is resolved.

More Information

- Creating [Defects \(Issues\)](#)
- Creating [Changes](#)
- Change and Issue [Element Properties](#)
- [Assigning People to Defects or Changes](#)

8.5.1 Defects (Issues)

An Issue element is a structured comment which contains information about defects and issues that relate to the system/model. This corresponds in some sense to a failure to meet defined requirements for the current system. EA allows you to generate and handle Issues in much the same way as you can handle requirements (see the section on requirement tracking for more information).

You can link Issues using Realization connections to model elements that are responsible for the Defect. You can even structure a hierarchy of Issues using aggregation.

Add an Issue Using the UML Toolbox

To add an Issue to the model using the UML Toolbox (pictured right) you can either:

1. Open a Custom diagram.
2. Go to the *Custom* tab on the UML Toolbox.
3. Click on the *Issue* icon.
4. Create the issue by clicking on the diagram.
5. Enter the details as required.

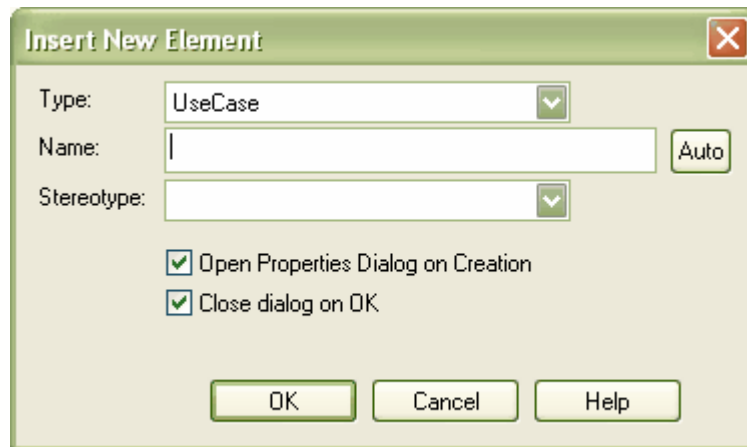
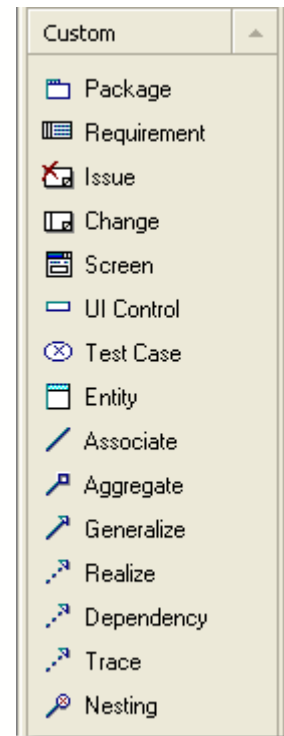
-OR-

1. Drag the *Issue* icon to a target spot on the diagram.
2. Enter details as required.

Add an Issue Using the Insert New Element Dialog

To add an Issue to the model using the *Insert New Element* dialog (pictured below) follow the steps below:

1. Right click on a Package in the Project Browser.
2. Select *New Element* from the *Insert* submenu.
3. Select the 'Issue' from the *Type* drop down list.
4. Enter a *Name* for the element.
5. Press *OK*.

**8.5.2 Changes**

A change element is a structured comment which contains information about changes that have been requested to the system/model. This corresponds in some sense to a change in requirements for the current system. EA allows you to generate and handle Changes in much the same way as you can handle requirements (see the section on requirement tracking for more information).

You can link Changes using Realization connections to model elements that will implement the Change, and you can structure a hierarchy of changes using aggregation.

Add a Change Using the UML Toolbox

To add an Change to the model using the UML Toolbox (pictured right) you can either:

1. Open a Custom diagram.
2. Go to the *Custom* tab on the UML Toolbox.
3. Click on the *Change* icon.
4. Create the change by clicking on the diagram.
5. Enter the details as required.

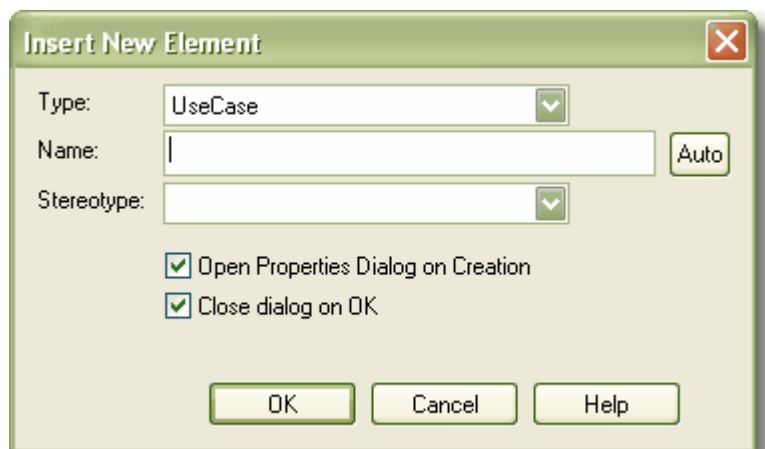
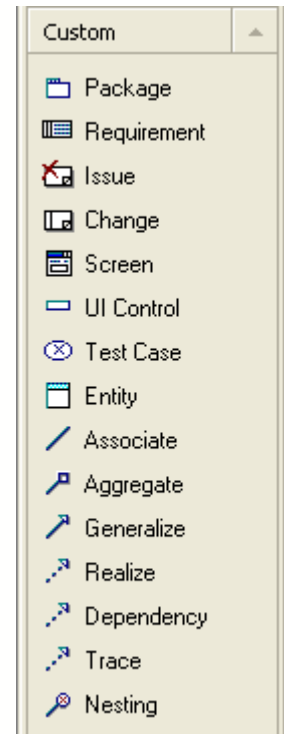
-OR-

1. Drag the *Change* icon to a target spot on the diagram.
2. Enter details as required.

Add an Issue Using the Insert New Element Dialog

To add a Change to the model using the *Insert New Element* dialog (pictured below) follow the steps below:

1. Right click on a Package in the Project Browser.
2. Select *New Element* from the *Insert* submenu.
3. Select 'Change' from the *Type* drop down list.
4. Enter a *Name* for the element.
5. Press *OK*.



8.5.3 Element Properties

The property dialog for Changes and Issues is similar to that used by *Requirements*. It has a *Properties* tab containing the name of the Issue and relevant management details (owner, dates, etc). You can also [associate files](#) with the issue and [add tagged values](#).

The screenshot shows the 'Issue' dialog box with the following details:

- Short Description: Problem with password encryption
- Status: Proposed
- Type: Functional
- Difficulty: Medium
- Phase: 1.0
- Priority: Medium
- Last Update: 1/11/2004
- Author: John Redfern
- Created: 1/11/2004
- Version: 1.0
- Details: When the user enters a password...

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 connections to Issues. Each highlighted square indicates a responsibility that a staff member has to work on or correct a named issue.

This same approach can be used for any mix of model elements.

Right click on the list to view the context menu, which allows you to add, modify and delete list items, and to set a status filter. You can also set the sort order by left clicking the title-bar of the column on which you wish to index the tasks.

For more information see the [Adding, Modifying and Deleting Tasks](#) topic.

Tip: Select the **Print List** menu option to print out the currently displayed items.

8.6.2 Adding, Modifying and Deleting Tasks

From the **Model Tasks** tab on the **System** window, use the **Task Detail** dialog to [Add](#), [Modify](#) and [Delete](#) tasks.

Add a Task

To add a task, follow the steps below:

1. Double click in the **Model Tasks** tab **-OR-** select **Add New** from the right click context menu. This will open the **Task Detail** dialog.

Task Detail

Details

Task: View user locks failed

Type: Defect Owner: Elosie Norman Start: 24/07/2003

Status: New Assigned to: End: 24/07/2003

Priority: Medium Total Time: Percent: 0

Phase: Actual Time:

Comments

History

2. Enter the details for the task - you can set the following:
 - The **Task** name
 - Press **Auto** if you have [auto counters](#) configured
 - The task **Type**

- The task *Owner*
 - The expected *Start* and *End* date for the task
 - The current *Status* of the task
 - The person this task has been *Assigned to*
 - The task *Priority* - high, medium or low
 - The expected *Total Time* for the task
 - The *Percent* complete
 - The *Phase* associated with this task
 - The *Actual Time* expended
3. Press *Save*.
 4. To create another entry, press *New*.
 5. To close, press *OK*.

Modify a Task

To modify a task, either:

1. Double click the task you want to modify in the list on the *Model Tasks* tab. This will open the *Task Detail* window for editing.

-OR-

1. Right click the entry you want to modify in the list on the *Model Tasks* tab. This will bring up the context menu.
2. Select the *Modify Selected* menu item. This will open the *Task Detail* window for editing.

Delete a Task

To delete a task, follow the steps below:

1. Right click the task you want to modify in the list on the *Model Task* tab. This will bring up the context menu.
2. Select the *Delete* menu item.

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. You can also save a report on project issues in Rich Text Format.

See:

- [Project Issues Dialog](#)
- [Model Issues Tab](#)
- [Add, Delete and Modify Issues](#)
- [Generate a Report](#)

8.7.1 Project Issues Dialog

The *Project Issues* dialog, accessed by selecting *Issues* from the *Project* menu, allows any identified issues to be recorded against the current project. Issues are raised with a description, date, owner and status. You can also save a report on project issues in Rich Text Format.

Tip: You can use the Project Issues dialog to [add, modify and delete issues](#). You can also [generate a report](#) of your issues.

Project Issues ✕

Details

Issue:

Priority: Date:

Status: Owner:

Desc:

Resolution

Resolver: Date:

Comments:

Project Issues & Discussion

Issue	Date	Owner	Status
Pre-production Environment Model...	13/05/2003	Chuck Wilson	Open
The test servers will be delayed	13/06/2003	Claire Ramsay	Under Review
Public Holidays	13/06/2003	Claire Ramsay	Open
Compiler Version disparity	13/06/2003	Claire Ramsay	Under Review
Missing Training material	13/06/2003	Claire Ramsay	Open

Show Closed Issues

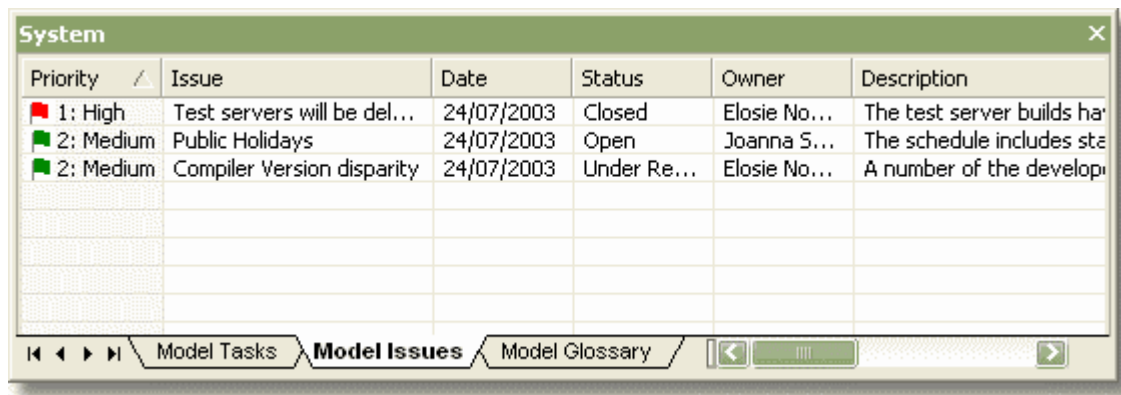
Component	Description
Issue	The name of the issue.
Auto	Press Auto if you have auto counters configured.
Priority	The priority of this issue - low, medium or high.
Date	The date the issue arose.
Status	The issue's current status.
Owner	The person owning the issue.
Description	Description of the issue.
Resolution	Notes on the resolution of the issue.
Resolver	Person who resolved the issue.
Date	The date the issue was resolved.
Comments	Any comments regarding the resolution of the issue.
Close Issue	Click to close the issue.
New	Create a new issue.
Save	Save the currently shown issue.
Delete	Delete the currently shown issue.
Issue List	List of all recorded issues.
Show Closed Issues	Include/excluded closed issues in the issue list.
View RTF	View the report generated by pressing Report.
Report	Generate a summary report of all of the recorded issues.

8.7.2 Model Issues Tab

The *Model Issues* tab in the *System* window allows any identified issues to be recorded against the current project. Issues are raised with a description, date, owner and status. You can also save a report on project issues in Rich Text Format.

Access this tab by opening the System window - select *System* from the *View | Other Windows* submenu or press **Alt+2**. Select the *Model Issues* tab.

Tip: You can right click on the list and select the *Print List* menu option to print out the currently displayed items.



You can use the *Issue Detail* dialog to [add](#), [modify](#) and [delete](#) issues. You can also [generate a report](#) of your issues.

8.7.3 Add, Delete and Modify Issues

Issues can be added, deleted and modified using either the [Project Issues dialog](#) or the [Model Issues tab](#).

8.7.3.1 Using the Project Issues Dialog

Select *Issues* from the *Project* menu to open the *Project Issues* dialog.

Add an Issue

To add an issue, follow the steps below:

1. Enter the details for the issue - you can set the following:
 - *Issue* name: the name of the issue
 - Press *Auto* if you have [auto counters](#) configured
 - Issue *Priority* - high, medium or low
 - Issue *Date*: date arose
 - Current *Status*
 - Issue *Owner*: person owning issue
 - *Description* of the issue
 - *Resolution*: Text notes on resolution
 - *Resolved By*
 - *Date* resolved
 - *Close Issue*: click to close the issue
 - *Comments* when resolving the issue
2. Press *Save*.
3. To create another entry, press *New*.
4. To close, press *OK*.

Modify an Issue

To modify an issue, follow the steps below:

1. Select the issue you wish to modify in the issues list. This will show the issue's details.
2. Modify the details as required.
3. Press *Save*.

Delete an Issue

To delete an issue, follow the steps below:

1. Select the issue you wish to delete in the issues list.
2. Press *Delete*.

8.7.3.2 Using the Model Issues Tab

Open the System window by selecting *System* from the *View | Other Windows* submenu. Alternatively, press *Alt+2*. Select the *Model Issues* tab.

Add an Issue

To add an issue, follow the steps below:

1. Double click in the *Model Issues* tab **-OR-** select *Add New* from the right click context menu. This will open the *Issue Detail* dialog.

2. Enter the details for the issue - you can set the following:
 - *Issue* name: the name of the issue
 - Press *Auto* if you have *auto counters* configured
 - Issue *Priority* - high, medium or low
 - Issue *Date*: date arose
 - Current *Status*
 - Issue *Owner*: person owning issue
 - *Description* of the issue
 - *Resolution*: Text notes on resolution
 - *Resolved By*
 - *Date* resolved
 - *Close Issue*: click to close the issue
 - *Comments* when resolving the issue
3. Press *Save*.
4. To create another entry, press *New*.

5. To close, press **OK**.

Modify an Issue

To modify an issue, either:

1. Double click the issue you want to modify in the list on the **Model Issues** tab. This will open the **Issue Detail** window for editing.

-OR-

1. Right click the entry you want to modify in the list on the **Model Issues** tab. This will bring up the context menu.
2. Select the **Modify Selected** menu item. This will open the **Issue Detail** window for editing.

Delete an Issue

To delete an issue, follow the steps below:

1. Right click the issue you want to modify in the list on the **Model Issue** tab. This will bring up the context menu.
2. Select the **Delete** menu item.

8.7.4 Generate a Report

An RTF report of your issue list can be generated and viewed using either the [Project Issues dialog](#) or the [Model Issues tab](#).

Tip: You can view sample report output in the [Report Output Sample](#) topic.

8.7.4.1 Reports - Using the Project Issues Dialog

To generate an RTF document of your issue log using the **Project Issues** dialog, follow the steps below:

1. Select **Issues** from the **Project** menu to open the **Project Issues** dialog
2. Press **Report**.
3. The **Save As** dialog will appear - select where to save your report and enter a name for it.
4. Press **Save**.
5. To view the report, press **View RTF**.

Tip: You can view sample report output in the [Report Output Sample](#) topic.

8.7.4.2 Reports - Using the Model Issues Tab

To generate an RTF document of your issue log using the **Model Issues** tab, follow the steps below:

1. Open the System window by selecting **System** from the **View | Other Windows** submenu. Alternatively,

press *Alt+2*.

2. Select the *Model Issues* tab.
3. Right click in the white area of the *Model Issues* tab and select *Create RTF Report* from the context menu.
4. Enter the file name and location to save your report to and press *Save*.
5. EA will generate your report. This should only take a few moments to complete.

Tip: You can view sample report output in the [Report Output Sample](#) topic.

8.7.4.3 Report Output Sample

An example of the output from a Issues report can be seen below:

List of Project Issues: 24-Jul-2003 9:47:00 AM

Issue	Date/Owner	Description	Resolution
Test servers will be delayed	24/07/2003 Elosie Norman	The test server builds have been delayed because the particular (unusual) memory requirements to match the customer's site are not available on shore. They are being sourced from Singapore but it will delay the builds and delivery of the machines.	Closed: 24/07/2003 Geoffrey Sparks The machines will be built and delivered using standard memory and the proprietary memory will be added later. All performance tests will be delayed until the memory is available.
Public Holidays	24/07/2003 Joanna Stoat	The schedule includes staff working on public holidays. A number of staff have indicated that contrary to what they stated earlier they will not be available.	Open: 24/07/2003
Compiler Version disparity	24/07/2003 Elosie Norman	A number of the developers have downloaded different version of a number the compilers. This has lead to unpredictable builds impacting on testing.	Under Review: 24/07/2003

8.8 Model Glossary

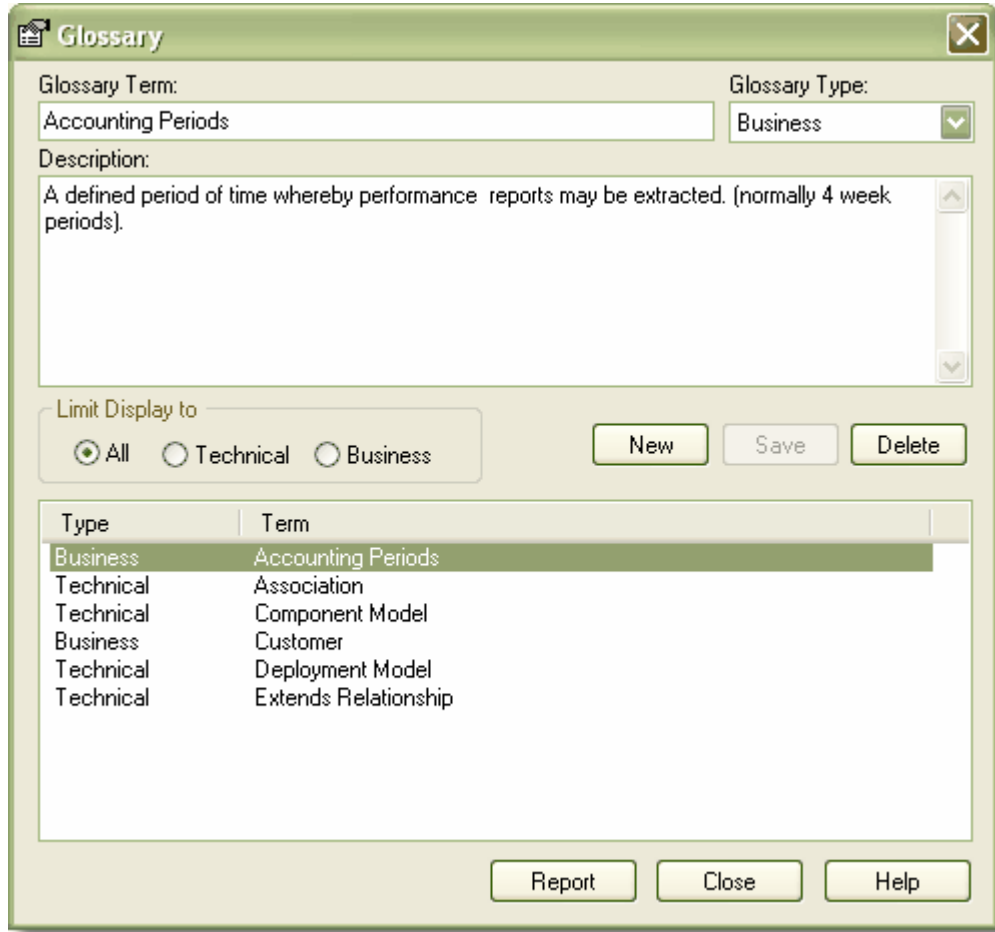
The glossary allows you to set up a list of defined terms for your project. You may further divide the items by category - for example, Business terms, Technical terms. The glossary can be saved in Rich Text format for inclusion as part of a larger project document.

You can access the Model Glossary through the [Glossary Dialog](#) or through the [Model Glossary tab](#) on the **System** window.

Tip: Include a [Glossary Report](#) in your project requirements or functional specifications document(s).

8.8.1 The Glossary Dialog

To open the *Glossary* dialog, select Glossary from the *Project* menu. Use this dialog to [add](#), [modify](#) and [delete](#) glossary entries. You may also [limit the display](#) to show only technical or business related entries.



Control	Description
Term	A term in the glossary.
Term type	Technical or business.
Description	Notes to describe the term.
Limit Display To	Filter glossary to Technical or Functional terms or both.
Report	Print a glossary report.
Defined Terms	List of defined glossary terms.

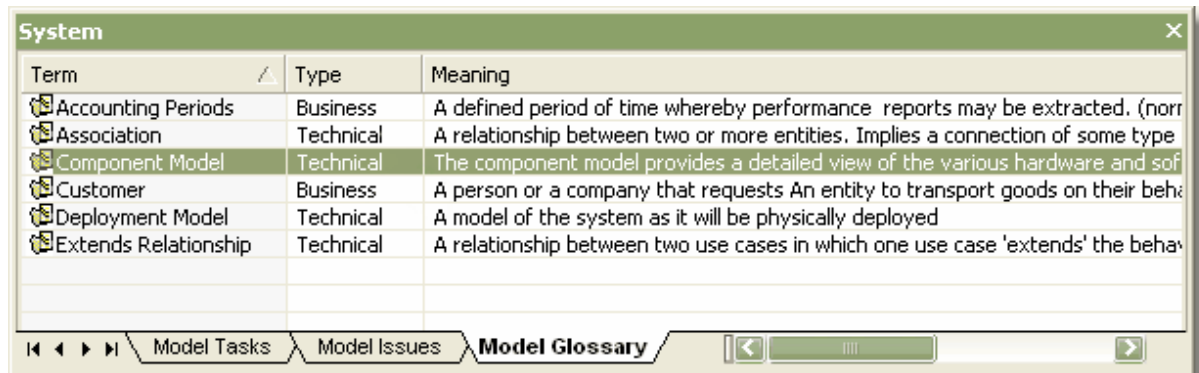
8.8.2 Model Glossary Tab

The *Model Glossary* tab in the *System* window shows all of the items in your model's glossary. This tab lists all the defined technical and business terms already defined for a model. You can add to the list, delete or

change items and filter the list to exclude by type.

Access this tab by opening the System window - select **System** from the **View | Other Windows** submenu or press **Alt+2**. Select the **Model Glossary** tab.

Tip: You can right click on the list and select the **Print List** menu option to print out the currently displayed items.



Term	Type	Meaning
Accounting Periods	Business	A defined period of time whereby performance reports may be extracted. (norm
Association	Technical	A relationship between two or more entities. Implies a connection of some type
Component Model	Technical	The component model provides a detailed view of the various hardware and sof
Customer	Business	A person or a company that requests An entity to transport goods on their beha
Deployment Model	Technical	A model of the system as it will be physically deployed
Extends Relationship	Technical	A relationship between two use cases in which one use case 'extends' the beha

You can use the **Glossary Detail** dialog to **add**, **modify** and **delete** glossary entries. This can also be done from the **Glossary dialog**.

Tip: Include a **Glossary Report** in your project requirements or functional specifications document(s).

8.8.3 Add, Delete and Modify Glossary Entries

Glossary entries can be added, deleted and modified using either the **Glossary dialog** or the **Model Glossary tab**.

8.8.3.1 Using the Glossary Dialog

Select **Glossary** from the **Project** menu to open the **Glossary** dialog.

Add a Glossary Entry

To add an entry to the glossary, follow the steps below:

1. Enter the details for the glossary item - the **Term**, the **Type** and the **Meaning**.
2. Click **Save**.
3. To enter another item, click **New**.

Modify a Glossary Entry

To modify a glossary entry, follow the steps below:

1. Select the entry you wish to modify. Its details will appear in the fields in the top half of the window.
2. Change the details as required.
3. Press **Save**.

Delete a Glossary Entry

To delete a glossary entry, follow the steps below:

1. Select the entry you wish to modify. Its details will appear in the fields in the top half of the window.
2. Press *Delete*.

Limit the Display

You can alter which entry categories are displayed in the list. You can choose from:

- View all glossary entries - select the *All* option
- View Technical categorized entries only - select the *Technical* option
- View Business categorized entries only - select the *Business* option

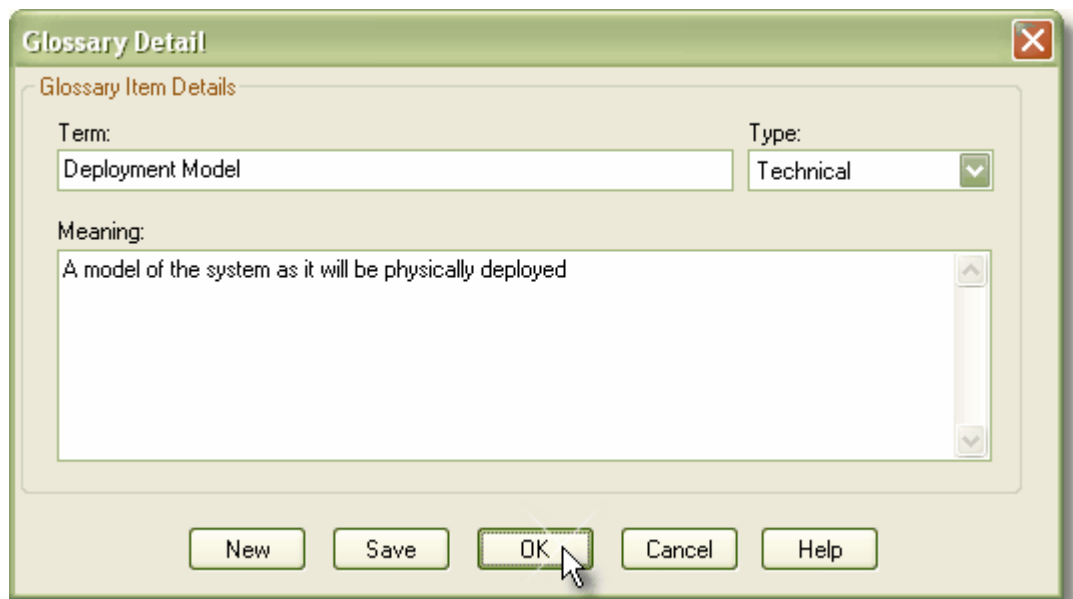
8.8.3.2 Using the Model Glossary Tab

Open the System window by selecting *System* from the *View | Other Windows* submenu. Alternatively, press *Alt+2*. Select the *Model Glossary* tab.

Add a Glossary Entry

To add an entry to the glossary, follow the steps below:

1. Double click in the *Model Glossary* tab **-OR-** select *Add New* from the right click context menu. This will open the *Glossary Detail* dialog.



2. Enter the details for the glossary item - the *Term*, the *Type* and the *Meaning*.
3. Press *Save*.
4. To create another entry, press *New*.
5. To close, press *OK*.

Modify a Glossary Entry

To modify a glossary entry, either:

1. Double click the entry you want to modify in the list on the *Model Glossary* tab. This will open the *Glossary Detail* window for editing.

-OR-

1. Right click the entry you want to modify in the list on the *Model Glossary* tab. This will bring up the context menu.
2. Select the *Modify Selected* menu item. This will open the *Glossary Detail* window for editing.

Delete a Glossary Entry

To delete a glossary entry, follow the steps below:

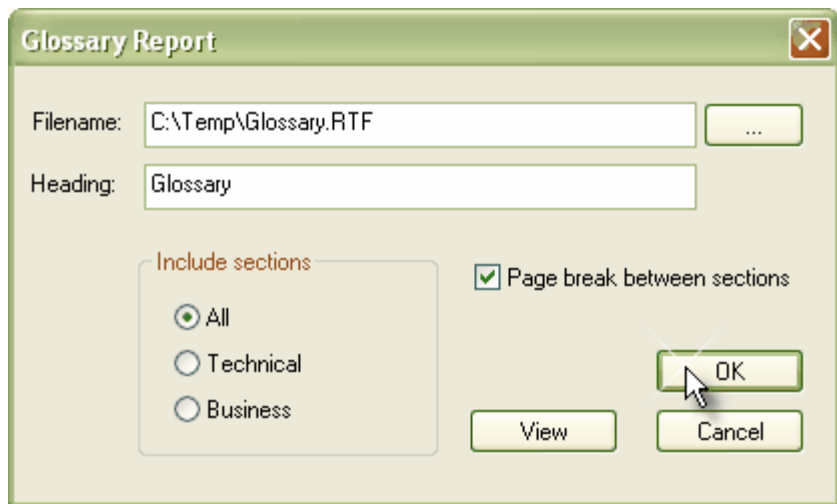
1. Right click the entry you want to modify in the list on the *Model Glossary* tab. This will bring up the context menu.
2. Select the *Delete* menu item.

8.8.4 Generate a Report

Generate a Report

You can generate a report of your model's glossary by following these steps:

1. Select *Glossary* from the *Project* menu to open the *Glossary* dialog.
2. Press *Report*. This will open the *Glossary Report* dialog.



3. Enter a filename and a heading for the glossary.
4. Select whether you want to include all sections, or only the Technical or Business sections.
5. If you want to include page breaks, check the *Page break between sections* option.
6. Press *OK* to generate the report.

7. Press **View** to open the report.

Tip: You can view sample report output in the [Glossary Report Output Sample](#) topic.

8.8.5 Glossary Report Output Sample

An example of the output from a Glossary report can be seen below:

Glossary

Business Terms

Accounting Periods

A defined period of time whereby performance reports may be extracted. (normally 4 week periods).

Customer

A person or a company that requests An entity to transport goods on their behalf.

Technical Terms

Association

A relationship between two or more entities. Implies a connection of some type - for example one entity uses the services of another, or one entity is connected to another over a network link.

Component Model

The component model provides a detailed view of the various hardware and software components that make up the proposed system. It shows both where these components reside and how they inter-relate with other components. Component requirements detail what responsibilities a component has to supply functionality or behavior within the system.

Deployment Model

A model of the system as it will be physically deployed

Extends Relationship

A relationship between two use cases in which one use case 'extends' the behavior of another. Typically this represents optional behavior in a use case scenario - for example a user may optionally request a list or report at some point in a performing a business use case.

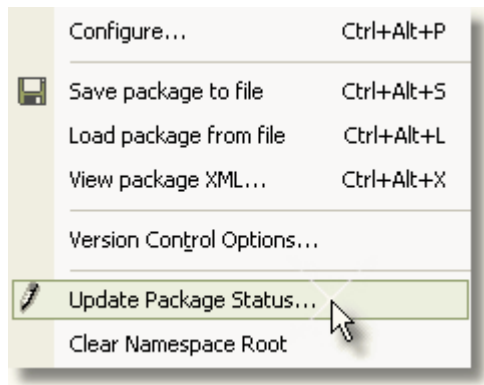
8.9 Update Package Status

Elements in EA may be assigned a current status, such as Proposed, Validated, Mandatory etc. Often a complete package structure will be updated from one status to another at the same time (or released). To help facilitate this, EA supports a 'bulk' update of element status at the same time

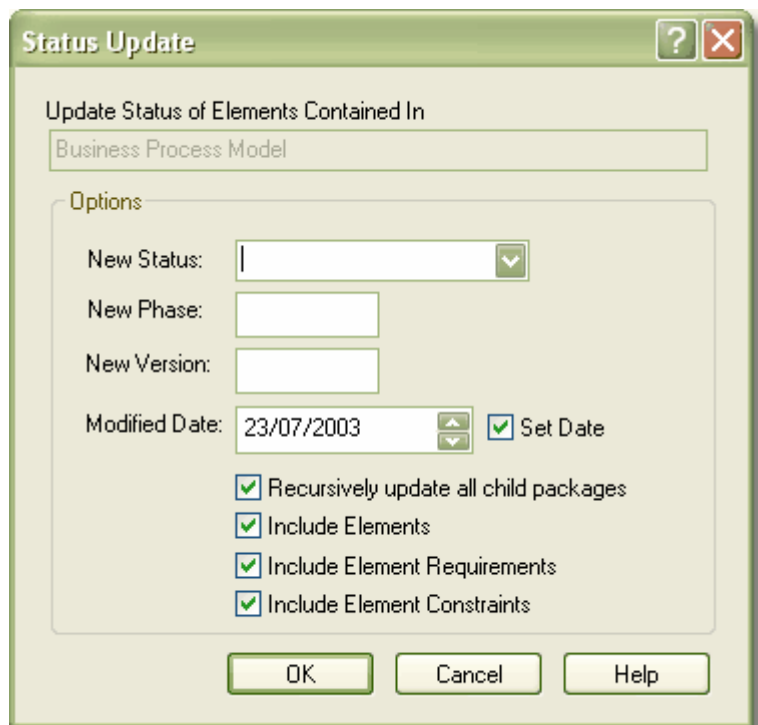
Update Element Status for a Complete Package Structure

To update element status for a complete package structure, follow the steps below:

1. In the Project Browser, right click on the package you need to update, to open the context menu.
2. Select **Update Package Status** from the **Package Control** submenu. This will open the **Status Update** dialog.



3. In the *Status Update* update dialog select:
 - The new status
 - Whether to recursively descend the package tree
 - Whether to include elements
 - Whether to include element requirements
 - Whether to include element constraints



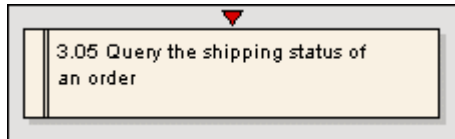
4. Press *OK* when you have configured the settings. EA will update all required elements to the new status.

8.10 Manage Bookmarks

Bookmarks are small red triangles that appear above elements in diagrams if the element has been 'bookmarked'. A Bookmark is a visual clue that something is different about an element - the meaning is up to you.

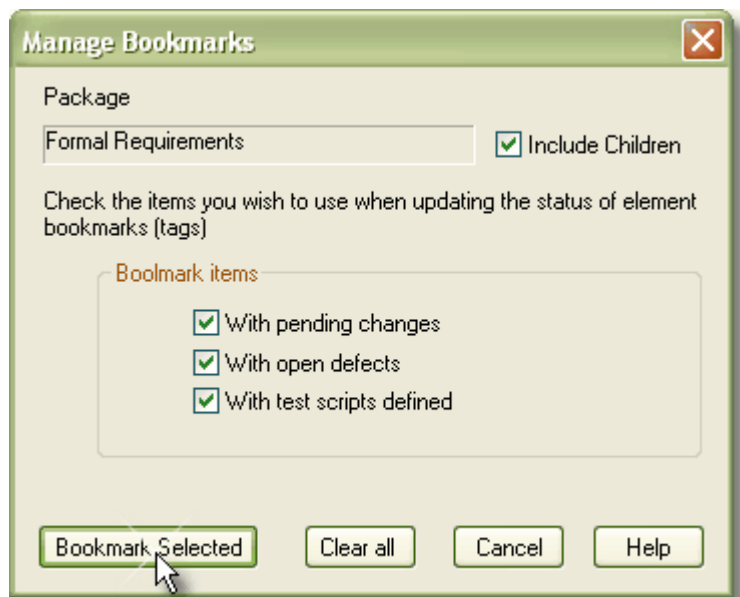
Tip: The [Search](#) dialog also allows searching based on bookmarked elements.

You can bookmark an element manually by pressing the **Alt+Spacebar** keys:



Bookmark Multiple Elements

You can also bookmark all elements in a folder (and their children) using the [Manage Bookmarks](#) dialog. This is accessible from the package context menu in the Project Browser - select the [Bookmarks](#) menu item.



This dialog lets you automatically bookmark elements with changes, with defects or with test scripts defined. This is useful to highlight elements that have additional project information, or alternatively, those elements that do not.

You can press **Clear All** to clear all elements in the current tree of bookmarks.

Part



9 Code Engineering

Code Engineering is a process which includes the processes of code generation, reverse engineering of source code and synchronization between the source code and model. Code Engineering is only available in the Professional and Corporate Editions of Enterprise Architect, Desktop edition owners can upgrade to either of the aforementioned editions from the [EA Upgrade.htm](#) page.

Enterprise Architect allows you to generate source code from UML models. In particular you may generate C++, C#, Delphi, Java, PHP, Visual Basic and VB.NET source code. The source code generated includes class definitions, variables and function stubs for each attribute and method in the UML class.

Generating code is also known as Forward Engineering. Importing source code into model elements is known as Reverse Engineering the languages that are available for generation with EA are also available for reverse engineering.

Synchronization is when changes in the model are exported to the source and changes to source are imported into the model. This allows you to keep your model and your source code up to date as the project develops.

Round trip engineering occurs as a combination of reverse and forward generation of code and should include synchronization between the source code and the model except in all but the most trivial of code engineering projects.

See:

- [Reverse Engineer Source Code](#)
- [Generate Source Code](#)
- [Code Engineering Settings](#)
- [Language Options](#)
- [Code Template Framework](#)
- [XML Schema Generation](#)

9.1 Reverse Engineer Source Code

Reverse Engineering in EA allows the user to import existing source code from a variety of code languages into a UML model. Existing source code structures will be mapped into their UML representations, for example a Java class will be mapped into a UML class element with the variables being defined as attributes, methods are modeled as operations and the interactions between the Java classes being displayed in the UML model class diagram with the appropriate connections.

Reverse Engineering allows users to examine legacy code and examine the functionality of code libraries for reuse or to bring the UML model up to date with the code that has been developed as part of a process called synchronization. Examining the code in a UML model allows user to identify the critical modules contained the code, allowing a starting point for understanding of the business and system requirements of the pre-existing system and to enable the developers to gain a better overall understanding of the source code.

To begin the process of importing existing code into EA an existing source of code needs to be [imported into EA](#), which may be a single directory or [directory structure](#). Several options are available when performing the reverse engineering process, these include specifying the default directory of the source code, choosing whether to include comments from the source into the model and when using C++ the imported file extension of the C++ file.

It is important to note that when a legacy system has been poorly designed that simply importing the source into EA will not create an easily understood UML model. When working with legacy system which is poorly designed it is useful to break down the code into manageable components by examining the code elements on a "per element" basis. This can be achieved by importing a specific class of interest into a diagram and then [inserting the related elements](#) at one level to determine immediate relationship to other classes. From this point it is possible to create use cases that identify the interaction between the legacy classes enabling an overview of the legacy systems operation.

Copyright ownership is an important issue to take into account when undertaking the process of reverse engineering. In some cases, software will have specific limitations which prohibit the process of reverse

engineering, it is important that a user address the issue of copyright before beginning the process of reverse engineering of code. Situations which typically lend themselves to the reverse engineering of source code include source code which is:

- Source code which you have already developed
- That is part of a 3rd party library which you have obtained permission to use.
- Part of a framework which your organization uses.
- That is being developed on a daily basis by your developers.

Enterprise Architect currently supports reverse engineering in the following programming languages

- [Java](#)
- [C++](#)
- [C#](#)
- [Delphi](#)
- [VB.NET](#)
- [PHP](#)
- [Visual Basic](#)

Note: Reverse Engineering of other languages including CORBA IDL and Python is currently available through the use of MDG Technologies from http://www.sparxsystems.com.au/mdg_technologies.htm.

9.1.1 Import Source Code

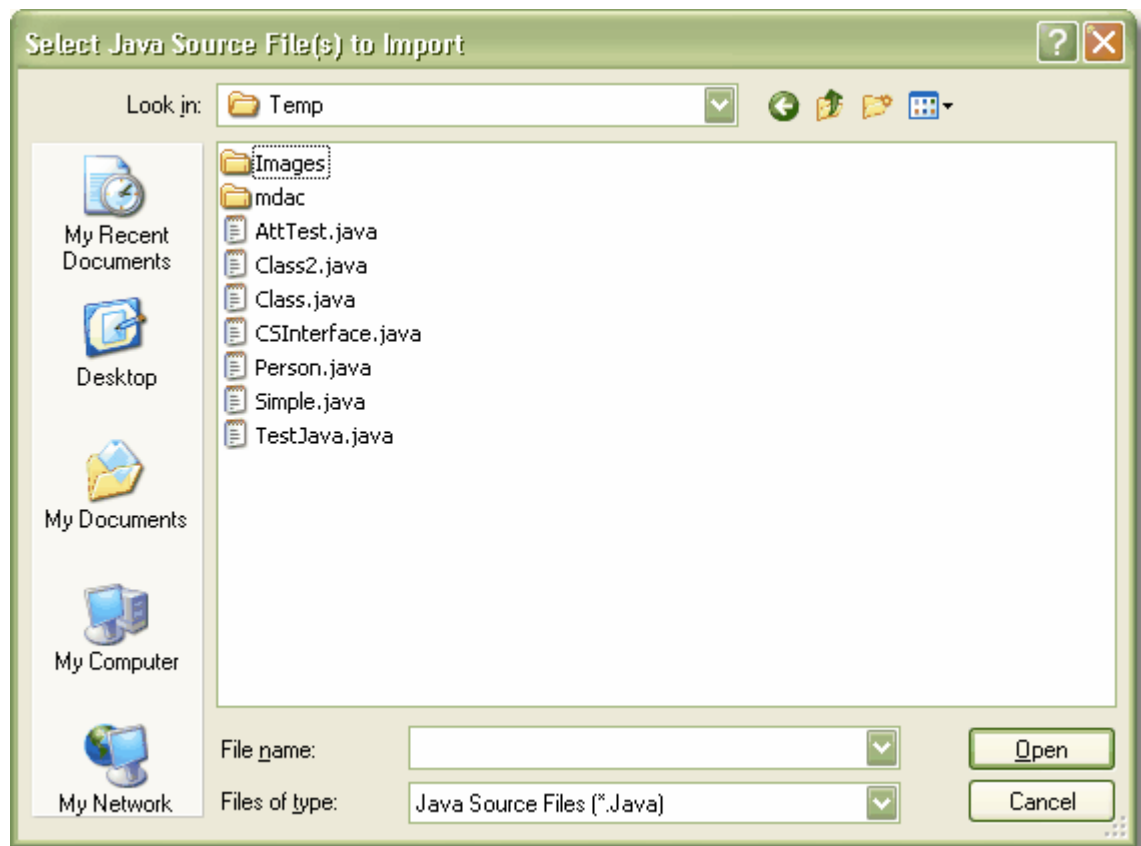
To import source code (reverse engineer) you will usually do the following:

1. In the Project Browser, select (or add) a diagram into which the classes will be imported.
2. Right click on the diagram background to open the context menu. Select the language you wish to import from the *Import from source file(s)* submenu.

-OR-

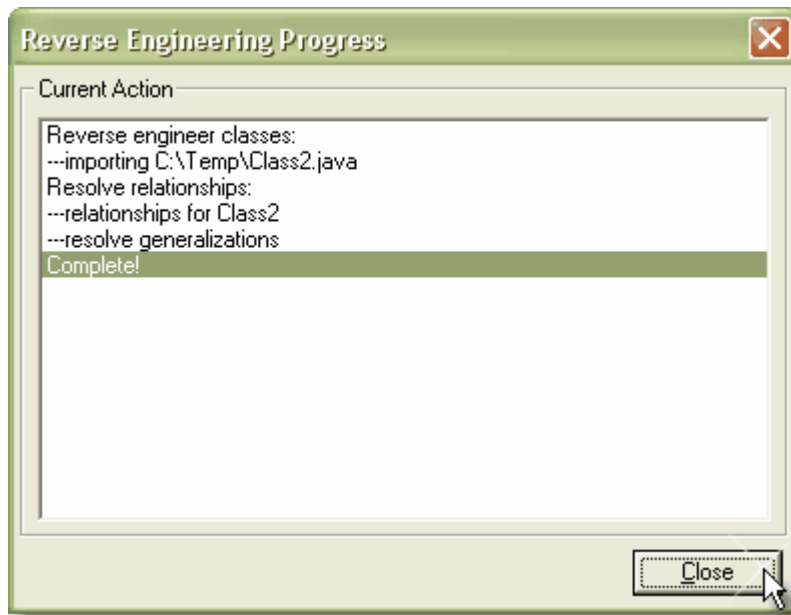
Use the drop down *Import Language* list in the *Code Generation* toolbar - select the *Import | Import xxx files* item, where *xxx* represents the language you wish to import

3. From the file browser that appears, select one or more source code files to import.



4. Press **OK** to start the import process.

As the import proceeds, EA will provide progress information. When all files are imported, EA makes a second pass to resolve and associations and inheritance relationships between the imported classes.

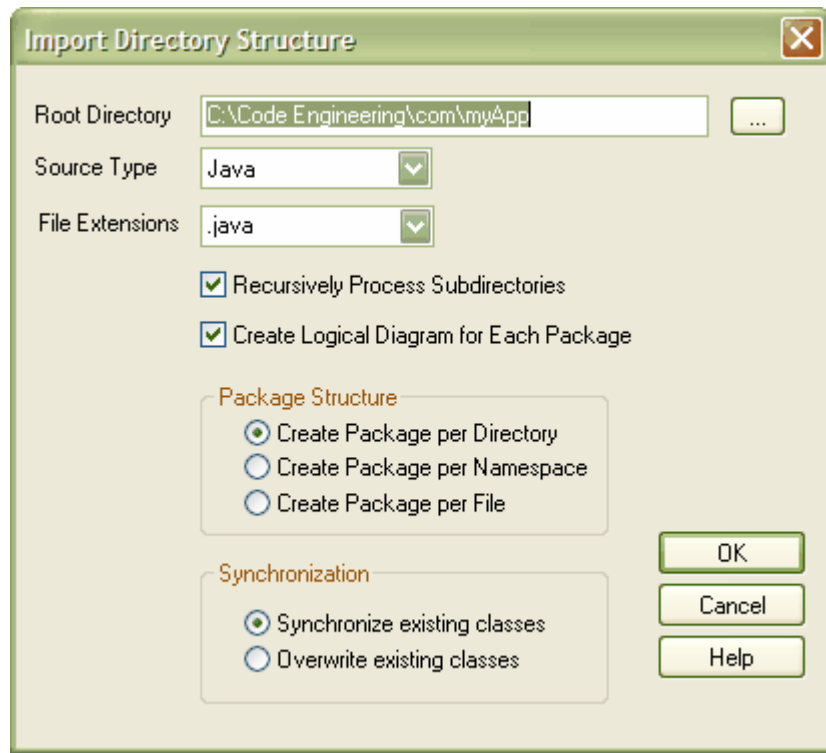


9.1.2 Import a Directory Structure

You may elect to import from all source files in a complete directory structure. This process allows you to import or synchronize multiple files in a directory tree in one pass. Enterprise Architect will create the necessary packages and diagrams during the import process.

To import a directory structure, follow the steps below:

1. In the Project Browser, right click on a package in the Project View.
2. From the context menu, select *Import Source Directory* from the *Code Engineering* submenu.
3. In the *Import Directory Structure* dialog, select the options you require. You may configure:
 - The source directory
 - The source type
 - The file extensions to look at
 - Whether to recurse sub directories
 - Whether to create a diagram for each package
 - To create a package for every directory, namespace or file. This may be restricted dependant on the source type selected.
 - Whether to Synchronize or Overwrite existing classes when found
4. Press *OK* to start



9.1.3 Import C++

C++ code may be imported into Enterprise Architect. When you import C++, you will need to select the appropriate header file (.h) file as the source code to import. EA does not use information in the .cpp file.

EA supports most C++ constructs and keywords; however it will not support some extensions and macros used by Visual Studio or Borland. Future releases of EA will support more of these non-standard extensions.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#)

9.1.4 Import C#

C# code may be imported into Enterprise Architect. When you import C#, you will need to select the appropriate source file (.cs) as the source code to import. EA supports most C# constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#)

9.1.5 Import Delphi

Delphi code may be imported into Enterprise Architect. When you import Delphi, you will need to select the appropriate source file (.pas) as the source code to import.

EA supports most Delphi constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#)

9.1.6 Import Java

Java code may be imported into Enterprise Architect. When you import Java, you will need to select the appropriate source file (.java) as the source code to import.

EA supports most Java constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#)

9.1.7 Import PHP

PHP code can now be imported into Enterprise Architect. When you import PHP, you will need to select the appropriate source file (.php, .php4, .inc) as the source code to import.

EA supports most PHP constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#).

9.1.8 Import Visual Basic

Visual Basic code may be imported into Enterprise Architect. When you import Visual Basic, you will need to select the appropriate class file (.cls) file as the source code to import.

EA supports most Visual Basic constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#)

9.1.9 Import VB.Net

VB.Net code may be imported into Enterprise Architect. When you import VB.Net, you will need to select the appropriate class file (.vb) file as the source code to import.

EA supports most VB.Net constructs and keywords.

If there is a particular feature you need support for that you feel is missing, please contact [Sparx Systems](#).

For details on how to import source code see: [Import Source Code](#)

9.2 Generate Source Code

Generating source code (forward engineering) takes the UML class or interface model elements and creates a source code equivalent for future elaboration and compilation. By forward engineering code from the model the mundane work involved with having to key in classes and attributes and methods is removed, and symmetry between model and code is ensured.

Code is generated from Class or Interface model elements, so you must create the required class and interface elements to generate from. Add attributes (which become variables) and operations (which become methods).

Before you generate code, you should ensure the default settings for code generation match your requirements. The default generation settings are located in the *Generation* section of the *Local Options* dialog (accessed by selecting *Options* from the *Tools* menu). Set up the defaults to match your required language and preferences. Preferences that may be defined include default constructors and destructors, methods for interfaces and the Unicode options for created languages. In addition to the default settings for generating code EA supports the following code languages with their own specific code generation options:

- [C++](#)
- [C#](#)
- [Delphi](#)
- [Java](#)
- [Visual Basic](#)
- [Visual Basic .NET](#)

The Code Template Framework (CTF) allows for the customization of the way EA generates source code and also allows for the generation of languages that are not specifically supported by EA.

When you have completed the design of your classes, you may generate source code.

See also

- [How to Generate Code](#)
- [The Code Generation Dialog](#)
- [Generate a Group of Classes](#)
- [Generate a Package](#)
- [Generate Package Dialog](#)
- [Update Package Contents](#)
- [Namespaces](#)

9.2.1 How to Generate Code

Before you generate code, you should ensure the default settings for code generation match your requirements. The default generation settings are located in the *Generation* section of the *Local Options* dialog (accessed by selecting *Options* from the *Tools* menu). Set up the defaults to match your required language and preferences. Languages such as Java support namespaces and may be configured to specify a namespace root.

Code is generated from Class or Interface model elements, so you must create the required class and interface elements to generate from. Add attributes (which become variables) and operations (which become methods). When you have completed the design of your classes, you may generate source code.

Before generating code, you should also familiarize yourself with the way Enterprise Architect handles local path names. Local path names allow you to substitute tags for directory names (eg. %SRC% = C:\Source).

See also:

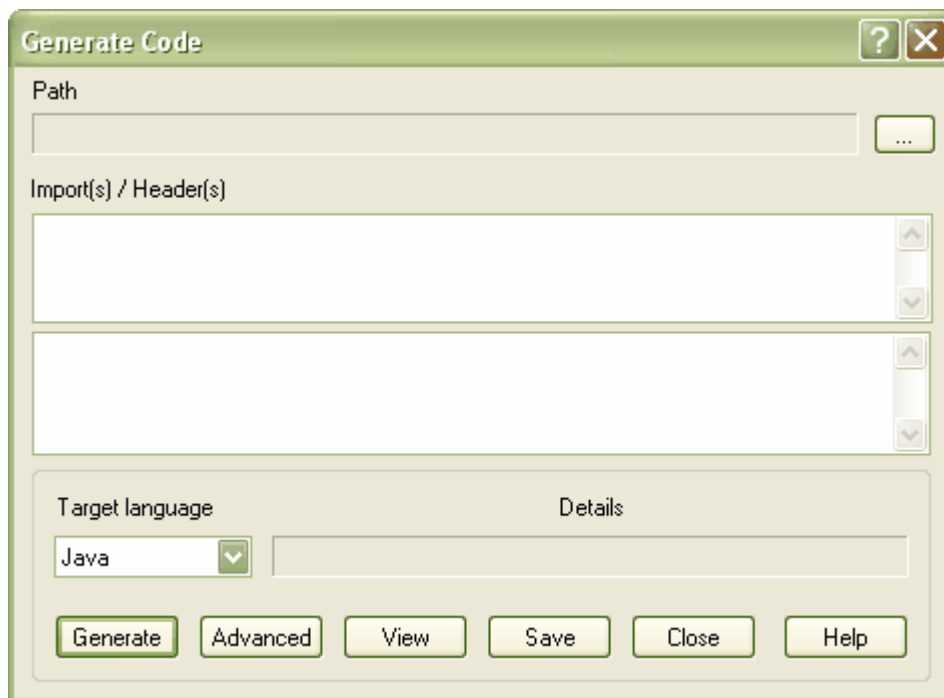
- [Generate a Single Class](#)
- [Generate a group of Classes](#)
- [Generate a Package](#)

9.2.2 Generate a Single Class

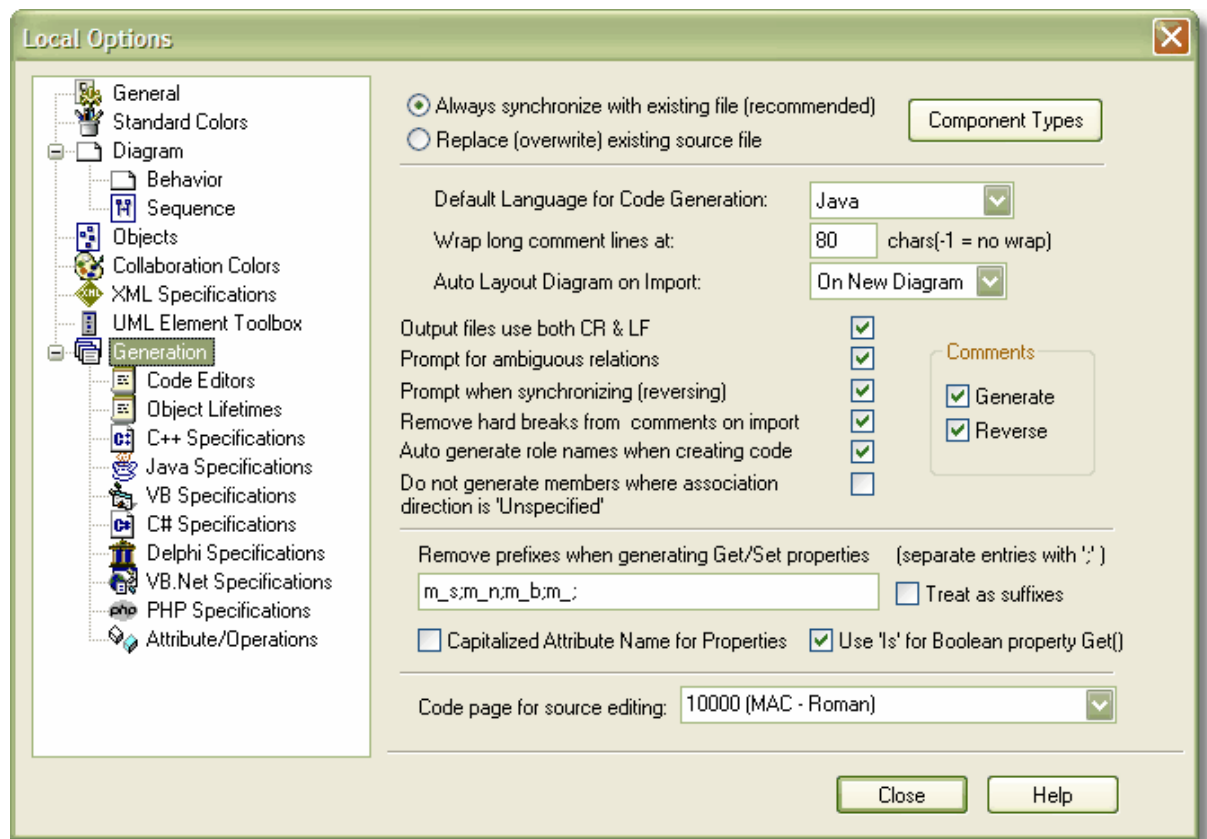
To generate code for a single class, first ensure the design of the model element (class or interface) is complete. Also ensure you have added inheritance links to parents and associations to other classes which are used. Also add inheritance links to Interfaces that your class implements - Enterprise Architect offers the option to generate function stubs for all interface methods that a class implements. Once the design is satisfactory, follow the steps below.

To Generate Code for a Single Class

1. Open the diagram containing the class or interface for which you wish to generate code.
2. Right click on the class/interface and select *Generate Code...* from the context menu.
3. The *Generate Code* dialog appears. Enter a *Path* name for your source code



4. Press *Advanced* to set any custom options (for this class alone).



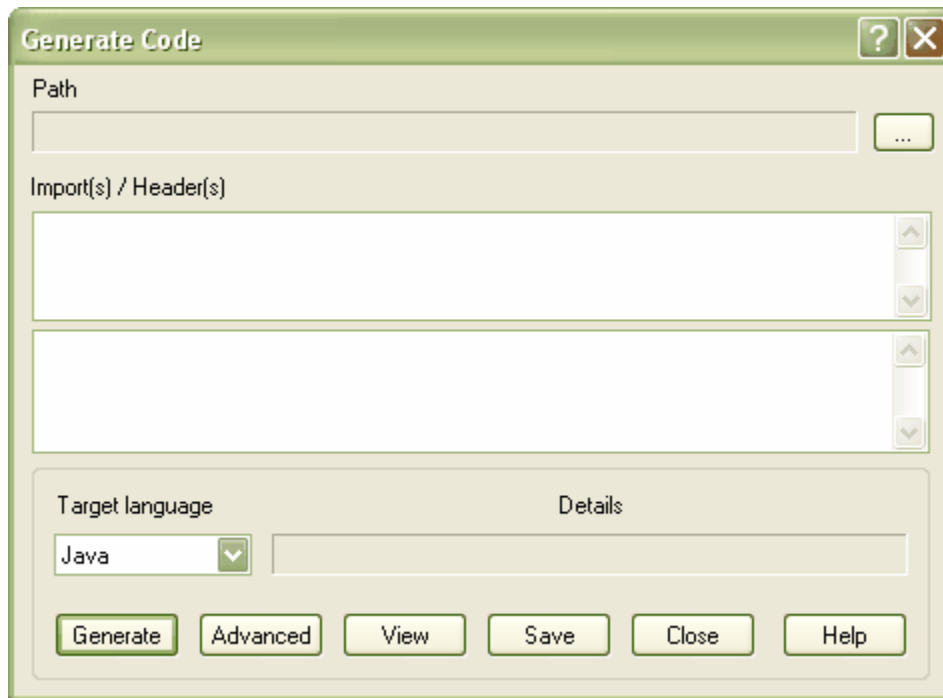
5. Add any import statements, #includes or other header information in the spaces provided (note that in the case of Visual Basic this information is ignored, in the case of Java the two import text boxes are merged and in the case of C++ the first import text area is placed in the header file and the second in the body (.cpp) file)
6. Press **Generate** to create the source code

When complete, you may press **View** to see what has been generated. Note that you should set up your default viewer/editor for each language type first.

See also: [Source Code Viewer](#)

9.2.3 The Code Generation Dialog

The **Code Generation** dialog allows you to control how and where your source code is generated. Normally you will access this dialog from the context menu of a single class or interface. Right click on the class or interface and select **Generate Code** from the context menu. Alternatively, select the class or interface and press **Ctrl+G**.



This dialog allows you to set

- The *Path* where the source will be generated. Press the *Browse [...]* button to bring up a file browser dialog.
- The *Target Language* for generation. Select the language to generate - this will then become the permanent option for that class, so change it back if you only want to do one pass in another language.
- *Advanced* settings. Note that the settings you make here only apply to the current class.
- *Import statements #1*. An area for you to enter any special import statements (or #include in C++). For C++ this area is placed in the header file.
- *Import statements #2*. An area to define additional import or include statements (or even macros and #defines in C++). In C++ this area is placed in the CPP file, in Java it is appended to the first import statements and placed in the .java file.
- *Generate*. Press this to generate your source code - you will be advised of progress as the generation proceeds.
- *View*. Press this to view the generated source code in your default editor. You may also set up the default editor on the *Code Editors* section of the *Local Options* dialog (*Tools | Options*).

9.2.4 Generate a Group of Classes

In addition to being able to generate code for an individual class, you may also select a group of classes for batch code generation. When you do this, you will accept all the default code generation options for each class in the set.

To Generate Multiple Classes

1. Select a group of classes and/or interfaces in a diagram.
2. Right click on an element in the group to open the context menu.
3. Select *Generate all Selected objects with default options* from the *Code Generation* submenu.
4. If no output name has been specified for the class/interface already, EA will prompt you for a suitable

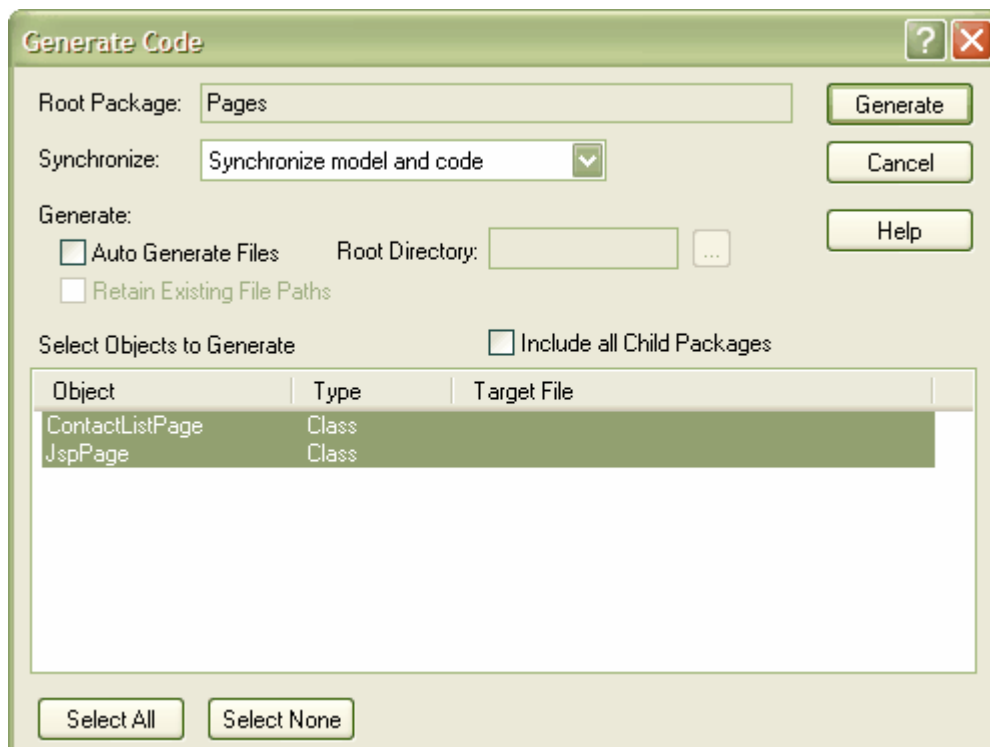
name as the generation proceeds.

Note: If all the elements selected are not classes or interfaces the option to generate code will not be available.

9.2.5 Generate a Package

In addition to generating source code from single classes and groups of classes, you can also generate code from a package. This feature provides options to recursively generate child packages and automatically generate directory structures based on the package hierarchy. This allows you to generate a whole branch of your project model in one step.

The Package Code Generation dialog is accessed by selecting *Generate Package Source Code* from the *Project | Source Code Engineering* submenu. Alternatively, right click on a package from the Project Browser and select *Generate Source Code* from the *Code Engineering* submenu.



Control	Description
Root Package	The name of the package to be generated
Synchronize	These options specify how existing files should be generated
Auto Generate Files	Specifies whether EA should automatically generate file names and directories, based on the package hierarchy
Root Directory	If Auto Generating Files, specifies the path under which the generated directory structure will be created
Retain Existing File Paths	If Auto Generating Files, specifies whether to use existing file paths associated with classes. If unselected, EA generates classes to automatically determined paths, regardless of whether source files are already associated with classes.
Include all Child Packages	If checked, all classes from all sub-packages of the target package are included in the list. This option facilitates recursive generation of a given package and its sub-packages.
Select Objects to Generate	Lists all classes that are available for generation under the target packages. Only selected (highlighted) classes will be generated. Classes are listed with their target source file.
Select All	Marks all classes in the list as selected
Select None	Marks all classes in the list as unselected
Generate	Starts the generation of all selected classes
Cancel	Exits the Code Template Editor dialog. No classes will be generated

Generate a Package

To generate a package, follow these steps:

1. In the Project Browser, right click on the package you wish to generate code for.
2. In the context menu, select *Generate Source Code* from the *Code Engineering* submenu.
3. The *Generate Code* dialog will appear.
4. Select the appropriate synchronize option:
 - *Synchronize model and code* : Classes with existing files will be forward synchronized with that file. Classes with no existing file, will be generated to the displayed target file.
 - *Overwrite code* : All selected target files will be overwritten (forward generated).
 - *Do not generate* : Only selected classes that do not have an existing file will be generated. All other classes will be ignored.
5. Highlight the classes you wish to generate. Leave unselected any you do not wish to generate.
6. If you would like EA to automatically generate directories and filenames based on the package hierarchy, check the *Auto Generate Files* option. You will be prompted to select a root directory under which the source directories will be generated. By default, the Auto Generate feature ignores any file paths that are already associated with a class. You can change this behavior by checking the *Retain Existing File Paths* option.
7. If you wish to include all sub-packages in the output, check the *Include Child Packages* option. When you have made your selection, press *Generate* to start the process.

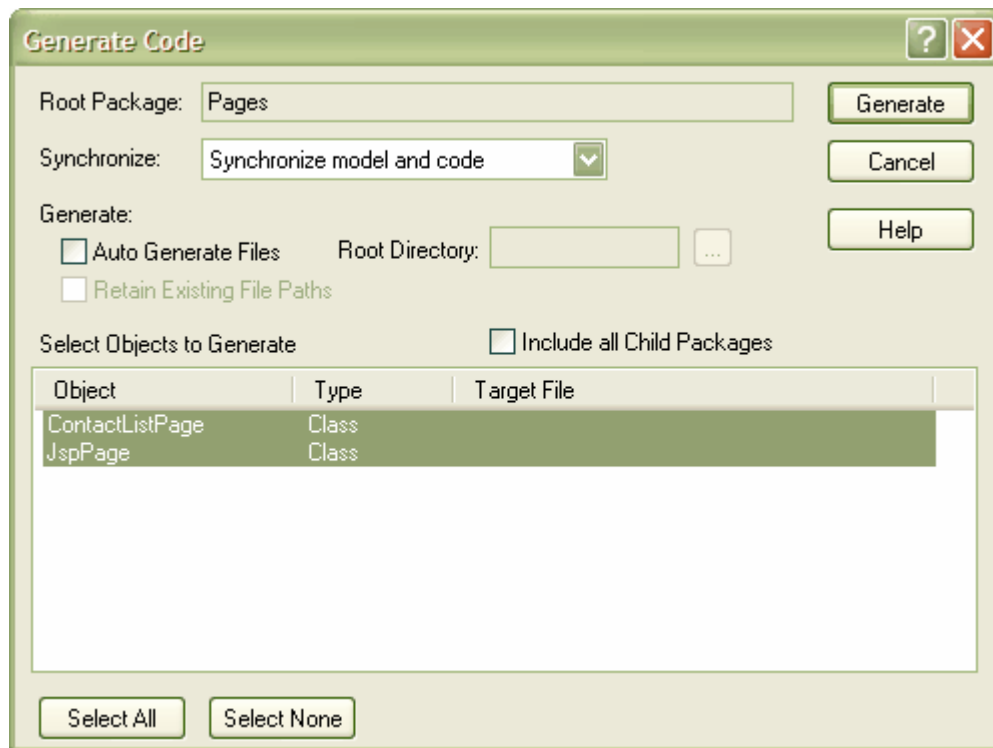
As the code generation proceeds you will be kept informed of progress. If a class requires an output filename you will be prompted to enter one at the appropriate time.

9.2.6 Generate Package Dialog

This dialog allows you to set up which classes will be generated into source code during a batch code generation run.

To Access the Generate Package Dialog

1. In the Project Browser, right click on the package to generate code for, to open the context menu.
2. Select *Generate Source Code* from the *Code Engineering* submenu.



3. If you wish to include classes in sub-packages, check the *Include Child Packages* option.
4. Next, highlight the classes in the list that you wish generated and leave unselected those you do not.
5. You may notice that many classes have no file name associated with them - as the generation proceeds you will be prompted to enter a suitable output file name for each class.
6. Finally, press *Generate* to start the process.

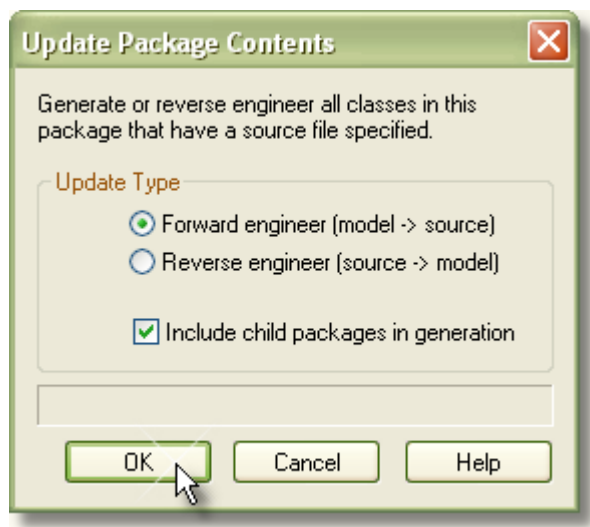
9.2.7 Update Package Contents

EA allows the synchronization of a directory tree. Follow the steps below:

1. Click on the Root package of the tree to synchronize in the Project Browser.
2. Select *Code Engineering* | *Synchronize Package Contents*.
3. Select whether to *Forward Engineer* or to *Reverse Engineer* the package classes.

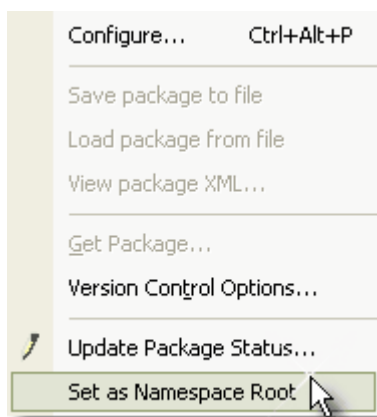
4. Select whether to *Include child packages* in the synchronization.
5. Press *OK* to start.

EA will use the directory names specified when the project source was first imported/generated and update either the model or the source code depending on the option chosen.



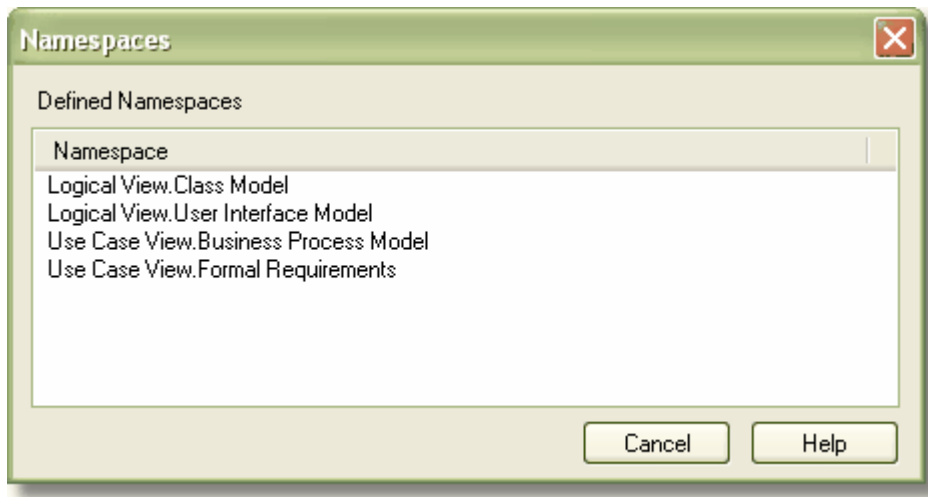
9.2.8 Namespaces

Languages such as Java support package structures or namespaces. EA lets you specify a package as a namespace root. To do this, right click on the package in the Project Browser, and select *Set as Namespace Root* from the *Package Control* submenu.

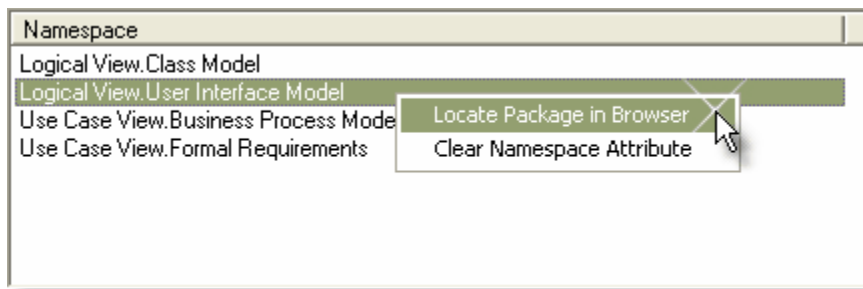


Once you have set a namespace root - Java code generated beneath this root will automatically add a package declaration at the head of the generated file indicating the current package location.

View a list of namespaces in the *Namespaces* dialog - select *Namespaces* from the *Project* menu.



Right click on an entry to remove the Namespace definition or locate the package in the main Project Browser



9.3 Code Engineering Settings

You can set the default code options such as the editors for each of the programming languages available for EA and special options for how source code is generated.

See:

- [General Options](#)
- [Local Paths](#)
- [Local Path Dialog](#)
- [Language Macros](#)
- [Setting Collection Classes](#)

9.3.1 General Options

The following topics describe general options that apply to all languages when generating code from EA. These options are all available under the *Generation* section on the *Local Options* dialog (*Tools | Options*).

See:

- [Options - Generation](#)
- [Options - Code Editors](#)
- [Options - Object Lifetimes](#)
- [Options - Attribute/Operations](#)

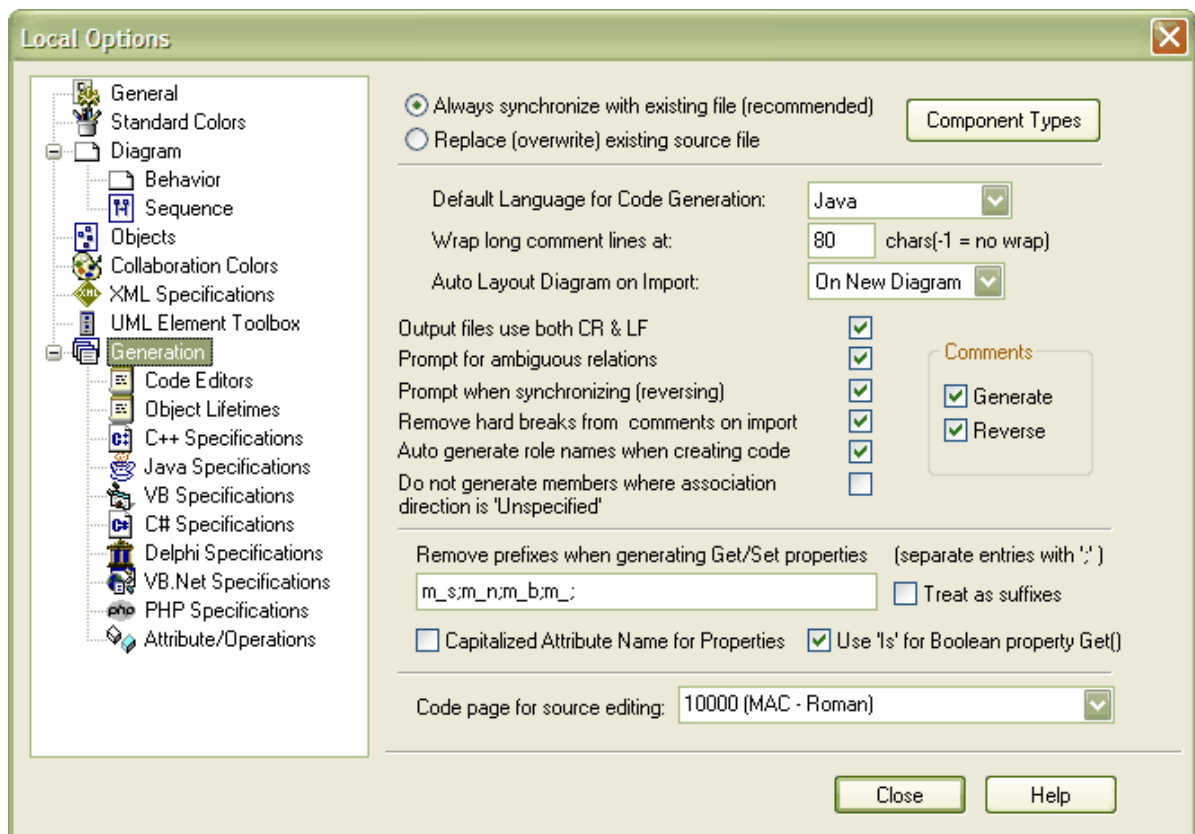
- [Synchronize Model and Code](#)
- [Code Page for Source Editing](#)

9.3.1.1 Options - Generation

When you generate code for a particular language, there are some options you may set. These include things like:

- Create a default constructor
- Create a destructor
- Generate copy constructor
- Select default language
- Generate methods for implemented interfaces
- Set the unicode options for code generation.

These options are accessed via the *Generation* section of the *Local Options* dialog (*Tools / Options*). Below is the main dialog; navigate using the side browser to access other option pages. Most of the settings are self-explanatory. *Remove prefixes when generating Get/Set properties* provides a field to specify prefixes used in your variable naming conventions, if you want those prefixes removed in the variables' corresponding get/set functions.

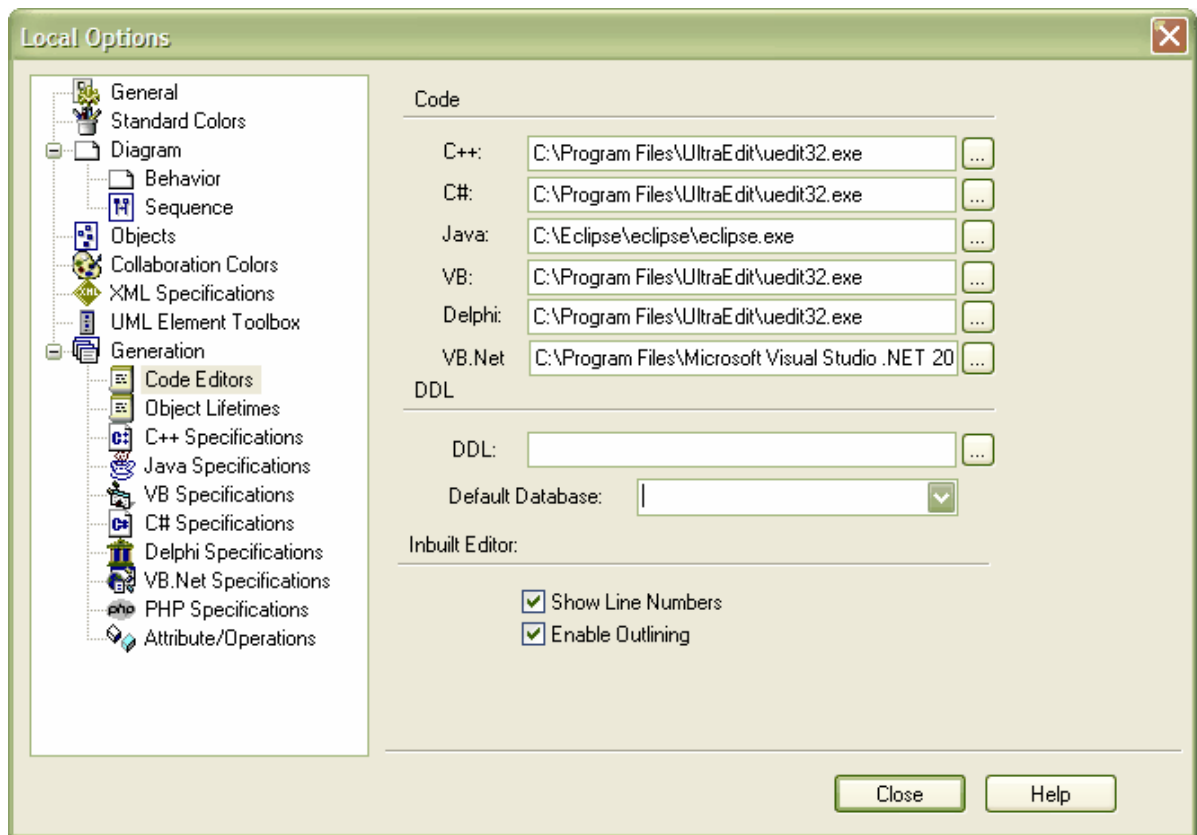


Note: It is worthwhile to configure these settings, as they serve as the defaults for all classes in the model. You may override these on a per-class basis using the custom settings (from the **Code Generation** dialog).

9.3.1.2 Options - Code Editors

This set of options allows you to configure your default editors for each language. You may also configure an editor for DDL viewing/editing.

This option is accessed via the **Generation** section of the **Local Options** dialog (**Tools | Options**). The source code viewer to display line numbers and to enable outlining by checking the **Show Line Numbers** checkbox and **Enable Outlining** Checkbox respectively.

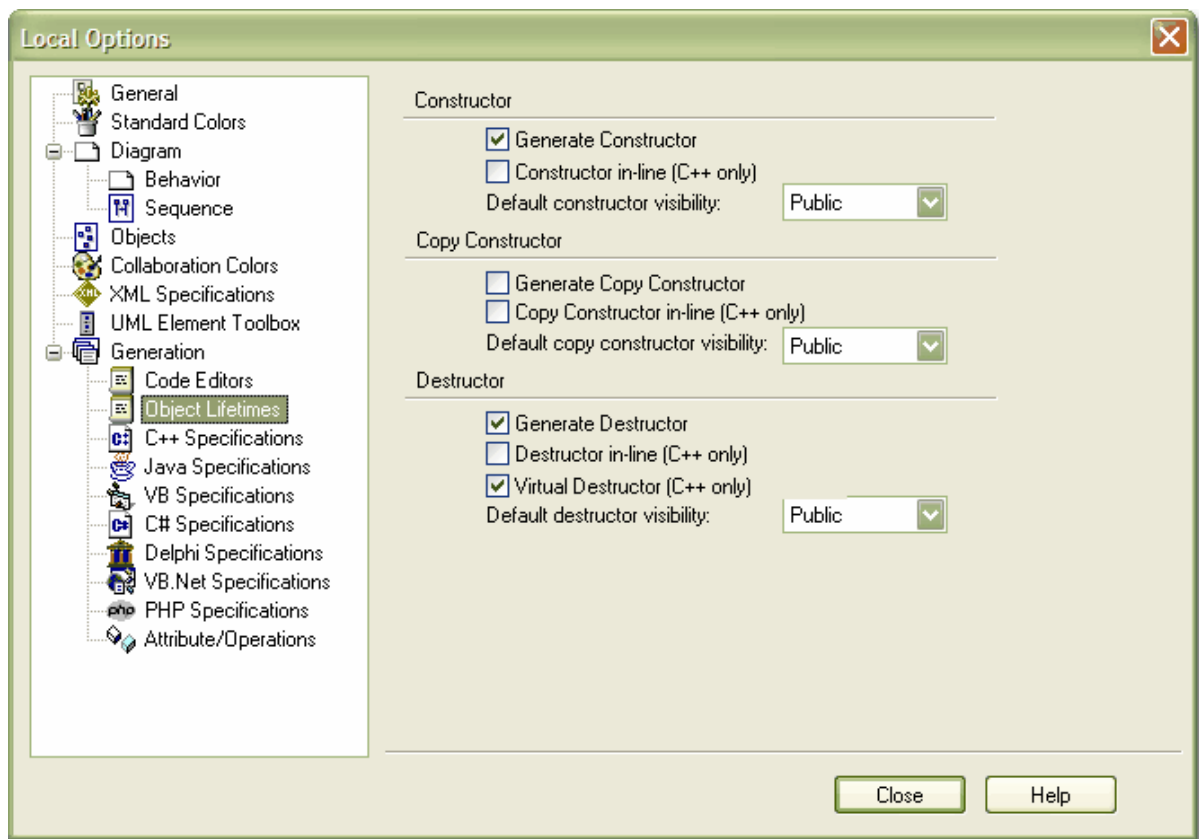


9.3.1.3 Options - Object Lifetimes

This set of options allows you to configure:

- Constructor details when generating code
- Whether to create a copy constructor
- Destructor details

This option is accessed via the **Generation** section of the **Local Options** dialog (**Tools | Options**).



9.3.1.4 Options - Attribute/Operations

This set of options allows you to configure:

- The default name generated from imported attributes
- Generate methods for implemented interfaces
- Delete model attributes not included in the code during reverse synchronization
- Delete model methods not included in the code during reverse synchronization
- Delete code from features contained in the model during forward synchronization.

This option is accessed via the *Generation* section of the *Local Options* dialog (*Tools | Options*).

Attribute Specifications

Default name for associated attrib:

Generate Get/Set methods for associated attributes

On reverse synch, delete model attributes not in code

Operation Specifications

Generate methods for implemented interfaces

On reverse synch, delete model methods not in code

Options

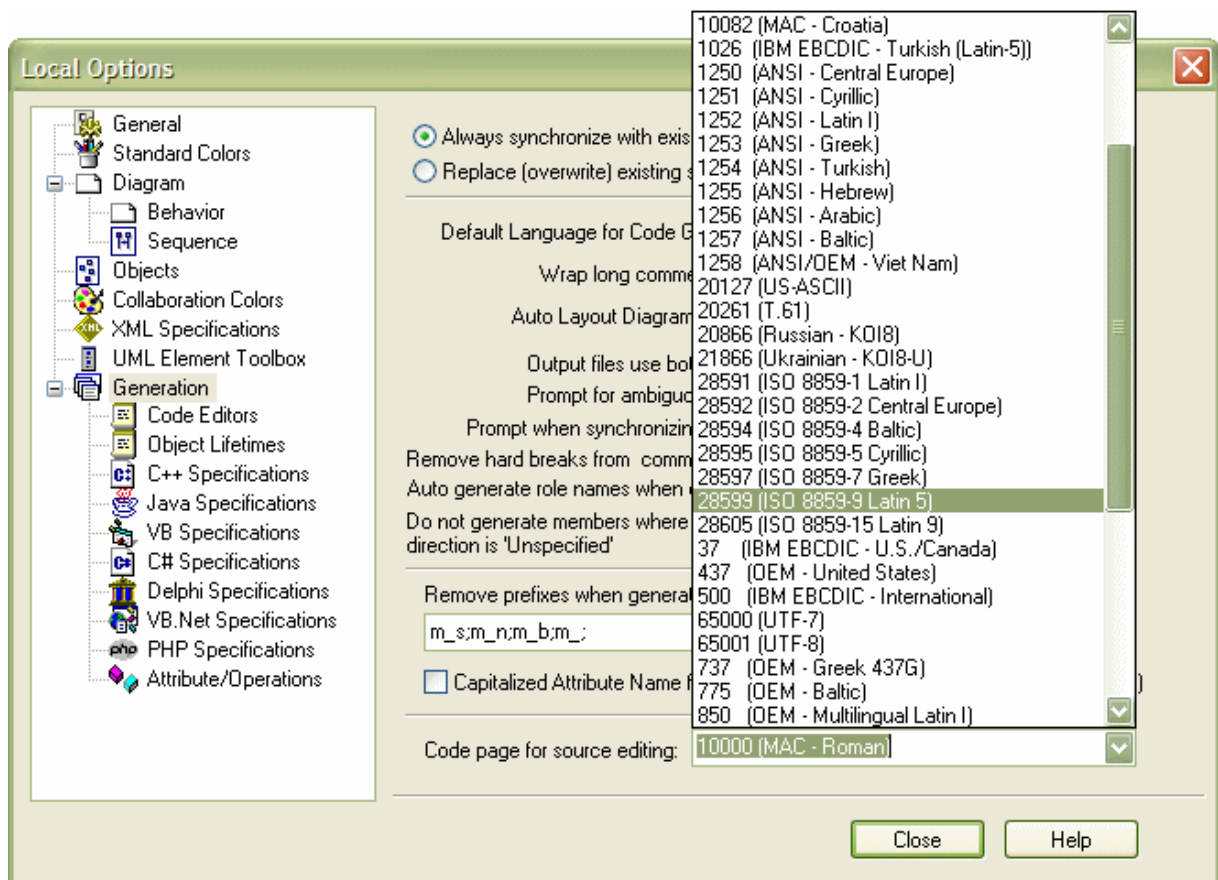
After save, re-select edited item

On forward synch, prompt to delete code features not in model

9.3.1.5 Code Page for Source Editing

The Unicode version of EA allows the user to define the Unicode character set for code generation. To set the Unicode character set use the following instructions:

1. Open the Local Options dialog by selecting **Tools | Options**.
2. Open the Generation item and select the **Code page for source editing** drop down menu.



3. Select the appropriate Unicode character set.

Note: This feature is only applicable to the Unicode version of EA.

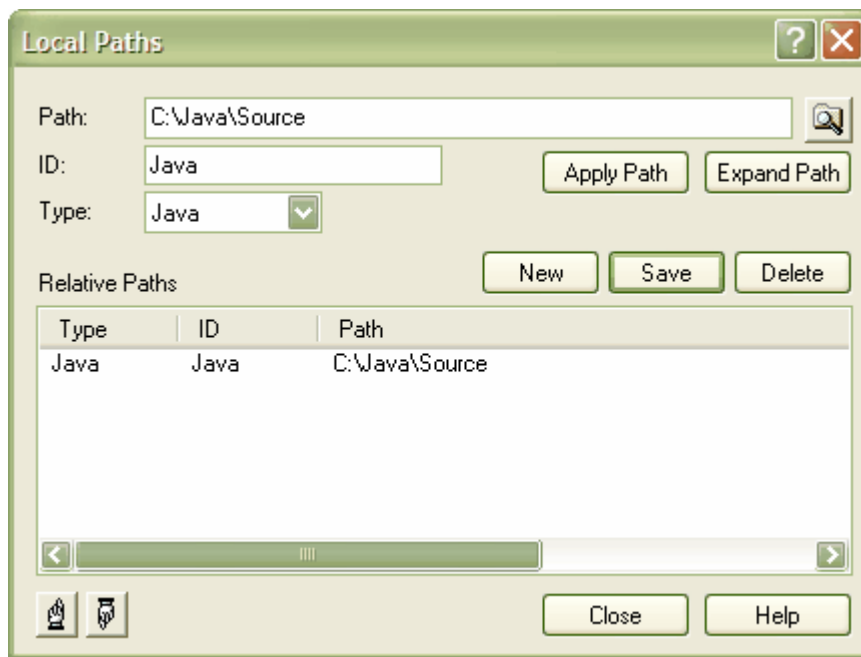
9.3.2 Local Paths

In many cases, a team of developers may be working on the same Enterprise Architect model. Often, each developer will store their version of the source code in their local file system, but not always at the same location as their fellow developers.

Local paths take a bit of setting up, but if you want to work collaboratively on source and model concurrently, the effort is well worthwhile.

For example: developer A stores their .java files in a 'C:\Java\Source' directory, while developer B stores theirs in 'D:\Source'. Meanwhile, both developers want to generate and reverse engineer into the same EA model located on a shared network drive (or replicated).

To handle this scenario, EA lets you define local paths for each EA user. In the above scenario developer A may define a local path of JAVA_SOURCE = "C:\Java\Source" - so that all classes generated and stored in the EA project will be stored as %JAVA_SOURCE%\<xxx.java>. Developer B now defines a JAVA_SOURCE local path as D:\Source. Now, EA will store all java files in these directories as %JAVA_SOURCE%\<filename>, and on each developer's machine, the filename is expanded to the correct local version.



9.3.3 Local Path Dialog

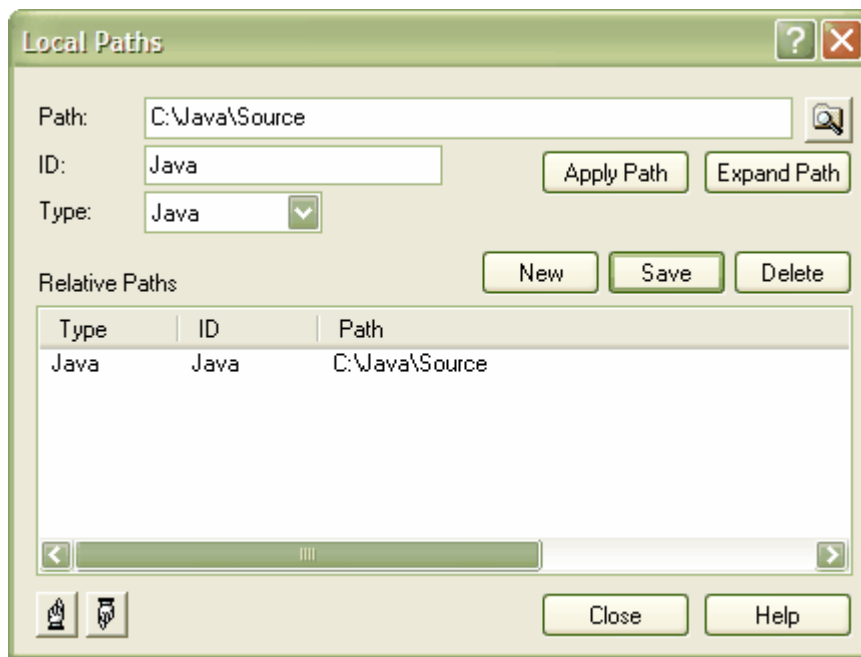
The *Local Paths* dialog allows you to set up local paths for a single user on a particular machine. For a description of what Local Paths are used for, see [Local paths](#).

The *Local Paths* dialog allows you to define:

- *Path* - this is the local directory in the file system (eg. d:\java\source)
- *ID* - this is the shared ID which will be substituted for the Local Path (eg. JAVA_SRC)
- *Type* - this is the language type- (eg. Java)

And also to:

- *Apply Path*. Select a path and press this to update any existing paths in the model (with full path names) to the shared relative path name (so "d:\java\source\main.java" may become "%JAVA_SRC%\main.java)
- *Expand Path*. The opposite of the item above. This lets you remove a relative path and substitute the full path name.
- Using the two above items you can update and change existing paths.



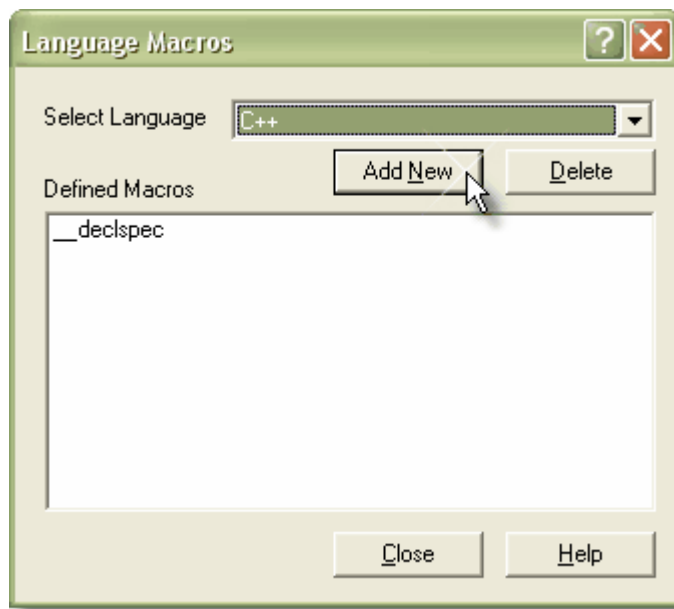
9.3.4 Language Macros

When reverse engineering a language like C++, often there are pre-processor directives scattered throughout the code. This may make code management easier - but can hamper parsing of the underlying C++ language.

To help remedy this, you may define any number of MACRO definitions, which will be ignored during the parsing phase of the reverse engineering. It is still preferable, if you have the facility, to pre-process the code using the appropriate compiler first - this way, complex macro definitions and defines are expanded out and may be readily parsed. If you don't have this facility, then this option provides a convenient substitute.

To Define a MACRO

1. Select *Language Macros* from the *Configuration* menu.
2. On the *Language Macros* dialog, press *Add New*.
3. Enter details for your MACRO.
4. Press *OK*.



Macros Embedded within Declarations

Macros are sometimes used within the declaration of classes and operations as in the following examples:

```
class __declspec Foo
{
    int __declspec Bar(int p);
};
```

If declspec is defined as a C++ macro as outlined above, the imported class and operation will contain a tagged value named "DeclMacro1" with value "__declspec". (Subsequent macros would be define as "DeclMacro2", "DeclMacro3" etc). During forward engineering, these tagged values are used to regenerate the macros in code.

Defining Complex Macros

It is sometimes useful to define rules for complex macros that may span multiple lines. EA will ignore the entire code section defined by the rule. Such macros can be defined in EA as in the following example:

```
BEGIN_INTERFACE_PART ^ END_INTERFACE_PART
```

where the ^ symbol represents the body of the macro. Note : The spaces surrounding the ^ symbol are required.

9.3.5 Setting Collection Classes

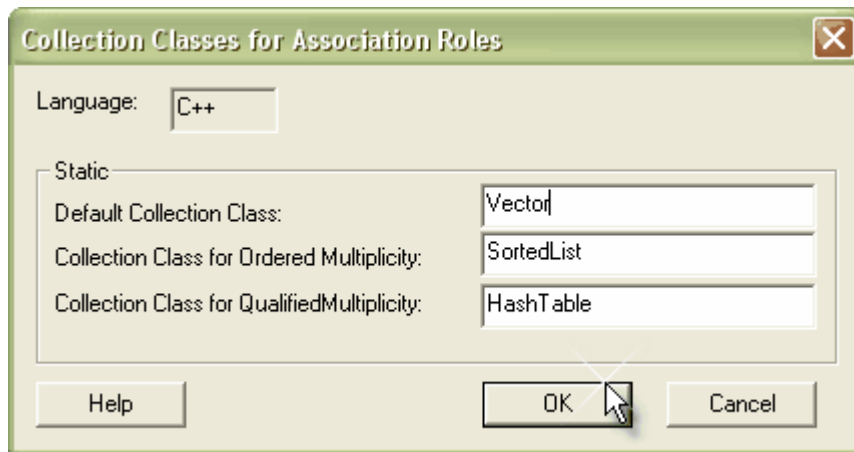
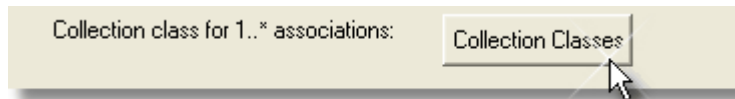
EA lets you define collection classes for generating code from association links where the target role has a multiplicity setting greater than 1.

There are two options for doing this:

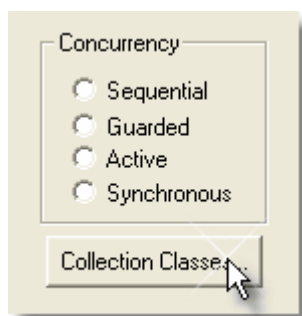
1. In the *Local Options* dialog (*Tools | Options*), on each page of the *Generation* section is a *Collection*

Classes button. This opens a dialog where you can define:

- The default collection class for 1..* roles
- The ordered collection class to use for 1..* roles
- The qualified collection class to use for 1..* roles



2. A **Collection Classes** button can also be found on the **Detail** tab of the **Class Properties** dialog (accessible from the right click context menu of any class). This opens the same dialog as above, where you can define:
 - The default collection class for 1..* roles *for this class only*
 - The ordered collection class to use for 1..* roles *for this class only*
 - The qualified collection class to use for 1..* roles *for this class only*



When EA generates code for a link that has a multiplicity role >1, the collection class is calculated as follows:

1. If the **Qualifier** is set use the qualified collection:
 - for the class if set
 - else use the code language qualified collection
2. If the **Order** option is set use the ordered collection:
 - for the class if set
 - else use the code language ordered collection

3. Else use the default collection:
 - for the class if set
 - else use the code language default collection

Note: You can include the marker "#TYPE#" in the collection name, and EA will replace this with the name of the class being collected at source generation time (eg. Vector<#TYPE#> would become Vector<foo>)

Additionally, there is a **Member Type** field on both the **Source Role** and **Target Role** tabs on the **Association Property** dialog (accessible from the right click context menu of any association). If you set this, the value entered will override all the above options. The example below shows a defined PersonList - when code is generated, because this has a **Multiplicity** > 1 and the **Member Type** is defined, the variable created will be of type PersonList.

The screenshot shows the 'UseCase Properties' dialog box with the 'Source Role' tab selected. The 'Login Role' is 'm_PersonList' and 'Access' is 'Protected'. The 'Role Notes' field contains 'A list of staff in order'. Under 'Containment', 'Not Specified' is selected. 'Multiplicity' is '1..*' and 'Multiplicity is Ordered' is checked. 'Aggregation' is 'none' and 'Changeable' is set to 'none'. The 'Member Type' field is 'PersonList'. Buttons for 'OK', 'Cancel', and 'Help' are at the bottom.

9.3.6 Language Options

You can set up various options for how EA will handle a particular language when generating code. These options are accessible on the **Local Options** dialog (select **Options** from the **Tools** menu). From the **Generation** section, select the required language. The following topics outline the options available for each language.

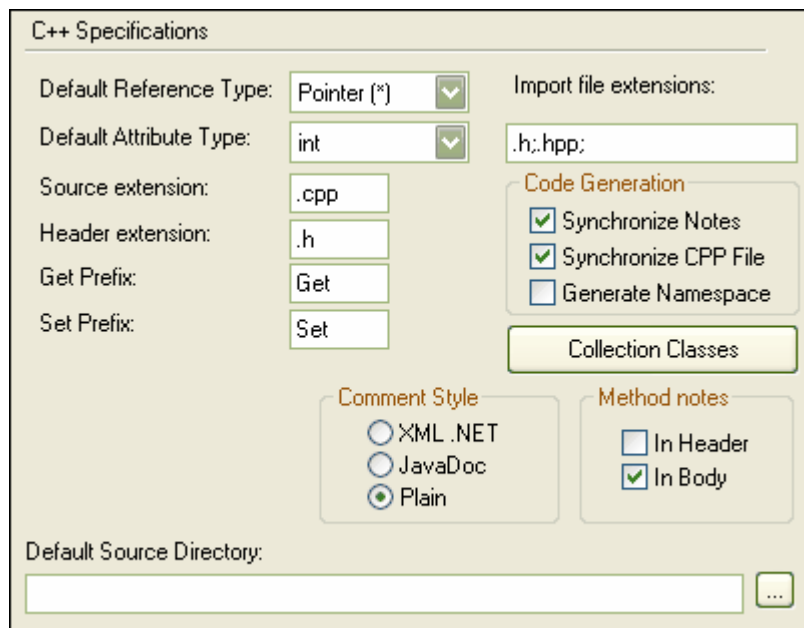
See the code generation options for:

- [C++](#)
- [Java](#)
- [Visual Basic](#)
- [C#](#)
- [Delphi](#)
- [VB.Net](#)
- [Delphi Properties](#)
- [PHP](#)
- [Reset Options](#)

9.3.6.1 Options - C++

Configure options for C++ code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *C++ Specifications* in the *Generation* section. The options you can specify include:

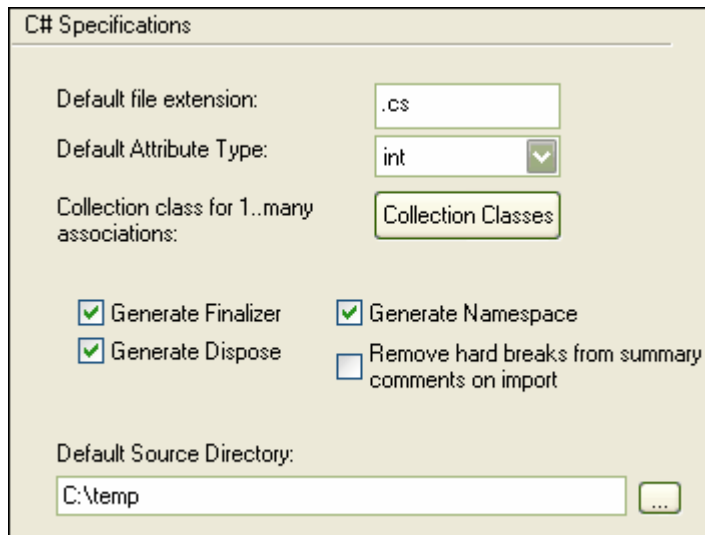
- The default reference type for getters and setters
- The default file extension(s)
- Default Get/Set prefixes
- Default source directory
- You may also set a default directory for opening and saving C++ source code



9.3.6.2 Options - C#

Configure options for C# code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *C# Specifications* in the *Generation* section. The options you can specify include:

- The default file extension
- The default 'Get' prefix
- The default 'Set' prefix
- You may also set a default directory for opening and saving C# source code



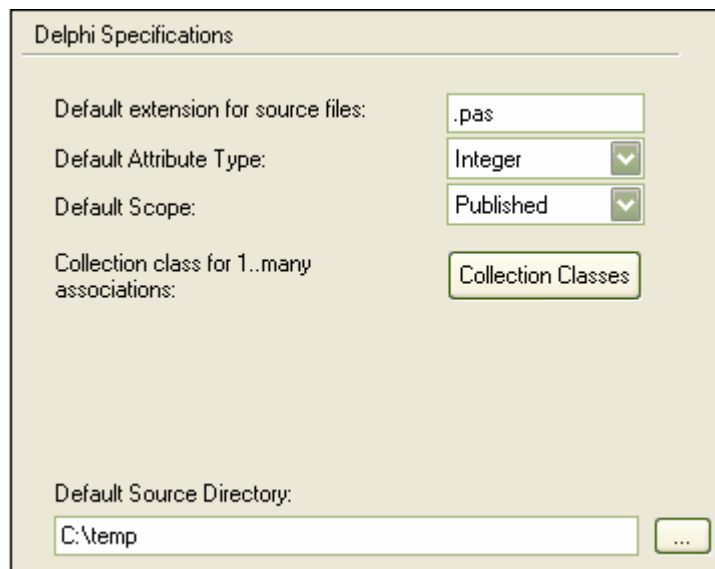
The screenshot shows the 'C# Specifications' dialog box. It contains the following fields and options:

- Default file extension:
- Default Attribute Type: (dropdown arrow)
- Collection class for 1..many associations:
- Generate Finalizer:
- Generate Namespace:
- Generate Dispose:
- Remove hard breaks from summary comments on import:
- Default Source Directory: (with a browse button '...')

9.3.6.3 Options - Delphi

Configure options for Delphi code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *Delphi Specifications* in the *Generation* section. The options you can specify include:

- Default file extension.
- Default source directory.
- You may also set a default directory for opening and saving Delphi source code.

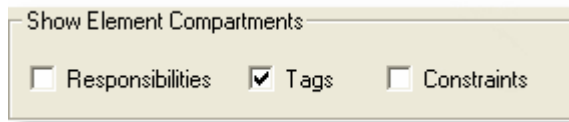


The screenshot shows the 'Delphi Specifications' dialog box. It contains the following fields and options:

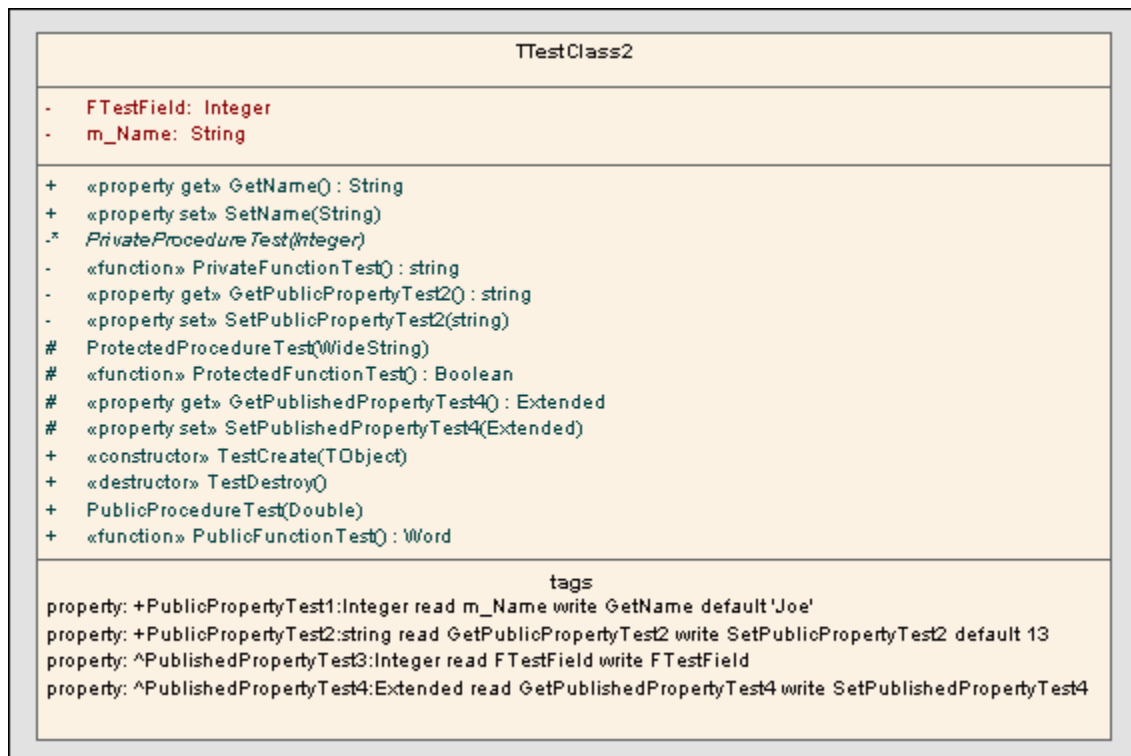
- Default extension for source files:
- Default Attribute Type: (dropdown arrow)
- Default Scope: (dropdown arrow)
- Collection class for 1..many associations:
- Default Source Directory: (with a browse button '...')

9.3.6.3.1 Delphi Properties

Enterprise Architect has comprehensive support for Delphi properties. These are implemented as tagged values - with a specialized property editor to help create and modify class properties. The class image below illustrates the appearance of a Delphi class that has had properties added to it. These are stored as tagged values, and by using the *Specify Feature Visibility* element context menu option, you can display the 'tags' compartment which contains the properties. Imported Delphi classes with properties will automatically have this feature made visible for your convenience.

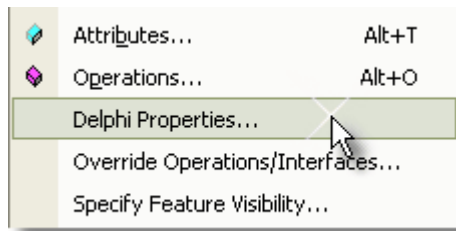


Note: when you use the *Create Property* dialog from the Attribute screen, EA will generate a pair of get and set functions, together with the required property definition as tagged values. You may manually edit these tagged values if required.



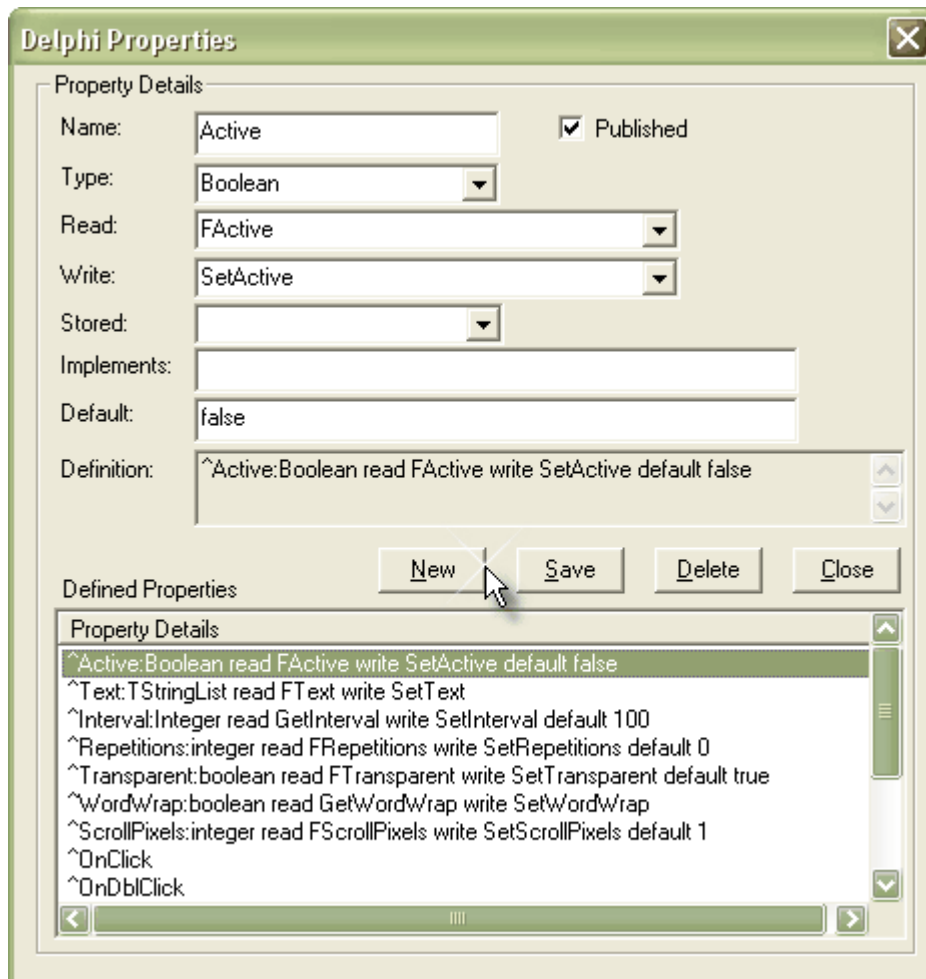
To manually activate the property editor

1. Ensure the class you have selected has the code generation language set to Delphi
2. Right click on the class and select *Delphi Properties* to open the editor



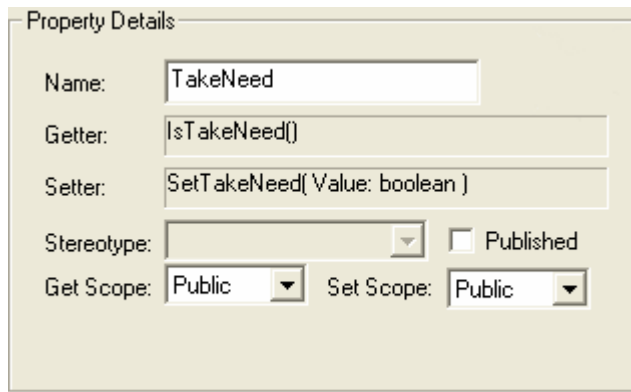
The Delphi Properties editor allows you to build properties in a simple and straightforward manner. From here you can:

- Change the name and scope (only Public and Published are currently supported).
- Change the property type(drop down list includes all defined classes in the project).
- Set the read and write information ... drop down list has all the attributes and operations from the current class. You may also enter free text.
- Set stored to True or False.
- Set the Implements information.
- Set the Default value is one exists.



Note: Public properties are displayed with a '+' symbol prefix and published with a '^'.

Note: When creating a property in the property wizard accessed through the Attributes dialog, you can set the scope to be Published if the property type is Delphi - see example below.



Property Details

Name:

Getter:

Setter:

Stereotype: Published

Get Scope: Set Scope:

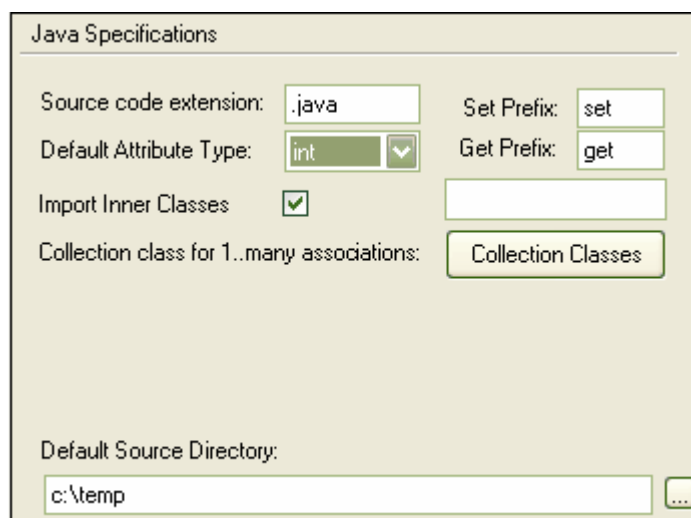
Limitations

- Only Public and Published supported.
- If you change the name of a property and forward engineer, a new property is added, but the old one must be manually deleted from the source file.

9.3.6.4 Options - Java

Configure options for Java code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *Java Specifications* in the *Generation* section. The options you can specify include:

- The default file extension
- The default 'Get' prefix
- The default 'Set' prefix
- You may also set a default directory for opening and saving Java source code



Java Specifications

Source code extension: Set Prefix:

Default Attribute Type: Get Prefix:

Import Inner Classes

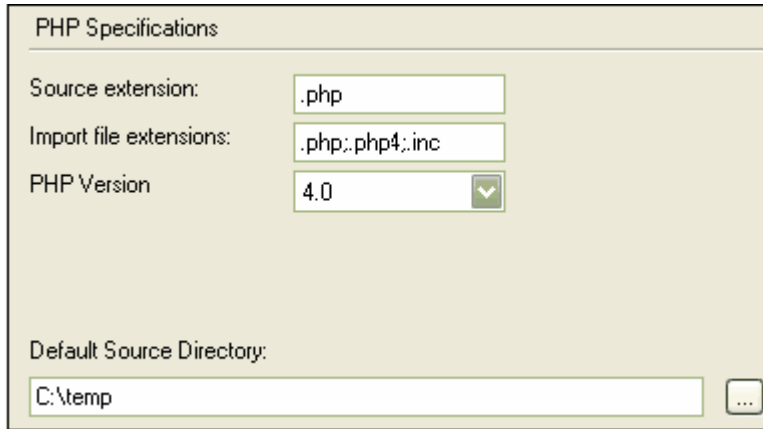
Collection class for 1..many associations:

Default Source Directory:

9.3.6.5 Options - PHP

Configure options for Java code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *PHP Specifications* in the *Generation* section. The options you can specify include:

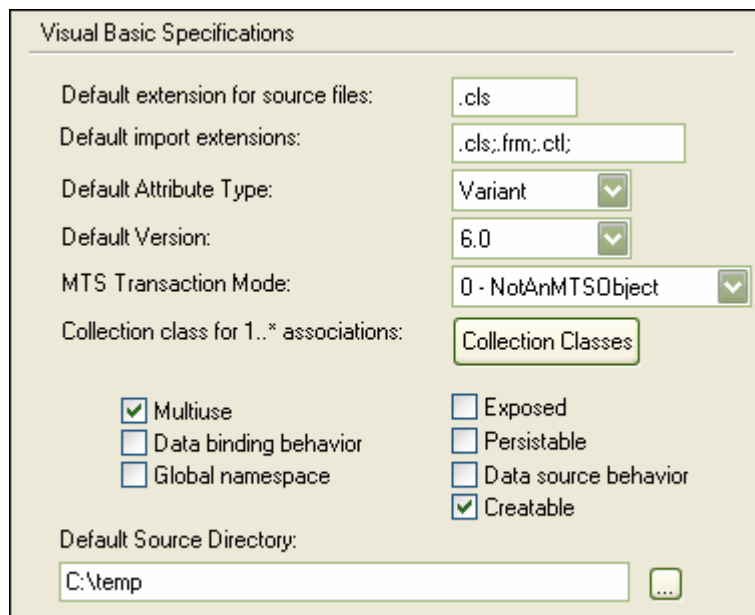
- The default source extension - Specify the extension to be used when creating files for PHP source.
- The default import extension - A semi-colon separated list of extensions to look at when doing a [directory code import](#) for PHP.
- The default PHP version - Version of PHP to generate.
- You may also set a default directory for opening and saving PHP source code



9.3.6.6 Options - Visual Basic

Configure options for Visual Basic code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *VB Specifications* in the *Generation* section. The options you can specify include:

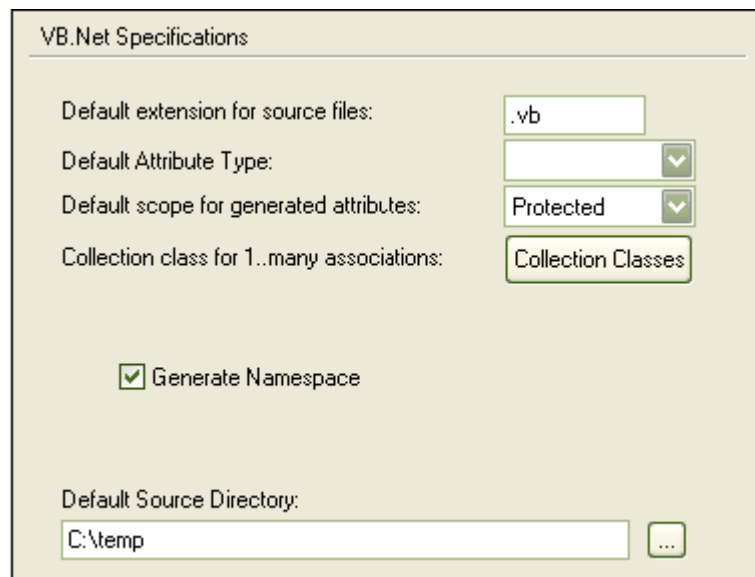
- The default file extension when reading/writing
- Default Visual Basic version
- MTS transaction mode for MTS objects
- Multiuse (true or false)
- Persistable
- Data binding
- Global namespace
- Exposed
- Data source behavior
- Creatable



9.3.6.7 Options - VB.Net

Configure options for VB.Net code generation by opening the *Local Options* dialog (select *Options* from the *Tools* menu). Select *VB.Net Specifications* in the *Generation* section. The options you can specify include:

- Default file extension
- Default source directory



9.3.6.8 Reset Options

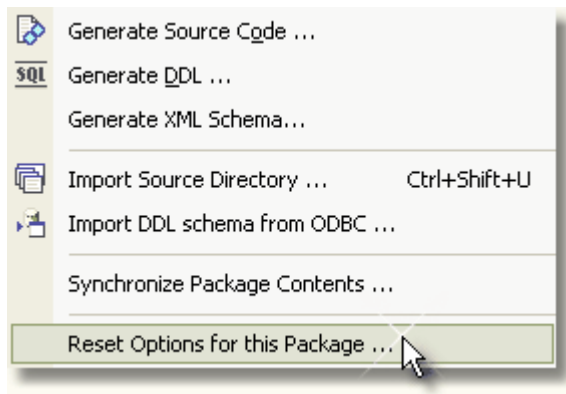
EA stores some of the options for a class when it is first created - and some are global. For example \$LinkClass is stored when you first create the class, so it won't automatically pick up the global change in the *Options* screen in existing classes. You need to modify the options for the existing class.

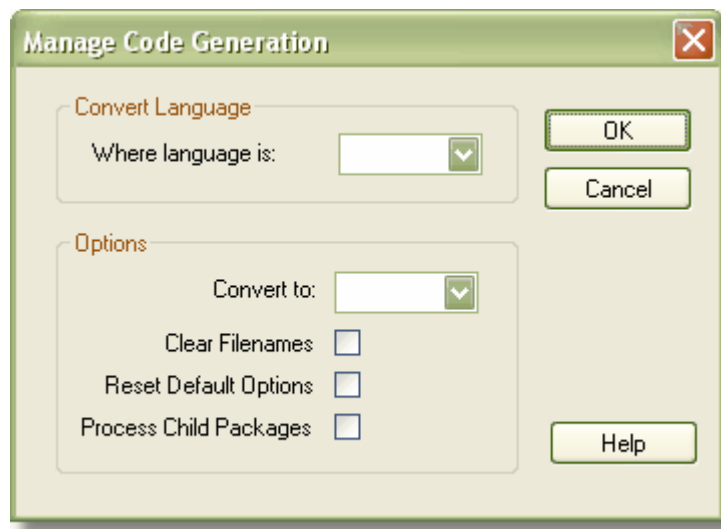
To Modify Options for a Single Class

1. Right click on the class to change and select *Generate Code* from the context menu.
2. Press *Advanced* on the Generate Code dialog.
3. In the Object Options dialog, select *Attributes/Operations*.
4. Change options and press *Close* to apply changes.

To Modify Options for All Classes Within a Package

1. Right click on the package Project Browser to view the context menu.
2. From the *Code Engineering* submenu, select *Reset Code Generation Options for Package*.
3. The *Manage Code Generation* dialog provides the ability to reset some defaults for each existing class.
4. Press *OK* to apply changes.





9.4 Synchronize Model and Code

In addition to generating code and importing code - EA has the option to synchronize the model and source code. This allows the model and the source code to be synchronized, creating a model which is up to date with the latest changes in the source code and vice versa.

For example - you may have already generated some source code, but have made subsequent changes to the model. When you generate again, EA will add any new attributes or methods to the existing source code, leaving intact what already exists. This means that developers may work on the source code, and then generate additional methods as required from the model, without having their code overwritten or destroyed.

In another scenario, changes may have been made to a source file, but the model has detailed notes and characteristics you do not want to lose. By synchronizing from the source into the model, additional attributes and methods are imported, but other model elements are left alone.

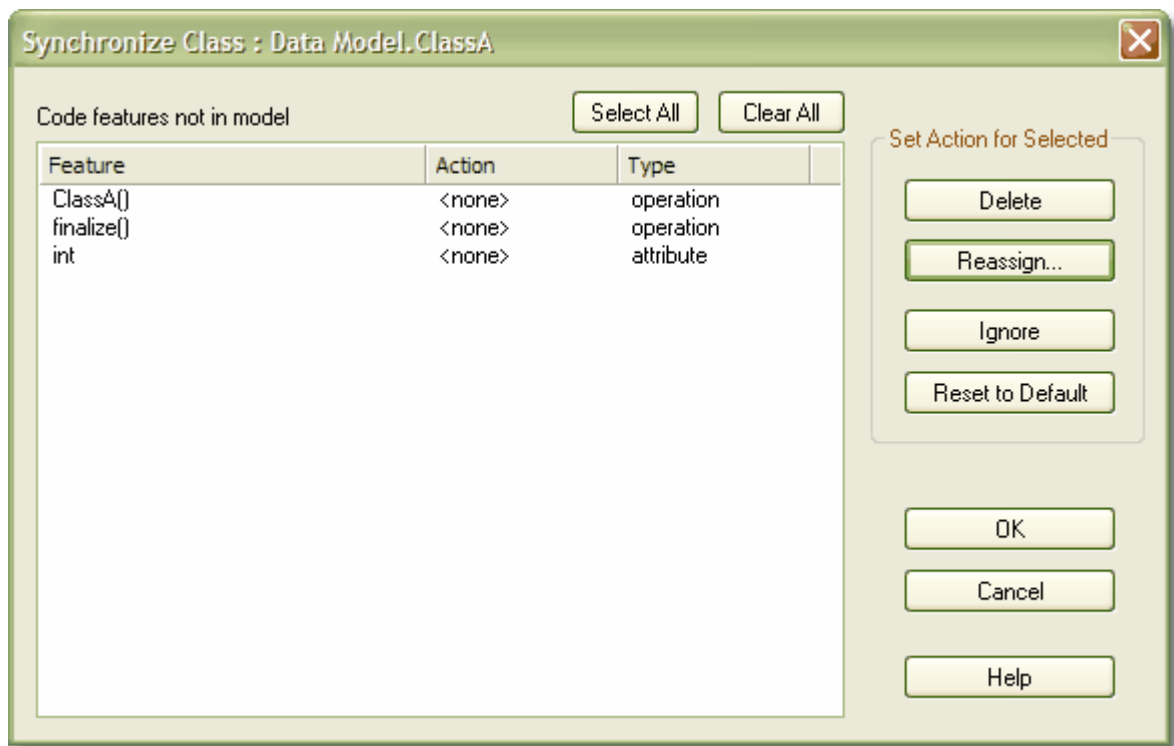
Using the two synchronization methods above, it is simple to keep source code and model elements up to date and synchronized.

Synchronizing Classes on Forward Generation

When there are features present in the code but not in the model the following options may be used during forward synchronization:

Note: these features are only available when the *On forward synch, prompt to delete code features not in model* in the [Options - Attributes and Operations](#).

- **Delete**, when the **Delete** button is pressed the selected code features will be removed from the code.
- **Reassign**, when the **Reassign** button is pressed it is possible to reassign code elements to elements in the model (this is only possible when an appropriate model element is present and is not already defined in the code).
- **Ignore**, when this button is selected the code elements not present in the model are ignored completely.
- **Reset to Default**, when this button is selected the settings for synchronizing during forward generation is set to **Ignore** meaning that the elements present in the code but not in the model are ignored completely.



9.5 MDG Link and Code Engineering

The MDG Link for Eclipse and MDG Link for Visual Studio.NET are standalone products which provide an enhanced code engineering functionality between EA and the development environments.

The MDG Link programs provide a lightweight bridge between EA and the development environment offering enhanced code generation, reverse engineering and synchronization between code and the UML model. Merging changes can be achieved with minimal effort and navigation between model and source code is significantly enhanced.

A trial version of MDG Link for Eclipse may be downloaded from http://www.sparxsystems.com.au/MDG_link_Eclipse.htm and MDG Link for Visual Studio.NET may be downloaded from http://www.sparxsystems.com.au/MDG_link_VS.htm.

9.6 Code Template Framework

The **Code Template Framework (CTF)** is used during the forward engineering of UML models.

The CTF allows:

The Code Template Framework allows:

- Generation of source code from UML models.
- Customization of the way in which EA generates source code.
- Forward engineering of languages not specifically supported by EA.

The CTF consists of:

The Code Template Framework consists of:

- Default [Code Templates](#) which are built into EA for forward engineering supported languages.
- A [Code Template Editor](#) for creating and maintaining User-defined Code Templates.

See also: [Synchronizing Code](#).

9.6.1 Code Templates

EA's code templates specify the transformation from UML elements to the various parts of a given programming language. The templates are written as plain text with a syntax that shares some aspects of both mark-up languages and scripting languages. A simple example of a template used by EA is the "Class template". It is used to generate source code from a UML class:

```
%ClassNotes%
%ClassDeclaration%
%ClassBody%
```

The above template simply refers to three other templates, namely ClassNotes, ClassDeclaration and ClassBody. The enclosing percent (%) signs indicate a "macro." Code Templates consist of various types of macros, each resulting in a substitution in the generated output. For a language such as C++, the result of processing the above template might be:

```
/**
 * This is an example class note generated using code templates
 * @author Sparx Systems
 */
class ClassA : public ClassB
{
  ...
}
```

See:

- [Base Templates](#)
- [Custom Templates](#)
- [Execution of Code Templates](#)
- [Code Template Syntax](#)

9.6.1.1 Base Templates

The CTF consists of a number of base templates. Each base template transforms particular aspects of the UML to corresponding parts of object-oriented languages. The following tables lists and briefly describe the base templates used in the CTF. The second table lists templates used for generating code for languages which have separate interface and implementation sections.

Template	Description
Attribute	Top-Level template to generate member variables from UML attributes
Attribute Declaration	Used by Attribute template to generate member variable declaration
Attribute Notes	Used by Attribute template to generate member variable notes
Class	Top-Level template for generating classes from UML classes
Class Base	Used by Class template to generate a base class name in the inheritance list of a derived class
Class Body	Used by Class template to generate the body of a class
Class Declaration	Used by Class template to generate the declaration of a class
Class Interface	Used by Class template to generate an interface name in the inheritance list of a derived class
Class Notes	Used by Class template to generate the class notes
File	Top-Level template for generating the source file. For languages such as C++ this corresponds to the header file.
Import Section	Used in the File template to generate external dependencies
Linked Attribute	Top-Level template for generating attributes derived from UML associations.
Linked Attribute Notes	Used by Linked Attribute template to generate the attribute notes
Linked Attribute Declaration	Used by Linked Attribute template to generate the attribute declaration
Namespace	Top-Level template for generating Namespaces from UML packages. (Although not all languages have namespaces, this template can be used to generate an equivalent construct, such as packages in Java.)
Namespace Body	Used by Namespace template to generate the body of a namespace
Namespace Declaration	Used by Namespace template to generate the namespace declaration
Operation	Top-Level template for generating operations from a UML class's operations
Operation Body	Used by Operation template to generate the body of a UML operations
Operation Declaration	Used by Operation template to generate the operation declaration
Operation Notes	Used by Operation template to generate documentation for an operation
Parameter	Used by Operation Declaration template to generate parameters.

Template	Description
Class Impl	Top-level template for generating the implementation of a class
Class Body Impl	Used by Class Impl to generate the implementation of class members.
File Impl	Top-Level template for generating the implementation file.
File Notes Impl	Used by the File Impl template to generate notes in the source file
Import Section Impl	Used by the File Impl template to generate external dependencies
Operation Impl	Top-Level template for generating operations from a UML class's operations
Operation Body Impl	Used by Operation template to generate the body of a UML operations
Operation Declaration Impl	Used by Operation template to generate the operation declaration
Operation Notes Impl	Used by Operation template to generate documentation for an operation

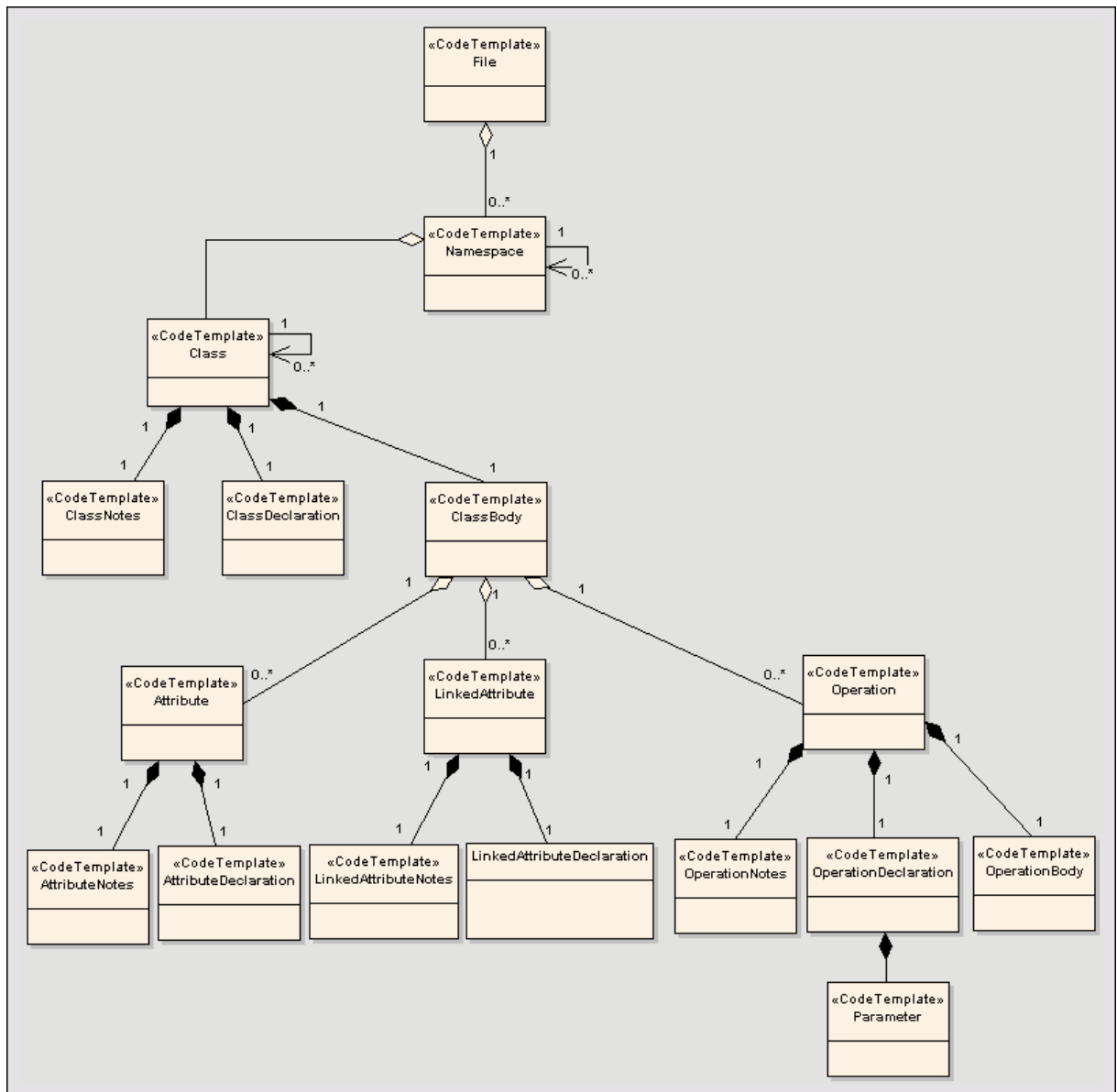
The base templates form a hierarchy, which varies slightly across different programming languages. A typical template hierarchy relevant to a language like C# or Java (which do not have header files) is shown below. In this figure we have modeled the templates as classes (in reality they are just plain text). This hierarchy would be slightly more complicated for languages like C++ and Delphi, which have separate implementation templates.

Each of the base templates need to be specialised to be of use in code engineering. In particular, each template is specialised for the supported languages (or "products"). For example, there is a ClassBody template defined for C++, and another for C#, and yet another for Java and so on. By specialising the templates, we can tailor the code generated for the corresponding UML entity.

Once the base templates are specialised for a given language, they may then be further specialised based on:

- A class's stereotype
- A feature's stereotype (where the feature can be an operation or attribute)

This type of specialisation allows a C# operation for example, which is stereotyped as <<property>>, to have a different OperationBody template than an ordinary operation. The OperationBody template may then be specialised further, based on the class stereotype.



Above: Class Model showing the hierarchy of Code Generation templates for a language such as C# or Java. The aggregation links denote references between templates.

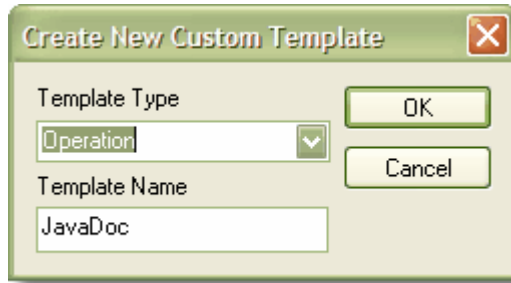
9.6.1.2 Custom Templates

Custom templates give users the option to generate an element in multiple different ways. EA Allows you to define custom templates that are associated with given elements and call these templates from existing templates. You can even add stereotype overrides to your custom templates.

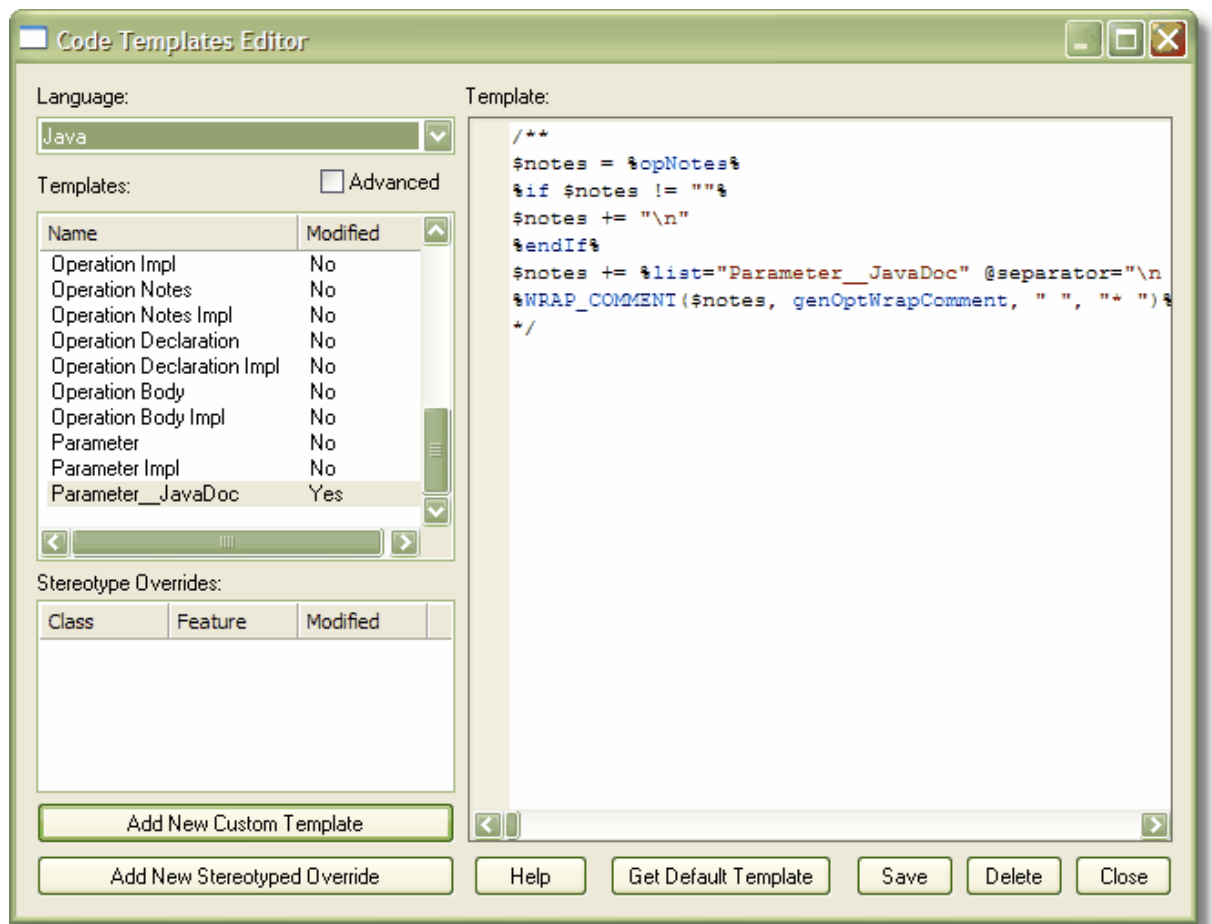
For example, you may want to list all of your parameters and their notes in your method notes. To create a new custom template:

1. Select Code Generation Templates from the Configuration menu to open the Code Templates Editor or use the **CTRL + Shift + P**.

2. Select the appropriate language, from the *Language* list.
3. Press *Add New Custom Template*.
4. This will open the Create New Custom Template dialog and choose the type from the Template Type list. The elements currently supported are:
 - Attribute
 - Class
 - Operation
 - Parameter



5. Give the template an appropriate name, then press the *OK* button.
6. The new template will appear in the templates list and be marked as modified. It will be called <Element Type>__<Template Name>.
7. Select the appropriate template from the list of templates and create a template which fulfils your requirements.



- Press the **Save** button. This stores the new stereotyped template in the .EAP file. The template is now available from the list of templates and via direct substitution for use

9.6.1.3 Execution of Code Templates

A reference to a template (such as the %ClassNotes% macro, from our previous example) results in the execution of that template.

Each template is designed for use with a particular element. For example the ClassNotes template is to be used with UML Class elements.

The element that is currently being generated is said to be "in scope". If the element in scope is stereotyped EA looks for a template that has been defined for that stereotype. If a match is found, the specialised template is executed. Otherwise the default implementation of the base template is used.

Templates are processed sequentially, line by line, replacing each macro with its underlying text value from the model.

9.6.1.4 Code Template Syntax

Code Templates are written as plain text, using EA's code template editor (see [The Code Template Editor](#)). The template syntax centres around three basic constructs:

- [Literal Text](#)
- [Macros](#)
- [Variables](#)

Templates may contain any or all of these constructs, which are described in the following sections.

9.6.1.4.1 Literal Text

All text within a given template that is not part of a macro or a variable definition/reference, is considered literal text. With the exception of blank lines, which are ignored, literal text is directly substituted from the template into the generated code.

Consider the following excerpt from the java Class Declaration template:

```
%PI=" "%
%CONVERT_SCOPE(classScope)%

%classStereotype=="static" ? "static" : ""%
%classStereotype=="final" ? "final" : ""%
%classStereotype=="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 need to be generated from within the templates, they can be safely reproduced using the following direct substitution macros:

Macro	Description
%dl%	produces a literal \$ character
%pc%	produces a literal % character
%qt%	produces a literal " character

9.6.1.4.2 Macros

Macros provide access to element fields within the UML model and are also used to structure the generated output. All macros are enclosed within percent (%) signs. The CTF contains five basic types of macros:

- [Template substitution macros](#)
- [Field substitution macros](#)
- [Tagged value substitution macros](#)
- [Control macros](#)
- [Function macros](#)

In general, macros (including the % delimiters) are substituted with literal text in the output. For example consider the following snippet from the Class Declaration template:

```
... class %className% ...
```

The field substitution macro, %className%, would result in the current class' name being substituted in the output. So if the class being generated was named "Foo", the output would be:

```
... class Foo ...
```

Each of the basic macro types are explained in the following topics.

9.6.1.4.2.1 Template Substitution Macros

The template substitution macros correspond to the templates listed in [Base Templates](#). These macros result in the execution of the named template. By convention, template macros are named according to Pascal casing.

Structure: %<TemplateName>%

where <TemplateName> may be one of the templates listed below.

When a template is referenced from within another template, it is generated with respect to the elements currently in scope. The specific template is selected based on the stereotypes of the elements in scope.

As noted previously, there is an implicit hierarchy among the various templates. Some care should be taken in order to preserve a sensible hierarchy of template references. For example, it does not make sense to use the %ClassInherits% macro within any of the attribute or operation templates. Conversely, the Operation and Attribute templates are designed for use within the ClassBody template.

The CTF contains the following template substitution macros:

- AttributeDeclaration
- AttributeNotes
- Attribute
- Class
- ClassImpl
- ClassBase
- ClassBody
- ClassBodyImpl
- ClassDeclaration
- ClassDeclarationImpl
- ClassInherits
- ClassInterface
- ClassNotes
- ClassParameter
- File
- FileImpl
- ImportSection
- ImportSectionImpl
- InnerClass
- InnerClassImpl
- LinkedAttribute
- LinkedAttributeNotes
- LinkedAttributeDeclaration
- LinkedClassBase
- LinkedClassInterface
- Namespace
- NamespaceBody
- NamespaceDeclaration
- NamespaceImpl
- Operation
- OperationBody
- OperationBodyImpl
- OperationDeclaration
- OperationDeclarationImpl
- OperationImpl
- OperationNotes
- Parameter

9.6.1.4.2.2 Field Substitution Macros

The field substitution macros provide access to data in the model. In particular, they are used to access data fields from:

- Packages
- Classes
- Attributes

- Operations
- Parameters

Field substitution macros are named according to Camel casing. By convention, the macro is prefixed with an abbreviated form of the corresponding model element. For example, attribute-related macros begin with "att", as in the `%attName%` macro, to access the name of the attribute in scope.

The following table lists each of the field substitution macros with a description of the result. Note that macros which represent checkboxes return a value of "T" if the box is checked. Otherwise the value is empty.

Macro Name	Description
attAlias	Attribute dialog: "Alias"
attCollection	Attribute Detail dialog: "Attribute is a Collection" checkbox
attConst	Attribute dialog: "Const" checkbox
attContainerType	Attribute Detail dialog: "Container Type"
attContainment	Attribute dialog: "Containment"
attDerived	Attribute dialog: "Derived" checkbox
attGUID	The unique GUID for the current attribute
attInitial	Attribute dialog: "Initial"
attName	Attribute dialog: "Name"
attNotes	Attribute dialog: "Notes"
attProperty	Attribute dialog: "Property" checkbox
attQualType	The namespace qualified attribute type (dot delimited). If the attribute classifier has not been set, attQualType is equivalent to the attType macro.
attScope	Attribute dialog: "Scope"
attStatic	Attribute dialog: "Static" checkbox
attStereotype	Attribute dialog: "Stereotype"
attType	Attribute dialog: "Type"
attVolatile	Attribute Detail dialog: "Transient" checkbox
classAbstract	Class dialog: "Abstract" checkbox
classAlias	Class dialog: "Alias"
classArguments	Class Detail dialog: "C++ Templates: Arguments"
classAuthor	Class dialog: "Author"
classBaseName	Type Hierarchy dialog : "Class Name" (for use where no link exists between child and base class)
classComplexity	Class dialog: "Complexity"
classGUID	The unique GUID for the current class
classHasParent	True, if the class in scope has one or more base classes
classImports	Code Gen dialog: "Imports". For supported languages this also includes dependencies derived from associations
classIsActive	Class Advanced dialog: "Is Active" checkbox
classIsInstantiated	True, if the class is an instantiated template class
classIsLeaf	Class Advanced dialog: "Is Leaf" checkbox
classIsRoot	Class Advanced dialog: "Is Root" checkbox
classIsSpecification	Class Advanced dialog: "Is Specification" checkbox
classKeywords	Class dialog: "Keywords"
classLanguage	Class dialog: "Language"
classMacros	A space separated list of macros defined for the class
classMultiplicity	Class Advanced dialog: "Multiplicity"
className	Class dialog: "Name"
classNotes	Class dialog: "Note"
classParamDefault	Class Detail dialog

classParamName	Class Detail dialog
classParamType	Class Detail dialog
classPersistence	Class dialog: "Persistence"
classPhase	Class dialog: "Phase"
classQualName	The class name prefixed by its outer classes. Class names are separated by double colons (::).
classScope	Class dialog: "Scope"
classStereotype	Class dialog: "Stereotype"
classStatus	Class dialog: "Status"
classVersion	Class dialog: "Version"
eaDateTime	The current time with format: DD-MMM-YYYY HH:MM:SS AM/PM
eaGUID	A unique GUID for this generation
eaVersion	Program Version (Located in an EA dialog by selecting Help About EA)
elemType	The element type: "Interface" or "Class"
fileExtension	The file type extension of the file being generated
fileName	The name of the file being generated
fileNameImpl	The filename of the implementation file for this generation if applicable
fileHeaders	Code Gen dialog: "Headers"
fileImports	Code Gen dialog: "Imports"
genOptDefaultAssocAttName	Attribute Specifications dialog: "Default name for associated attrib"
genOptDefaultConstructorScope	Object Lifetimes dialog: "Default Constructor Visibility"
genOptDefaultCopyConstructorScope	Object Lifetimes dialog: "Default Copy Constructor Visibility"
genOptDefaultDestructorScope	Object Lifetimes dialog: "Default Destructor Constructor Visibility"
genOptCPPCommentStyle	C++ Specifications dialog: "Comment Style"
genOptCPPDefaultAttributeType	C++ Specifications dialog: "Default Attribute Type"
genOptCPPDefaultReferenceType	C++ Specifications dialog: "Default Reference Type"
genOptCPPDefaultSourceDirectory	C++ Specifications dialog: "Default Source Directory"
genOptCPPGenMethodNotesInHeader	C++ Specifications dialog: "Method Notes In Header" checkbox
genOptCPPGenMethodNotesInBody	C++ Specifications dialog: "Method Notes In Body" checkbox
genOptCPPGetPrefix	C++ Specifications dialog: "Get Prefix"
genOptCPPHeaderExtension	C++ Specifications dialog: "Header Extension"
genOptCPPSetPrefix	C++ Specifications dialog: "Set Prefix"
genOptCPPSourceExtension	C++ Specifications dialog: "Source Extension"
genOptCPPSynchCPPFile	C++ Specifications dialog: "Synchronize Notes"
genOptCPPSynchNotes	C++ Specifications dialog: "Synchronize CPP File"
genOptCSDefaultAttributeType	C# Specifications dialog: "Default Attribute Type"
genOptCSSourceExtension	C# Specifications dialog: "Default file extension"
genOptCSGenDispose	C# Specifications dialog: "Generate Dispose"
genOptCSGenFinalizer	C# Specifications dialog: "Generate Finalizer"

genOptCSGenNamespace	C# Specifications dialog: "Generate Namespace"
genOptCSDefaultSourceDirectory	C# Specifications dialog: "Default Source Directory"
genOptGenCapitalisedProperties	Generation dialog: "Capitalize Attribute Names for Properties" checkbox
genOptGenComments	Generation dialog: "Generate Comments" checkbox
genOptGenConstructor	Object Lifetimes dialog: "Generate Constructor" checkbox
genOptGenConstructorInline	Object Lifetimes dialog: "Constructor Inline" checkbox
genOptGenCopyConstructor	Object Lifetimes dialog: "Generate Copy Constructor" checkbox
genOptGenCopyConstructorInline	Object Lifetimes dialog: "Copy Constructor Inline" checkbox
genOptGenDestructor	Object Lifetimes dialog: "Generate Destructor" checkbox
genOptGenDestructorInline	Object Lifetimes dialog: "Destructor Inline" checkbox
genOptGenDestructorVirtual	Object Lifetimes dialog: "Virtual Destructor!" checkbox
genOptGenGetSetOpsForAssocAtts	Attribute Specifications dialog: "Generate Get/Set for associated attributes" checkbox
genOptGenImplementedInterfaceOps	Operation Specifications dialog: "Generate methods for implemented interfaces" checkbox
genOptGenPrefixBoolProperties	Generation dialog: "Use is prefix for boolean property Get()"
genOptGenRoleNames	Generation dialog: "Autogenerate role names when creating code"
genOptGenUnspecAssocDir	Generation dialog: "Do not generate members where Association direction is unspecified" checkbox
genOptJavaDefaultAttributeType	Java Specifications dialog: "Default attribute type"
genOptJavaGetPrefix	Java Specifications dialog: "Get Prefix"
genOptJavaDefaultSourceDirectory	Java Specifications dialog: "Default Source Directory"
genOptJavaSetPrefix	Java Specifications dialog: "Set Prefix"
genOptJavaSourceExtension	Java Specifications dialog: "Source code extension"
genOptPHPDefaultSourceDirectory	PHP Specifications dialog: "Default Source Directory"
genOptPHPGetPrefix	PHP Specifications dialog: "Get Prefix"
genOptPHPSetPrefix	PHP Specifications dialog: "Set Prefix"
genOptPHPSourceExtension	PHP Specifications dialog: "Default file extension"
genOptPHPVersion	PHP Specifications dialog: "PHP Version"
genOptVBMultiUse	VB Specifications dialog: "Multiuse" checkbox
genOptVBPersistable	VB Specifications dialog: "Persistable" checkbox
genOptVBDataBindingBehavior	VB Specifications dialog: "Data binding behavior" checkbox
genOptVBDataSourceBehavior	VB Specifications dialog: "Data source behavior" checkbox
genOptVBGlobal	VB Specifications dialog: "Global namespace" checkbox
genOptVBCreatable	VB Specifications dialog: "Creatable" checkbox
genOptVBExposed	VB Specifications dialog: "Exposed" checkbox
genOptVBMTS	VB Specifications dialog: "MTS Transaction Mode"
genOptVBNetGenNamespace	VBNet Specifications dialog: "Generate Namespace"
genOptVBVersion	VB Specifications dialog: "Default Version"

genOptWrapComment	Generation dialog: "Wrap length for comment lines"
importClassName	The name of the class being imported.
importFileName	The filename of the class being imported.
importFromAggregation	'T' if the class has an aggregation link to a class in this file, 'F' otherwise.
importFromAssociation	'T' if the class has an association link to a class in this file, 'F' otherwise.
importFromAtt	'T' if an attribute of a class in the current file is of the type of this class, 'F' otherwise.
importFromDependency	'T' if the class has a dependency link to a class in this file, 'F' otherwise.
importFromGeneralization	'T' if the class has a generalization link to a class in this file, 'F' otherwise.
importFromMeth	'T' if a method return type of a class in the current file is the type of this class, 'F' otherwise.
importFromParam	'T' if a method parameter of a class in the current file is of the type of this class, 'F' otherwise.
importFromRealization	'T' if the class has a realization link to a class in this file, 'F' otherwise.
importInFile	'T' if the class is in the current file, 'F' otherwise.
importPackagePath	The package path with a '.' separator of the class being imported.
linkAttAccess	Association Properties Target Role dialog: "Access"
linkAttCollectionClass	The collection appropriate for the linked attribute in scope
linkAttContainment	Association Properties Target Role dialog: "Containment"
linkAttName	Association Properties dialog: "Target"
linkAttNotes	Association Properties Target Role dialog: "Role Notes"
linkAttQualName	The namespace qualified Association target (dot delimited).
linkAttRole	Association Properties Target Role dialog: "Role"
linkAttTargetScope	Association Properties Target Role dialog: "Target Scope"
linkCard	Link Properties Target Role dialog: "Multiplicity"
linkGUID	The unique GUID for the current link
linkParentName	Generalization Properties dialog : "Target"
linkParentQualName	The namespace qualified Generalization target (dot delimited)
linkStereotype	The stereotype of the current link
linkVirtualInheritance	Generalization Properties dialog : "Virtual Inheritance"
opAbstract	Operation dialog: "Virtual" checkbox
opAlias	Operation dialog: "Alias"
opBehaviour	Operation Behavior dialog: "Behavior"
opConcurrency	Operation dialog: "Concurrency"
opConst	Operation dialog: "Const" checkbox
opGUID	The unique GUID for the current operation
opIsQuery	Operation dialog: "IsQuery" checkbox
opMacros	A space separated list of macros defined for the operation
opName	Operation dialog: "Name"

opName	Operation dialog: "Notes"
opPure	Operation dialog: "Pure" checkbox
opReturnArray	Operation dialog: "Return Array" checkbox
opReturnQualType	The namespace qualified operation return type (dot delimited). If the return type classifier has not been set, opReturnQualType is equivalent to the opReturnType macro.
opReturnType	Operation dialog: "Return Type"
opScope	Operation dialog: "Scope"
opStatic	Operation dialog: "Static" checkbox
Operation dialog: "Stereotype"	Operation dialog: "Stereotype"
opSynchronized	Operation dialog: "Synchronized" checkbox
packageAlias	Package dialog: "Alias"
packageName	Package dialog: "Name"
packageGUID	The unique GUID for the current package
packagePath	The string representing the hierarchy of packages, for the class in scope. Each package name is separated by a dot (.)
paramDefault	Operation Parameters dialog: "Default"
paramFixed	Operation Parameters dialog: "Fixed" checkbox
paramGUID	The unique GUID for the current parameter
paramIsEnum	True, if the parameter uses the enum keyword (C++)
paramKind	Operation Parameters dialog: "Kind"
paramName	Operation Parameters dialog: "Name"
paramNotes	Operation Parameters dialog: "Notes field"
paramQualType	The namespace qualified parameter type (dot delimited). If the parameter classifier has not been set, paramQualType is equivalent to the paramType macro.
paramType	Operation Parameters dialog: "Type"

Field substitution macros may be used in one of two ways:

Use 1: Direct Substitution

This form directly substitutes the corresponding value of the element in scope into the output.

Structure: `%<macroName>%`

where `<macroName>` can be any of the macros listed above.

Examples:

- `%className%`
- `%opName%`
- `%attName%`

Use 2: Conditional Substitution

This form of the macro allows alternative substitutions to be made depending on the macro's value.

Structure: `%<macroName> [== "<test>"] ? <subTrue> [: <subFalse>]%`

Where:

- `[<text>]` denotes that `<text>` is optional.
- `<test>` is a string representing a possible value for the macro.
- `<subTrue>` and `<subFalse>` may be a combination of quoted strings and the keyword value. Where the value is used, it gets replaced with the macro's value in the output.

Examples:

- `%classAbstract=="T" ? "pure" : ""%`
- `%opStereotype=="operator" ? "operator" : ""%`
- `%paramDefault != "" ? " = " value : ""%`

The above three examples will output nothing if the condition fails. In this case the false condition can be omitted resulting in the following usage.

Examples:

- `%classAbstract=="T" ? "pure"%`
- `%opStereotype=="operator" ? "operator"%`
- `%paramDefault != "" ? " = " value%`

The third example of both blocks shows a comparison checking for a non-empty value or existence. This test can also be omitted.

- `%paramDefault ? " = " value : ""%`
- `%paramDefault ? " = " value%`

All of the above examples containing `paramDefault` are equivalent. If the parameter in scope had a default value of 10, we would expect the output from each of them to be:

= 10

Note: In a conditional substitution macro, any whitespace following `<macroName>` is ignored. If whitespace is required in the output, it should be included within the quoted substitution strings.

9.6.1.4.2.3 Tagged Value Macros

Tagged value macros are a special form of field substitution macros, which provide access to element tags and the corresponding tagged values.

Use 1: Direct Substitution

This form of the macro directly substitutes the value of the named tag into the output.

Structure: %<macroName>:"<tagName>"%
 where <macroName> can be one of:

- attTag
- classTag
- opTag
- paramTag

which corresponds to the tags for attributes, classes, operations and parameters respectively. <tagName> is a string representing the specific tag name.

Examples:

- %opTag:"attribute"%

Use 2: Conditional Substitution

This form of the macro mimics the conditional substitution defined for field substitution macros.

Structure: %<macroName>:"<tagName>" [== "<test>"] ? <subTrue> [: <subFalse>]%

Where:

- <macroName> and <tagName> are as defined above.
- [<text>] denotes that <text> is optional.
- <test> is a string representing a possible value for the macro.
- <subTrue> and <subFalse> may be a combination of quoted strings and the keyword value. Where the value is used, it gets replaced with the macro's value in the output.

Examples:

- %opTag:"opInline" ? "inline" : ""%
- %opTag:"opInline" ? "inline"%
- %classTag:"unsafe" == "true" ? "unsafe" : ""%
- %classTag:"unsafe" == "true" ? "unsafe"%

Tagged value macros use the same naming convention as field substitution macros.

9.6.1.4.2.4 Function Macros

Function macros are a convenient way of manipulating and formatting various element data. Each function macro returns a result string. There are two primary ways to use the result of function macros:

- Direct substitution of the returned string into the output, such as: %TO_LOWER(attName)%
- Storing the returned string as part of a variable definition such as: \$name = %TO_LOWER(attName)%

Function macros can take parameters. Parameters may be passed to function macros as:

- string literals, enclosed within double quotation marks.
- direct substitution macros without the enclosing percent signs.
- variable references.
- numeric literals

Multiple parameters are passed using a comma separated list.

The available function macros are described below. Parameters are denoted by angle brackets, as in:
 FUNCTION_NAME(<param>)

CONVERT_SCOPE(<umlScope>)

Converts <umlScope> to the appropriate scope keyword for the language being generated. The following table shows the conversion of <umlScope> with respect to the given language.

Language	Package	Public	Private	Protected
C++	public	public	private	protected
C#	internal	public	private	protected
Delphi	protected	public	private	protected
Java		public	private	protected
PHP	public	public	private	protected
VB	Protected	Public	Private	Protected
VB .Net	Friend	Public	Private	Protected

The CONVERT_SCOPE macro is for use with supported languages.

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.

FIND(<src>, <subString>)

Position of the first instance of <subString> in <src>, -1 if none.

GET_ALIGNMENT()

Gives a string where all of the text on the current line of output is converted into spaces and tabs.

JAVADOC_COMMENT(<wrap_length>)

Converts the notes for the element currently in scope to javadoc style comments.

LEFT(<src>, <count>)

The first <count> characters of <src>.

LENGTH(<src>)

Length of <src>.

MID(<src>, <count>)

MID(<src>, <start>, <count>)

Substring of <src> starting at <start> and including <count> characters. Where count is omitted the rest of the string is included.

REMOVE_DUPLICATES(<source>, <separator>)

Where <source> is a <separator> separated list, this will remove any duplicate or empty strings.

REPLACE(<string>, <old>, <new>)

Replaces all occurrences of <old> with <new> in the given string <string>.

RESOLVE_OP_NAME()

Resolves clashes in interface names where two method from interfaces have the same name.

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(<text>, <wrap_length>, <start_string>)

Wraps <text> as designated to be <wrap_length>, adding <start_string> to the beginning of every line.

WRAP_COMMENT(<comment>, <wrap_length>, <indent>, <start_string>)

Wraps the text <comment> at width <wrap_length> putting <indent> and <start_string> at the beginning of each line.

Example:

```
$behaviour = %WRAP_COMMENT(opBehaviour, "40", " ", "//")%
```

Note that <wrap_length> must still be pass as a string, even though WRAP_COMMENT treats this parameter as an integer.

XML_COMMENT(<wrap_length>)

Converts the notes for the element currently in scope to xml style comments.

Function macros are named according to the All-Caps style, as in: %CONVERT_SCOPE (opScope)%

9.6.1.4.2.5 Control Macros

Control macros are used to control the processing and formatting of the templates. The basic types of control macros include:

- The list macro, for generating multiple elements, such as attributes and operations
- The branching macros, which form if-then-else constructs to conditionally execute parts of a template
- The PI macro for formatting newlines in the output
- The synchronization macros

In general, control macros are named according to Camel casing.

Lists

The list is used to generate multiple elements. The basic structure is:

```
%list=<TemplateName> @separator=<string> @indent=<string>%
```

where <string> is a double-quoted literal string and <TemplateName> may refer to one the following template names:

- Namespace
- Class
- ClassImpl
- Attribute
- InnerClass
- InnerClassImpl
- Operation
- OperationImpl
- Parameter
- ClassBase
- ClassInterface
- Custom Template, Custom templates allow the user to define their own templates more information is found in [Adding a Custom Template](#).

Example:

```
%list="Attribute" @separator="\n" @indent="  "%
```

The separator attribute, denoted above by @separator, specifies the space that should be used between each list item. This excludes the last item in the list.

The indent attribute, denoted by @indent, specifies the space by which each line in the generated output should be indented.

The above example would output the result of processing the Attribute template, for each attribute element of the class in scope. The resultant list would separate and indent its items by a single newline and two spaces respectively. If the class in scope had any stereotyped attributes, they would be generated using the appropriately specialised template.

There are some special cases to consider when using the list macro:

- If the Attribute template is used as an argument to the list macro, this will also generate attributes derived from associations by executing the appropriate LinkedAttribute template
- If InnerClass or InnerClassImpl is used as an argument to the list macro, these classes are generated using the Class and ClassImpl templates respectively. These arguments tell EA that it should process the templates based on the inner classes of the class in scope.

Branching (if-then-else Constructs)

The CTF supports a limited form of branching through the following macros:

- if
- elseif
- endif
- endTemplate

The basic structure of the if and elseif macros is:

```
%if <test> <operator> <test>%
```

where <operator> may be one of:

- ==
- !=

and <test> may be one of:

- a string literal, enclosed within double quotation marks.
- a direct substitution macro, without the enclosing percent signs
- a variable reference.

Branches may be nested, and multiple conditions may be specified using one of:

- and
- or

Note: When specifying multiple conditions, "and" and "or" have the same order of precedence and conditions are processed left to right.

The endif or endTemplate macros must be used to signify the end of a branch. In addition, the endTemplate macro causes the template to return immediately, if the corresponding branch is being executed.

Example:

```
%if 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 (\n, \t) as part of Literal Text
- Using the PI macro to format lines in the template that result in non-empty substitutions in the output

By default, each template line that generates a non-empty substitution also results in a newline being produced in the output. This behavior can be changed through the PI macro.

To demonstrate the use of the PI macro, consider the default C# Operation template:

```
%opTag:"Attribute"%
```

Default PI is \n, so any attributes would be on their own line

```
%PI=" "%
%opTag:"unsafe"=="true" ? "unsafe" : ""%
%CONVERT_SCOPE(opScope)%
%opTag:"new"=="true" ? "new" : ""%
%opAbstract=="T" ? "abstract" : ""%
%opConst=="T" ? "sealed" : ""%
%opStatic=="T" ? "static" : ""%
%opTag:"extern"=="true" ? "extern" : ""%
%opTag:"delegate"=="true" ? "delegate" : ""%
%opTag:"override"=="true" ? "override" : ""%
%opTag:"virtual"=="true" ? "virtual" : ""%
%opReturnType%%opReturnArray=="T" ? "[]" : ""%
%opStereotype=="operator" ? "operator" : ""%
%opName%(%list="Parameter" @separator=", "%)
```

Blank lines have no effect on the output

Set the PI, so keywords are separated by a space

Any keyword that does not apply, ie. the macro produces an empty result, will not result in a space

Only one space is generated for this line

The final line in the template will not generate a space

In the above example we want to arrange macros for the various keywords vertically for readability. In the output however, we want each relevant keyword to be separated by a single space. This is achieved by the line:

```
%PI=" "%
```

Notice how we do not need to specify the space between each of the possible keywords. This space is already implied by setting the PI to a single space. Essentially the PI acts as a convenience mechanism for formatting the output from within the templates.

The structure for setting the processing instruction is:

```
%PI=<value>%
```

where <value> may be a literal string enclosed by double quotes

The following points apply the PI macro:

- The value of the PI is not accessed explicitly
- Only template lines which result in a non-empty substitution cause the PI to be generated
- The last non-empty template line does not cause the PI to be generated
- The PI is not appended to the last substitution, regardless of which template line caused that substitution

The Synchronization Macros

The synchronization macros are used to provide formatting hints to EA when inserting new sections into the source code, during forward synchronization. The values for synchronization macros must be set in the File templates.

The structure for setting synchronization macros is:

```
%<name>=<value>%
```

where <name> may be one of the macros listed below and <value> is a literal string enclosed by double quotes.

Macro Name	Description
synchNewClassNotesSpace	Space to append to a new class note. Default value: \n
synchNewAttributeNotesSpace	Space to append to a new attribute note. Default value: \n
synchNewOperationNotesSpace	Space to append to a new operation note. Default value: \n
synchNewOperationBodySpace	Space to append to a new operation body. Default value: \n
synchNamespaceBodyIndent	Indent applied to classes within non-global namespaces. Default value: \t

9.6.1.4.3 Variables

Template variables provide a convenient way of storing and retrieving data within a template. The following topics explain how variables are defined and referenced.

See:

- [Variable Definitions](#)
- [Variable References](#)

9.6.1.4.3.1 Variable Definitions

Variable definitions take the basic form:

```
$<name> = <value>
```

where <name> may be any alpha-numeric sequence and <value> is derived from a macro or another variable

A simple example definition would be:

```
$foo = %className%
```

Variables may be defined, using values from:

- Substitution, function or list macros.
- String literals, enclosed within double quotation marks
- Variable references

The following rules apply to variable definitions:

- Variables have global scope within the template in which they are defined and are not accessible to other templates
- Each variable must be defined at the start of a line, without any intervening whitespace
- Variables are denoted by prefixing the name with \$, as in \$foo
- Variables do not need to be declared, prior to being defined
- Variables must be defined using either the assignment operator (=), or the addition-assignment operator (+=)
- Multiple terms may be combined in a single definition using the addition operator (+)

Examples

Using a substitution macro:

```
$foo = %opTag:"bar"%
```

Using a literal string:

```
$foo = "bar"
```

Using another variable:

```
$foo = $bar
```

Using a list macro :

```
$ops = %list="Operation" @separator="\n\n" @indent="\t"%
```

Using the addition-assignment operator (+=) :

```
$body += %list="Operation" @separator="\n\n" @indent="\t"%
```

The above definition is equivalent to the following:

```
$body = $body + %list="Operation" @separator="\n\n" @indent="\t"%
```

Using multiple terms:

```
$templateArgs = %list="ClassParameter" @separator=", "%
$template = "template<" + $templateArgs + ">"
```

9.6.1.4.3.2 Variable References

Variable values may be retrieved by using a reference of the form:

```
$<name>
```

where <name> may refer to a previously defined variable.

Variables references may be used in one of the following ways:

- As part of a macro, such as the argument to a function macro
- As a term in a variable definition
- As a direct substitution of the variable value into the output

Note: It is legal to reference a variable before it is defined. In this case, the variable is assumed to contain an empty string value: ""

Example:

Using variables as part of a macro. The following is an excerpt from the default C++ ClassNotes template.

```
$wrapLen = %genOptWrapComment%
$style = %genOptCPPCommentStyle%

%if $style == "XML.NET"%
%XML_COMMENT($wrapLen)%
%else%
%CSTYLE_COMMENT($wrapLen)%
%endif%
```

Define variables to store the style and wrap length options

Reference to \$style as part of a condition

Reference to \$wrapLen as an argument to function macro

Example:

Using variable references as part of a variable definitions:

```
$foo = "foo"
$bar = "bar"

$foobar = $foo + $bar
```

Define our variables

\$foobar now contains the value "foobar"

Example:

Substituting variable values into the output

```

$bases=%ClassInherits%
...
class %className%$bases

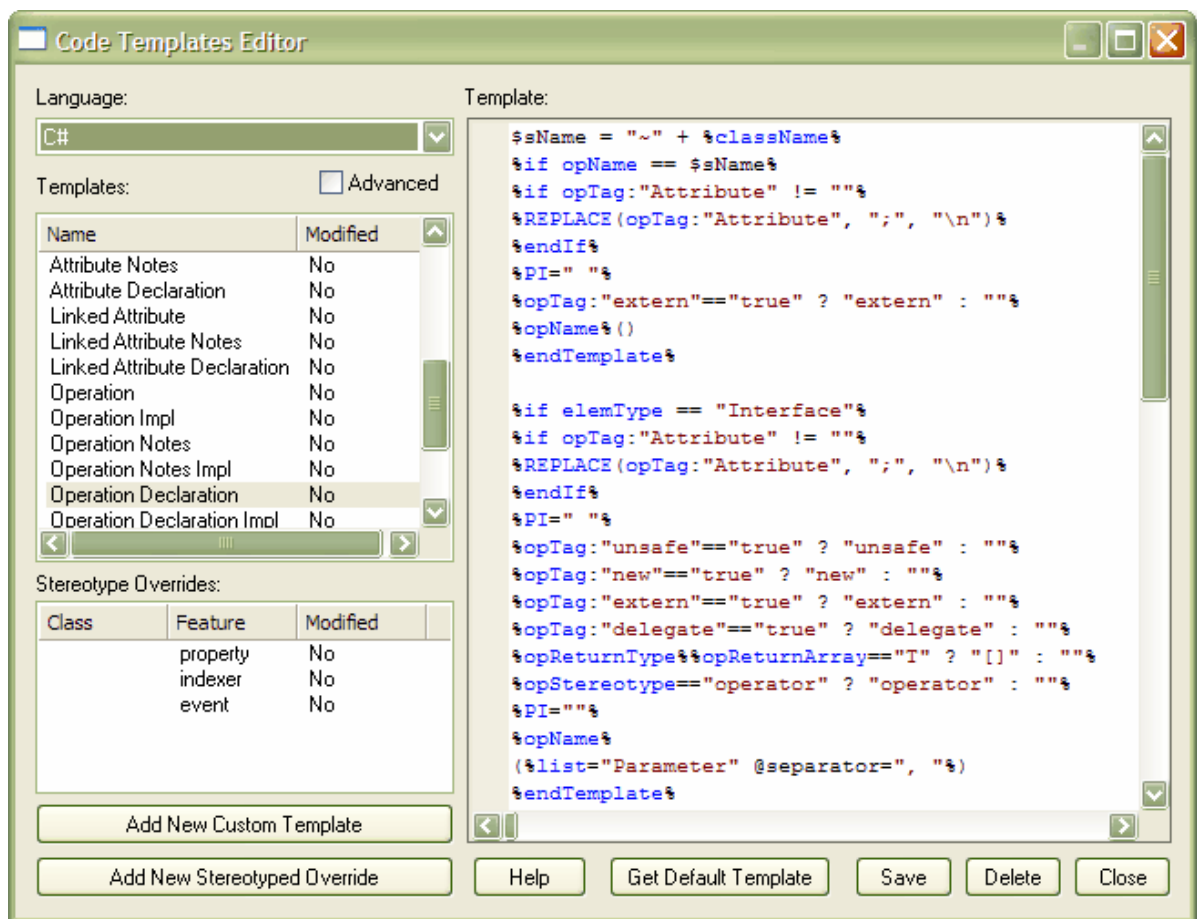
```

Store the result of the *ClassInherits* template in *\$bases*

Now output the value of *\$bases* after the class name

9.6.2 The Code Template Editor

The *Code Templates Editor* is accessed by selecting *Code Generation Templates* from the *Configuration* menu.



Control	Description
Language	Selects the programming language.
Template	Displays the contents of the active template. Provides the editor for modifying templates.
Templates	Lists the base code templates. The active template is highlighted. The modified field indicates whether the user has changed the default template for the current language.
Stereotype Overrides	Lists the stereotyped templates, for the active base template. The modified field indicates whether a default stereotyped template has been modified by the user.
Add New Stereotyped Override	Invokes a dialog for adding a stereotyped template, for the currently selected base template.
Add New Custom Template	Invokes a dialog for creating a custom stereotyped template.
Get Default Template	Updates the editor display with the default version of the active template.
Save	Overwrites the active templates with the contents of the editor.
Delete	If the active template has been overridden by the user, the override is deleted and replaced by the corresponding default code template.
Close	Exits the Code Template Editor dialog.

See:

- [Overriding Default Templates](#)
- [Adding New Stereotyped Templates](#)
- [Creating Templates For Custom Languages](#)
- [Importing and Exporting Code Templates](#)

9.6.2.1 Overriding Default Templates

EA has a set of built-in or default code generation templates. The *Code Templates Editor* allows users to modify these default templates, hence customizing the way in which EA generates code. Users may choose to modify any or all of the base templates to achieve their desired coding style.

Any templates that are overridden by the user are stored in the .EAP file. When generating code, EA first checks whether a template has been modified and if so, uses that template. Otherwise the appropriate default template is used.

Override a Default Template

To override a default code generation template, follow the steps below.

1. Select *Code Generation Templates* from the *Configuration* menu to open the *Code Templates Editor*.
2. Select the appropriate language from the *Language* list.
3. Select one of the base templates from the *Templates* list.
4. If the base template has stereotyped overrides, you may select one of these from the *Stereotype Overrides* list.
5. In the *Template* editor, make the desired modifications.
6. Press *Save*. This stores the modified version of the template to the .EAP file. The template will be marked as modified.

When generating code, EA will now use the overridden template, instead of the default template.

9.6.2.2 Adding New Stereotyped Templates

Sometimes it is useful to define a specific code generation template for use with elements of a given stereotype. This allows different code to be generated for elements, depending on their stereotype. EA provides some default templates, which have been specialised for commonly used stereotypes in supported languages. For example the Operation Body template for C# has been specialized for the property stereotype, so as to automatically generate its constituent get and set methods. Users may override the default stereotyped templates as described in the previous section. Additionally users may define templates for their own stereotypes, as described below.

Add A New Stereotyped Template

To override a default code generation template, follow the steps below.

1. Select *Code Generation Templates* from the *Configuration* menu to open the *Code Templates Editor*.
2. Select the appropriate language, from the *Language* list.
3. Select one of the base templates, from the *Templates* list.
4. Press *Add New Stereotyped Override*.
5. This opens the *New Template Override* dialog.



6. Select the desired *Feature* and/or *Class* stereotype and press *OK*.
7. The new stereotyped template override will appear in *Stereotype Overrides* list and be marked as modified.
8. Make the desired modifications in the *Template* editor.
9. Press *Save*. This stores the new stereotyped template in the .EAP file.

EA will now use the stereotyped template, when generated code for elements of that stereotype.

Note that class and feature stereotypes can be combined to provide a further level of specialization for features. For example if properties should be generated differently when the class has a stereotype "MyStereotype", then both "property" and "MyStereotype" should be specified in the *New Template Override* dialog.

9.6.2.3 Creating Templates For Custom Languages

EA can forward generate code for languages that it does not specifically support, if the appropriate code generation templates are defined for that language. This section outlines the steps required to define templates for custom languages.

Define a Template for a Custom Language

1. Create the custom language as a new product. To do this, go to the Language Datatypes dialog (select *Language Datatypes* from the *Configuration* menu), enter the name of the new language in the *Product Name* field and enter a datatype (one is enough to declare that the new language exists). See [Data Types](#) for more details.
2. Select *Code Generation Templates* from the *Configuration* menu to open the *Code Templates Editor*.
3. Select the custom language, from the *Language* list.
4. Select one of the base templates, from the *Templates* list.
5. Define the template using the *Template* editor.
6. Press *Save*. This stores the template in the .EAP file.
7. Repeat steps 1 to 6 for each of the relevant base templates for the custom language.

Note: The File template must be defined for the custom language. The File template can then refer to the Import Section, Namespace and Class templates.

9.6.2.4 Importing and Exporting Code Templates

User defined Code Templates can be imported and exported as Reference Data (see [Import and Export Reference Data](#)). The templates defined for each language appear in list of tables with the language name suffixed with "_Code_Templates".

9.6.3 Synchronizing Code

EA uses code templates during the forward synchronization of the following programming languages:

- C++
- C#
- Delphi
- Java
- VB
- VB.Net

Only a subset of the code templates are used during synchronization. This subset corresponds to the distinct sections that EA recognises in the source code. The following table lists the code templates and their corresponding code sections, which may be synchronized.

Code Template	Code Section
Class Notes	Comments preceding class declaration
Class Declaration	Up to and including, class parents
Attribute Notes	Comments preceding Attribute declaration
Attribute Declaration	Up to and including terminating character
Operation Notes	Comments preceding operation declaration
Operation Notes Impl	As for "Operation Notes"
Operation Declaration	Up to and including, terminating character
Operation Declaration Impl	Up to and including terminating character
Operation Body	Everything between and including the braces
Operation Body Impl	As for "Operation Body"

Three types of changes can occur in the source when it is synchronized with the UML model:

- [Synchronizing Existing Sections](#): for example, changing the return type in an operation declaration.
- [Adding New Sections to Existing Features](#): for example, adding notes to a class declaration, where there were previously none.
- [Adding New Features and Elements](#): for example, adding a new operation to a class.

Each of these changes must be handled differently by EA and their effect on the CTF is described in the following sections.

9.6.3.1 Synchronizing Existing Sections

When an existing section in the source code differs from the result generated by the corresponding template, that section is replaced. Consider for example, the following C++ class declaration:

```
[asm] class A : public B
```

Now assume we add an inheritance relationship from class A to class C, the entire class declaration would be replaced with something like:

```
[asm] class A : public B, public C
```

9.6.3.2 Adding New Sections to Existing Features

The following can be added as new sections, to existing features in the source code:

- Class Notes
- Attribute Notes
- Operation Notes
- Operation Notes Impl
- Operation Body
- Operation Body Impl

Assume class "A" from the previous example had no note when we originally generated the code. Now assume that we specify a note in the model for class "A." EA will attempt to add the new note from the model

during synchronization. It will do this by executing the Class Notes template.

To make room for the new section to be inserted, we can specify how much whitespace to append to the section via synchronization macros. These macros are described in the section: [Control Macros](#).

9.6.3.3 Adding New Features and Elements

The following features and elements can be added to the source code during synchronization:

- Attributes
- Inner Classes
- Operations

These are added by executing the relevant templates for each new element or feature in the model. EA attempts to preserve the appropriate indenting of new features in the code, by finding the indents specified in list macros of the Class. For languages that make use of namespaces, the `synchNamespaceBodyIndent` macro is available. Classes defined within a (non-global) namespace will be indented according to the value set for this macro, during synchronisation. This value is ignored for classes defined within a package setup as a root namespace, or if the Generate Namespace option is turned off.

9.7 XML Schema Generation

The XSD Generation facility converts a UML class model to a W3C XML Schema (XSD). This allows Data Modelers to start working at a conceptual level in UML, leaving the tedious aspects of XSD creation to EA. The schema generation can then be customized if necessary, by using the provided "UML Profile for XML" as described later.

An XML schema corresponds to a UML package. Therefore the XSD generation is a package-level operation in EA. To demonstrate the usage of the schema generator, we begin with an example model.

See:

- [Getting Started](#)
- [Steps to Generate XSD](#)
- [Example](#)
- [UML Profile for XML](#)

9.7.1 Getting Started

To use the schema generation facility you will require the following:

1. EA Professional or Corporate edition.
2. [XSDDataTypes Package](#): This package contains classes representing XSD primitive data types. This package is available as an XMI file. To import the file as a UML Package, use EA's [XML import](#) facility which is available from the menu item : *Project | Import/Export | Import Package from XMI*.
3. [UML Profile for XML](#): This resource file contains the stereotyped classes which allow the schema generation to be customized. The UML Profile for XML can be imported into a model using the Resource View (see [Using Profiles](#) for details on importing UML profiles into EA).

9.7.2 Steps to Generate XSD

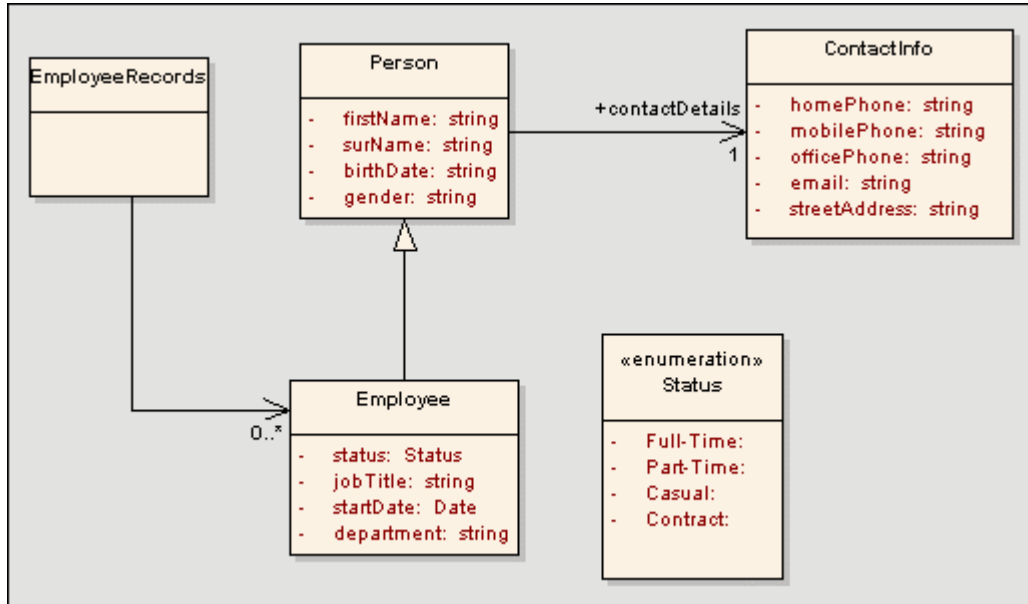
Follow the steps below to generate XSD:

1. In the Project Browser, right click on the package to be converted to XSD to open the context menu.
2. Select *Generate XML Schema* from the *Code Engineering* submenu.
3. Set the desired output file using the *Filename* field.
4. Set the desired xml encoding using the *Encoding* field.
5. Press *Generate* to generate the schema.
6. The progress of the schema generator will be shown in the *Progress* edit box.

Tip: The *Generate XML Schema* dialog may also be accessed for the active diagram by selecting *Generate XML Schema* from the *Project* menu.

9.7.3 Example

The Class diagram below models a simple "Employee Details" system, intended to store a company's employee contact information. The classes shown form the "EmployeeDetails" package. The UML attributes of the classes map directly to XML elements or attributes. Note that the classes have no methods, since there is no meaningful correspondence between class methods and XSD constructs.



The following figure shows the schema which is generated for the Employee Details package by default. Notice how each UML class corresponds to a complexType definition in the schema. The class attributes are generated as schema elements contained in a "sequence" model group within the definition. The enumeration class is the exception here- it maps directly to an XSD enumeration, contained within a simpleType definition.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Person" type="Person"/>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="Person.firstName" type="xs:string"/>
      <xs:element name="Person.surName" type="xs:string"/>
      <xs:element name="Person.birthDate" type="xs:string"/>
      <xs:element name="Person.gender" type="xs:string"/>
      <xs:element name="Person.contactDetails">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ContactInfo"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Employee" type="Employee"/>
  <xs:complexType name="Employee">
    <xs:complexContent>
      <xs:extension base="Person">
        <xs:sequence>
          <xs:element name="Employee.status" type="Status"/>
          <xs:element name="Employee.jobTitle" type="xs:string"/>
          <xs:element name="Employee.startDate" type="xs:date"/>
          <xs:element name="Employee.department" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ContactInfo" type="ContactInfo"/>
  <xs:complexType name="ContactInfo">
    <xs:sequence>
      <xs:element name="ContactInfo.homePhone" type="xs:string"/>
      <xs:element name="ContactInfo.mobilePhone" type="xs:string"/>
      <xs:element name="ContactInfo.officePhone" type="xs:string"/>
      <xs:element name="ContactInfo.email" type="xs:string"/>
      <xs:element name="ContactInfo.streetAddress" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="Status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Full-Time"/>
      <xs:enumeration value="Part-Time"/>
      <xs:enumeration value="Casual"/>
      <xs:enumeration value="Contract"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="EmployeeRecords" type="EmployeeRecords"/>
  <xs:complexType name="EmployeeRecords">
    <xs:sequence>
      <xs:element name="EmployeeRecords.Employee">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="Employee" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

The following table describes the default mapping of UML constructs to XSD constructs. The next section describes a UML Profile for XML which allows this default mapping to be altered to suit individual needs. Using the profile, we will alter our example class model to tailor the generation of the XML schema.

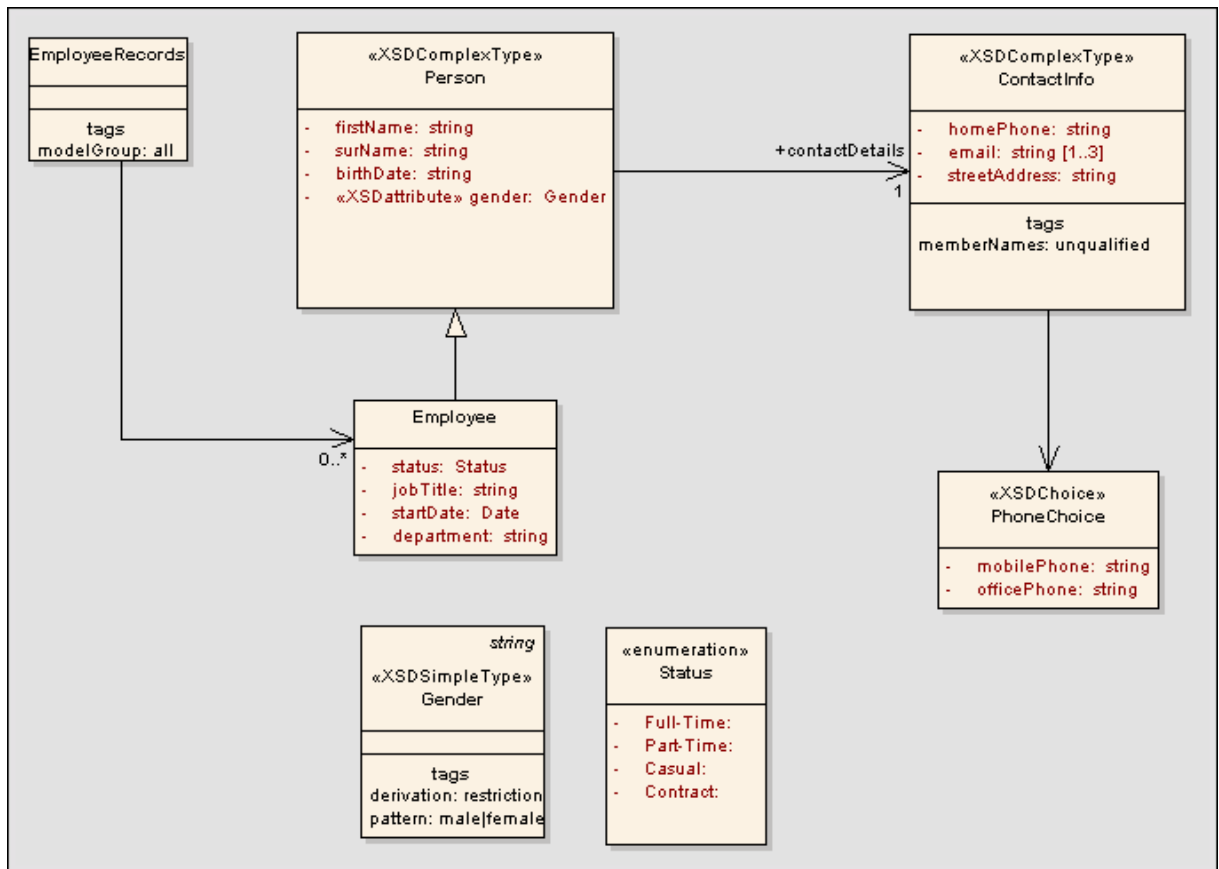
UML Construct	Default XSD Production Rules
Package	<p>A schema element is generated for the target package. If the target package includes classes from another package, which has the tagged values "targetNamespace" and "targetNamespacePrefix" set, these are included as attributes of the schema element.</p> <p>In addition, an import or include element is created for each referenced package. (An include element is used if the external package shares the same targetNamespace tagged value as the target package. An import element is used where the targetNamespaces differ).</p>
Class	A root-level element declaration and complexType definition are generated. The element name and type are the same as the class name. An XSD sequence model group is generated to contain UML attributes generated as elements.
Attribute	An element is declared for each class attribute. The element name is set to that of the UML attribute name. This is prepended with the class name to make the element unique. The minOccurs and maxOccurs attributes are set to reflect the attribute cardinality. (Note: If left unspecified, minOccurs and maxOccurs default to 1.) If the attribute refers to another class, the element declaration is followed a complexType definition, which contains a reference to the appropriate complexType.
Association	An element is declared for each of association owned by a class. The element name is set to that of the association role. The minOccurs and maxOccurs reflect the cardinality of the association. Note: if the direction of the association is unspecified, the owner is assumed to be the source.
Generalization (Inheritance)	For single inheritances, an extension element is generated with the base attribute set to the base class name. The UML attributes of the child class are then appended to an all model group within the extension element.
<<enumeration>> (stereotype)	A simpleType element is declared for the enumeration class with the name attribute set to the class name. A restriction element is generated with base set to string . Each of the class attributes are appended to the restriction element as XSD enumeration elements with value set to the UML attribute name. Any type specification for the UML attributes will be ignored by the schema generator.

9.7.4 UML Profile for XML

The UML profile for XML specifies a set of **stereotypes**, **tagged values** and **constraints** which may be applied to the UML model in order to change particular aspects of the resulting schema. For example, we may wish to have certain UML class attributes converted to XSD attributes or, we may need to use a different model group than the default "sequence".

The stereotype explicitly tells the generator to which XSD structure the UML construct maps. The tagged values further define aspects of the mapping, such as whether the elements should be qualified. The constraints define any conditions that must be satisfied for the stereotype to apply.

To demonstrate how the profile for XML can affect the schema generation, refer to the following class diagram - an adaptation of our previous example. Note the changes in the corresponding schema.



```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="Gender">
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Person" type="Person"/>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="Person.surName" type="xs:string"/>
      <xs:element name="Person.firstName" type="xs:string"/>
      <xs:element name="Person.birthDate" type="xs:string"/>
      <xs:element name="Person.contactDetails">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ContactInfo"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attributeGroup ref="Person.att"/>
  </xs:complexType>
  <xs:attributeGroup name="Person.att">
    <xs:attribute name="Person.gender" type="Gender"/>
  </xs:attributeGroup>
  <xs:element name="Employee" type="Employee"/>
  <xs:complexType name="Employee">
    <xs:complexContent>
      <xs:extension base="Person">
        <xs:sequence>
          <xs:element name="Employee.status" type="Status"/>
          <xs:element name="Employee.jobTitle" type="xs:string"/>
          <xs:element name="Employee.startDate" type="xs:date"/>
          <xs:element name="Employee.department" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ContactInfo" type="ContactInfo"/>
  <xs:complexType name="ContactInfo">
    <xs:sequence>
      <xs:element name="homePhone" type="xs:string"/>
      <xs:element name="email" type="xs:string" maxOccurs="3"/>
      <xs:element name="streetAddress" type="xs:string"/>
      <xs:choice>
        <xs:element name="mobilePhone" type="xs:string"/>
        <xs:element name="officePhone" type="xs:string"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="Status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Full-Time"/>
      <xs:enumeration value="Part-Time"/>
      <xs:enumeration value="Casual"/>
      <xs:enumeration value="Contract"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="EmployeeRecords" type="EmployeeRecords"/>
  <xs:complexType name="EmployeeRecords">
    <xs:all>
      <xs:element name="EmployeeRecords.Employee">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="Employee" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:all>
  </xs:complexType>
</xs:schema>

```

In particular:

- We have explicitly set the stereotype for the Person and ContactInfo classes to **XSDcomplexType**. EmployeeRecords uses the **modelGroup** tagged value to use the *all* construct instead of the default *sequence*. The ContactInfo also uses the **memberNames** tag to prevent the attribute's element names being prefixed with the class name.
- The **XSDattribute** stereotype causes the generation of an XSD *attribute* (instead of the default *element*), for the gender attribute of class Person.
- To restrict the possible values for the gender attribute to "male" or "female", we have created the Gender class. Class "Gender" is stereotyped **XSDsimpleType** and uses the **pattern** tagged value to restrict its possible values. We could also have used regular expressions in this way to restrict values for the "birthDate", "email" and phone number attributes.
- The use of the **XSDchoice** stereotype allows us to generate the "mobilePhone" and "officePhone" attributes as options within an XSD *choice* modelGroup. Note how the memberNames of attributes contained by this modelGroup are unqualified as determined by its containing complexType class. (This occurs for each owner of the modelGroup, where the multiple associated classes.)
- Less obvious from the diagram is the ordering of the attributes for the Person class. Notice how the "firstName" and "surName" attributes have had their position swapped in the schema. This is because the **position** tagged value has been used for the attribute of the Person class in the modified example.

The following table details the features of the UML Profile for XML. Tagged value names are shown in bold followed by the allowed values. If there is a default value used by EA's schema generator it is underlined.

<<XSDschema>>

UML Construct	Package; Component
Description	All classes in a package or component are defined within one schema. This stereotype can be used to specify schema-wide settings.
Tagged Values	
anonymousRole: (true false)	Specifies if the role name is included in the element declaration for the UML attribute.
anonymousType: (true false)	Specifies whether the class type is anonymous for attributes.
attributeFormDefault: (qualified unqualified)	Determines whether attribute instances must be qualified.
defaultNamespace:	The default namespace used in this schema. This value is used to specify the default namespace attribute (xmlns=), in the schema element.
elementDerivation: (true false)	Determines whether inheritances are generated using XSD extension or copy-down inheritance.
elementFormDefault: (qualified unqualified)	Determines whether element instances must be qualified.
memberNames: (qualified unqualified)	Determines whether elements generated from Class attributes have their name qualified by the corresponding class name.
modelGroup: (all sequence choice)	Specifies the default XSD model group used to generate complexType definitions.
schemaLocation:	The URI which identifies the location of the schema. This value is used in the import and include elements.
targetNamespace:	The URI which uniquely identifies this schema's namespace.
targetNamespacePrefix:	The prefix which abbreviates the targetNamespace.
version:	The version of this schema.
Constraints	None.

<<XSDcomplexType>>

UML Construct	Class
Description	complexType definitions are created for generic UML classes. This stereotypes helps tailor the generation of a complexType definition
Tagged Values	
memberNames: (qualified unqualified)	Determines whether elements generated from the UML class attributes and associations have their name qualified by the corresponding class name for this complexType definition.
mixed: (true false)	Determines whether this element may contain mixed element and character content. Refer to the W3C XML Schema recommendation.
modelGroup: (all sequence choice)	Overrides the default XSD model for generating this complexType definition.
Constraints	None.

<<XSDsimpleType>>

UML Construct	Class
Description	An XSD simpleType is generated for classes with this stereotype.
Tagged Values	
derivation: (<u>restriction</u> list)	Specifies the derivation of the simpleType. Refer to the W3C XML Schema recommendation.
length:	Refer to the W3C XML Schema recommendation.
minLength:	Refer to the W3C XML Schema recommendation.
maxLength:	Refer to the W3C XML Schema recommendation.
pattern:	Refer to the W3C XML Schema recommendation.
Constraints	This class can only participate in an inheritance relation with another simpleType. It cannot have any attributes or own any associations. They will be ignored if present.

<<XSDsequence>>

UML Construct	Class
Description	<p>The schema generator creates a sequence model group as the container for the attributes and associations owned by this class. The model group is in turn added to the model groups of this class' respective owners.</p> <p>Note: Tagged values specified by owners of this class persist through to the child elements of this model group. Thus if memberNames are unqualified for a complexType, so will be the children of this model group when added to that complexType.</p>
Tagged Values	None.
Constraints	This class must be the destination of unidirectional associations. If it is not, this class and its connectors are ignored, possibly invalidating other model group classes. Inheritance relations are ignored for this class.

<<XSDchoice>>

UML Construct	Class
Description	Creates an XSD choice element. Refer to XSDsequence for more details.
Tagged Values	None.
Constraints	As for XSDsequence.

<<XSDelement>>

UML Construct	Attribute; AssociationEnd
Description	By applying this stereotype to a UML class attribute or AssociationEnd, the corresponding UML entity is generated as an element within the parent complexType and not as an XSD attribute.
Tagged Values	
form: (qualified unqualified)	Overrides the schema's elementFormDefault value.
position:	Causes the elements to be ordered within a sequence model group of the containing complexType. Duplicated and invalid position tagged values are ignored and result in undefined ordering of the UML attributes. Missing position values cause the defined positions to be allocated as specified, with the remaining elements filling the missing positions in an undefined order.
anonymousRole: (true <u>false</u>)	Specifies if the role name is included in the element declaration for the UML attribute
anonymousType: (true <u>false</u>)	Specifies whether the class type is anonymous for attributes.
Constraints	None.

<<XSDataAttribute>>

UML Construct	Attribute; AssociationEnd
Description	By applying this stereotype to a UML class attribute or AssociationEnd, the corresponding UML entity is generated as an XSD attribute within the parent complexType and not as an XSD element.
Tagged Values	
form: (qualified unqualified)	Overrides the schema's attributeFormDefault value.
use: (prohibited <u>optional</u> required)	Refer to the W3C XML Schema recommendation.
default:	Refer to the W3C XML Schema recommendation.
fixed:	Refer to the W3C XML Schema recommendation.
Constraints	The attribute datatype should not refer to a class specification, it will be ignored otherwise.

<<XSDany>>

UML Construct	Class; Attribute
Description	If applied to a UML attribute, an XSD anyAttribute element is generated. If applied to a UML class, an XSD any element is generated.
Tagged Values	
namespace:	Refer to the W3C XML Schema recommendation.
processContents: (skip lax <u>strict</u>)	Refer to the W3C XML Schema recommendation.
Constraints	None.

<<XSDrestriction>>

UML Construct	Generalization
Description	Overrides the default use of XSD extension for inheritance and generates the child as a complexType with a restriction element instead.
Tagged Values	None.
Constraints	Applies only to UML class parent-child relations.

Part



10 Data Modeling

Database Modeling

Database modeling and database design are not explicitly covered by the UML Specification, but they may be achieved in Enterprise Architect using the **UML Data Modeling Profile**. This profile provides easy to use and easy to understand extensions to the UML standard, mapping the database concepts of tables and relationships onto the UML concepts of classes and associations. These extensions also allow you to model database keys, triggers, constraints, RI and other relational database features.

Supported Databases

EA supports import of database schema from these databases...

- DB2
- InterBase
- MS Access
- MySQL
- Oracle
- PostgreSQL
- MS SQL Server
- Sybase Adaptive Server Anywhere
- Firebird

Typical Tasks

Typical tasks you might want to perform when modeling or designing databases include:

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

Note: The **UML Data Modeling Profile** is not currently a ratified standard; however it has wide industry support and is a useful method for bridging the gap between the UML and conventional relational database modeling.

Note: Firebird 1.5 database tables can be modelled and generated as InterBase tables. Firebird tables can be imported but are treated as InterBase tables.

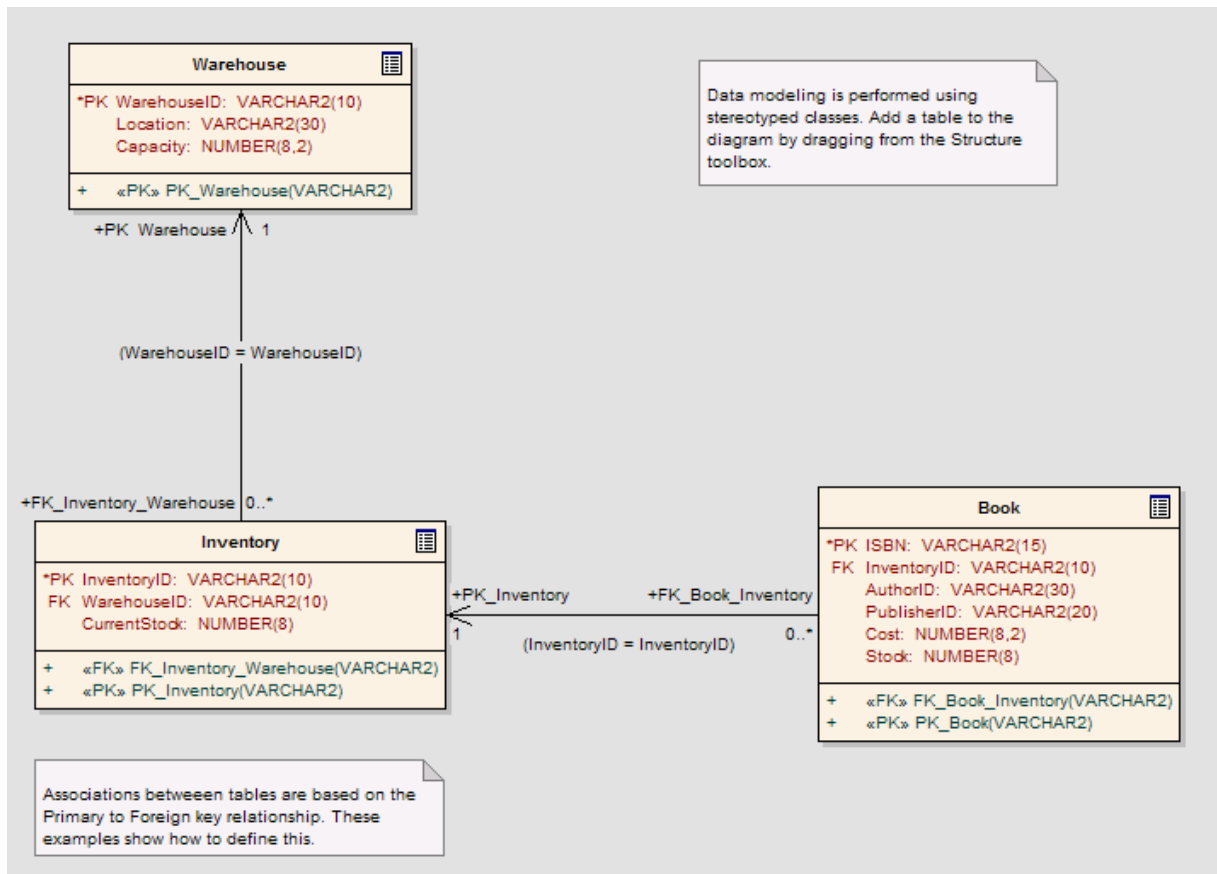
10.1 A Data Model Diagram

The diagram below is an example of a Data Model diagram. It shows three tables which are linked on primary to foreign key pairs with associated multiplicity.

Note also the use of stereotyped operations for Primary (PK) and Foreign (FK) keys. Operations could also be added for:

- Validation (check)
- Triggers

- Constraints
- Indexes



A Data Model diagram is represented in Enterprise Architect as a Class diagram, and is created [in exactly the same way](#) as other diagrams.

10.2 Create a Table

What is a Table?

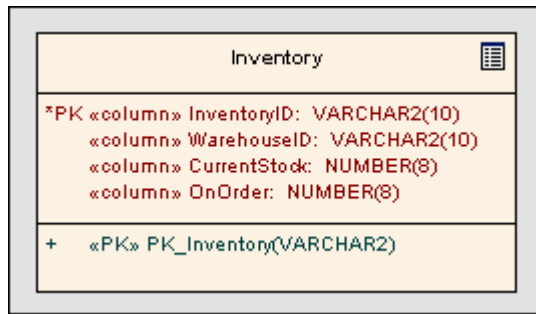
The basic modeling structure of a relational database is the Table. A table represents a set of records, or rows, with the same structure.

The *UML Data Modeling Profile* represents a table as a stereotyped class - that is, a class element with a stereotype of "table" applied to it. A pictorial table icon is shown in the upper right corner of the image when it is shown on a data model diagram.

Create a Table

To create a table, follow the steps below:

1. Select a diagram.
2. Open the *Structure* group on the UML Toolbox.
3. Left click on the *Table* element in the list of elements and then left click on the diagram.
4. Give the table a name and [set any other properties](#) as required.

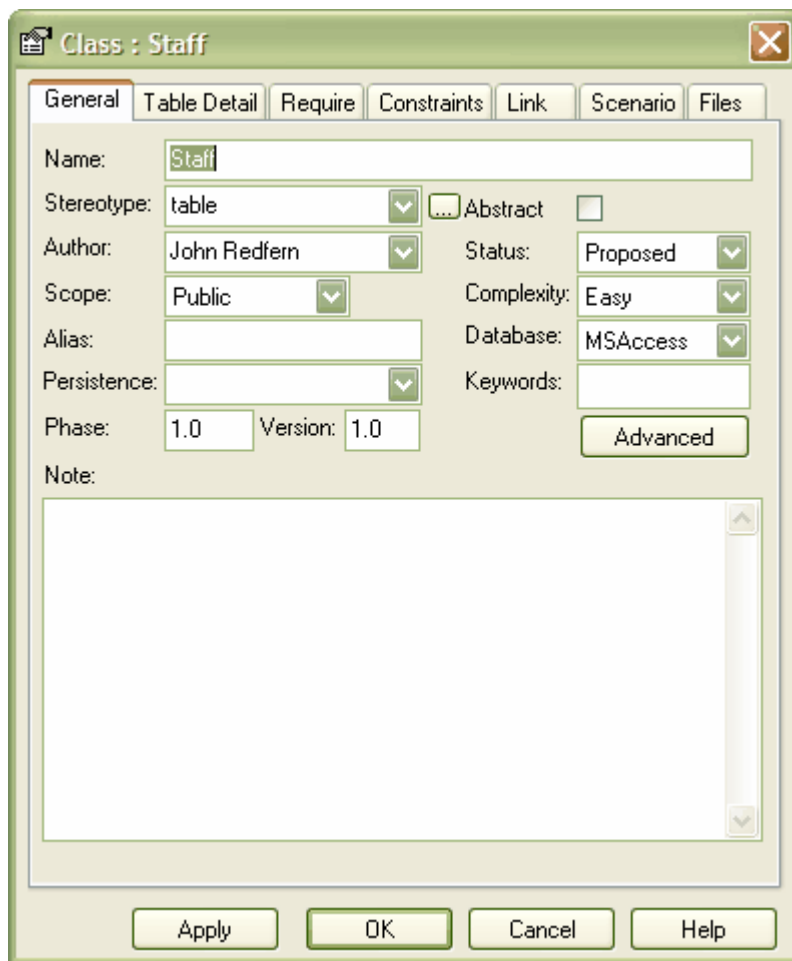


10.3 Set Table Properties

Once you have created your table, you may set its properties. Most table properties can be set from the *Properties* dialog, as described below. Some properties though will need to be entered as tagged values, and these are described elsewhere, i.e. setting the value of the [Table Owner](#), and for MySQL databases setting the [Table Type](#).

Set the Database type

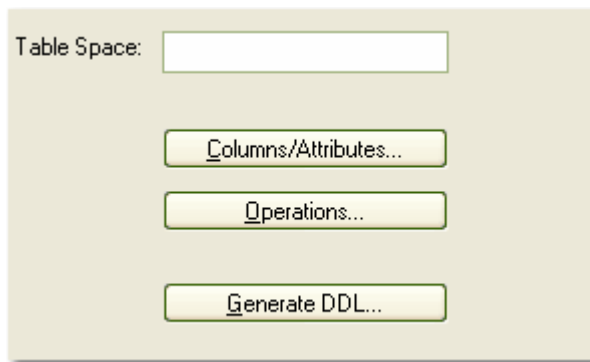
The most important property to set for a table (after its name) is the *Database* type. This defines the list of datatypes that will be available for defining columns, and also declares which dialect of DDL will be generated. Enterprise Architect supports the following databases: DB2, InterBase, MS Access, MySQL, Oracle, PostgreSQL, SQL Server 2000, SQLServer7 and Sybase.



To set the Database type, follow the steps below:

1. Double-click on the table element in a diagram to open the *Properties* dialog.
2. Select the *General* tab.
3. Select the Database type from the *Database* dropdown list.
4. Press *OK* to save changes.

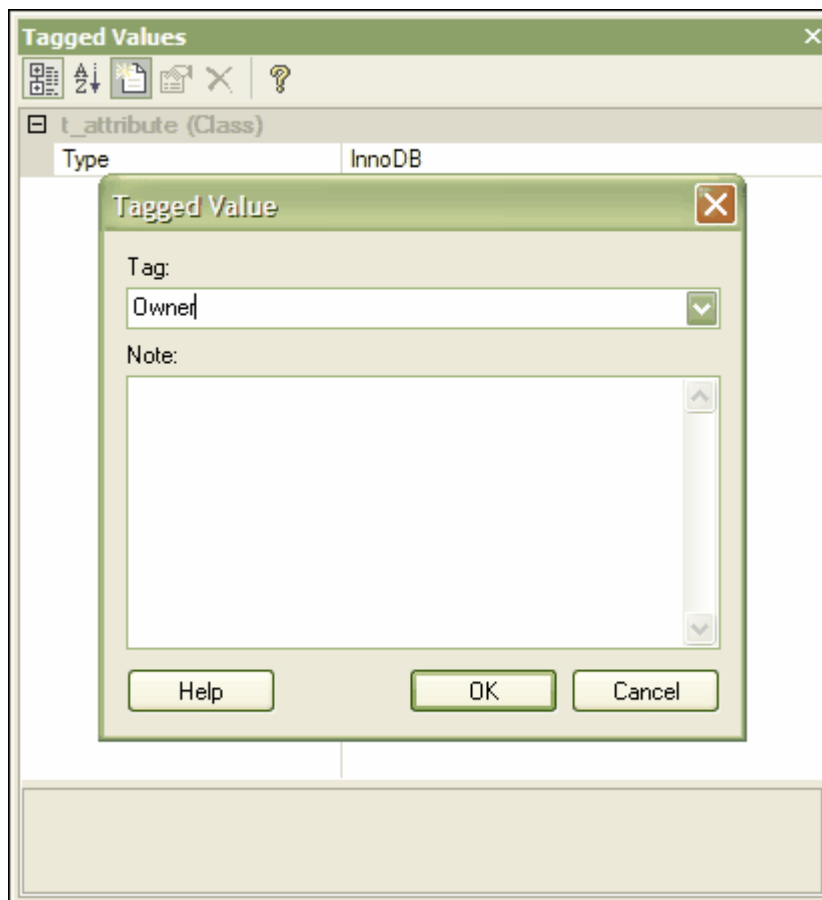
By clicking on the *Table Detail* tab on this dialog, you can then go to the [Columns dialog](#) or the [Operations dialog](#) or you can [Generate DDL](#) for this table.



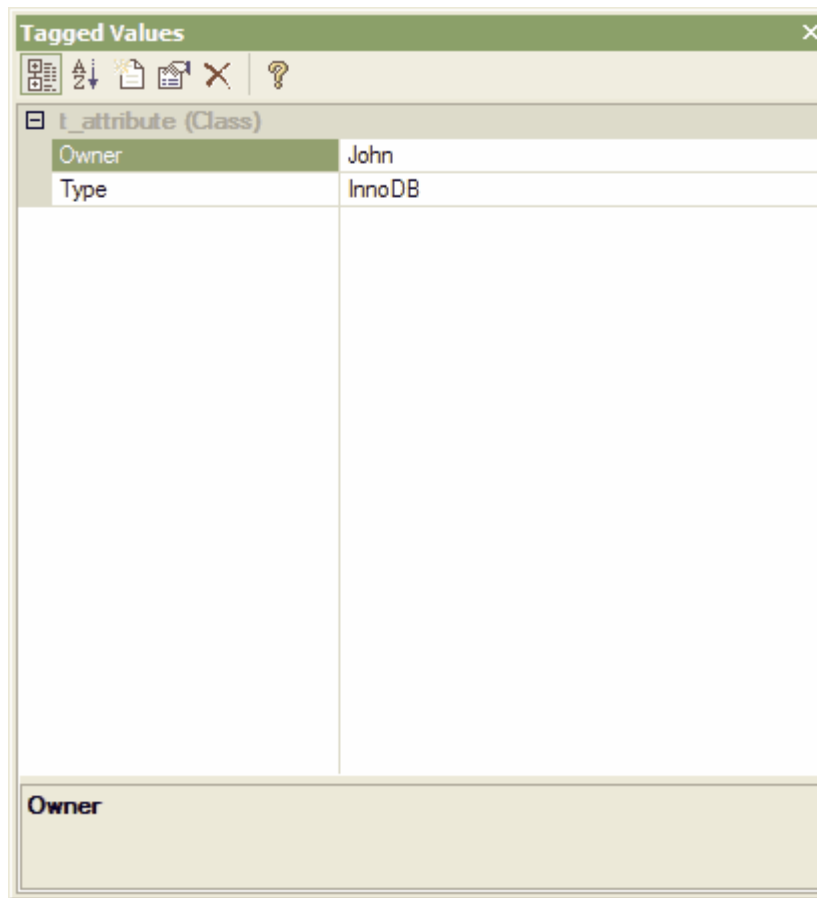
10.3.1 Set Table Owner

To define the owner of a table, follow the steps below:

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the table by clicking on it in a diagram or the *Project View* window. The *Tagged Values* window will now show the tags for the table selected.
3. Press the *New Tag* button.
4. Define a tag named "**Owner**" in the *Tag* field. Add a comment in the *Note* field, if required.



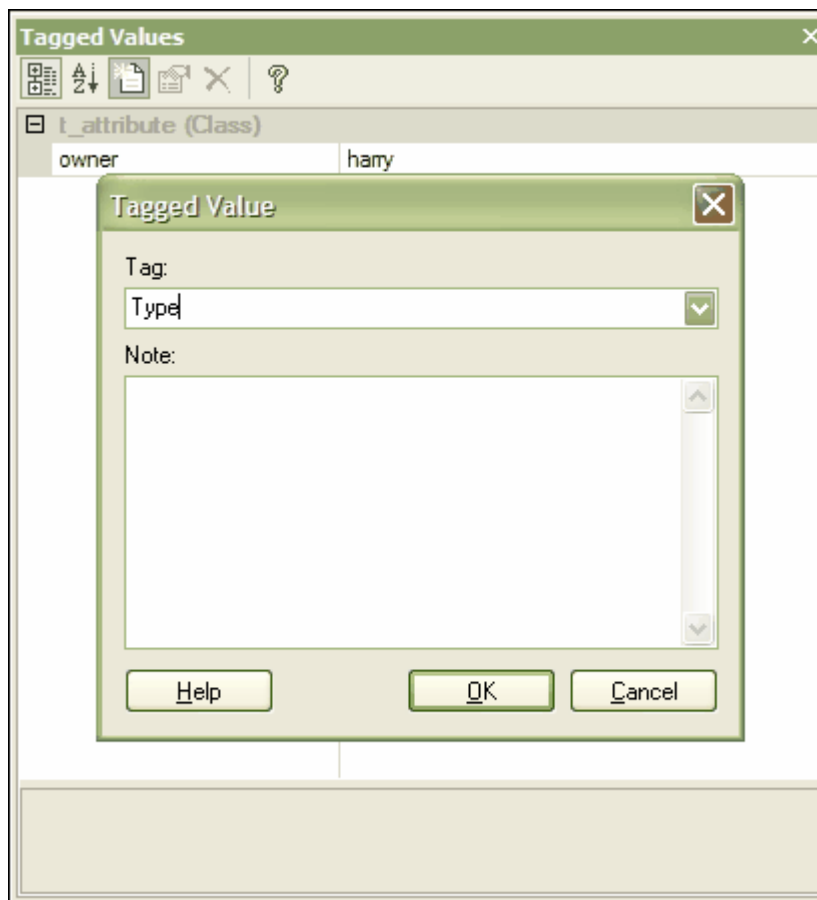
5. Press **OK** to confirm the operation.
6. In the *Tagged Values* Window enter a value for the "**Owner**" tag. Generated DDL will include the table owner in the SQL script.



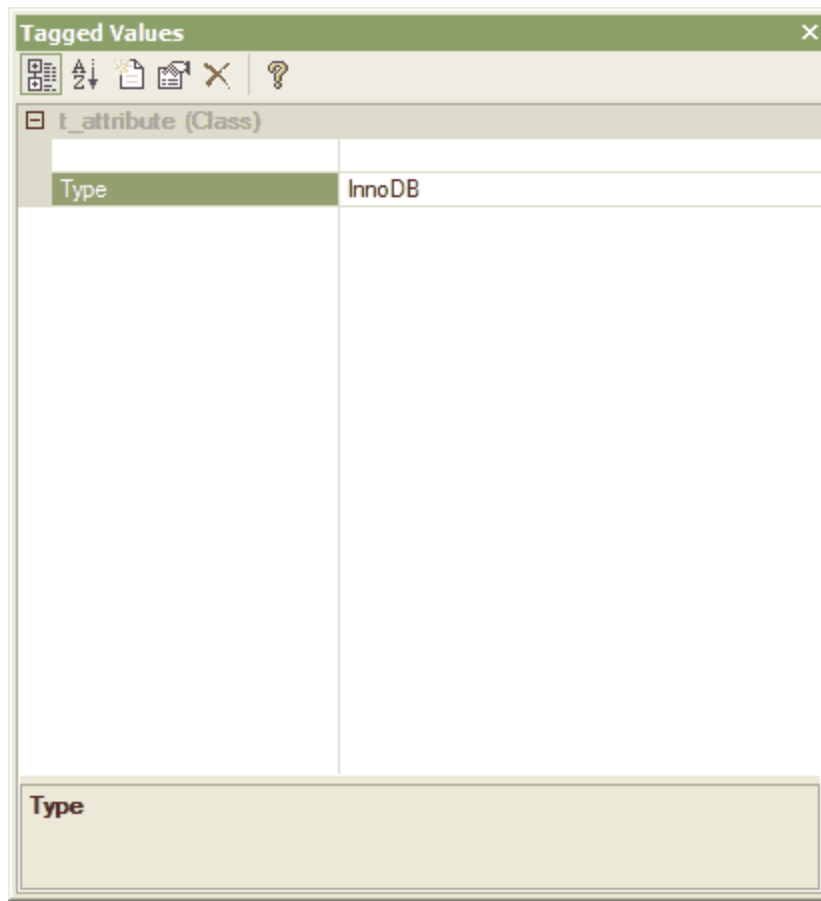
10.3.2 Set MySQL Table Type

In MySQL, if you wish to make use of foreign keys you will need to declare the table type as "InnoDB". To do this, follow the steps below:

1. Ensure the *Tagged Values* window is open by selecting *View | Other Windows | Tagged Values* (or press the *Ctrl + Shift + 6* hotkey combination).
2. Select the table by clicking on it in a diagram or the *Project View* window. The *Tagged Values* window will now have the table selected.
3. Press the *New Tag* button.
4. Define a tag named "**Type**" in the *Tag* field. Add a comment of required in the *Note* field.



5. Press **OK** the button to confirm the operation.
6. In the *Tagged Values* Window, enter the Value **"InnoDB"** as the value for the "Type" tag. Generated DDL will include the table type in the SQL script.



10.4 Create Columns

What is a Column?

The basic organisational element of a relational database is the Column. Every individual item of data entered into a relational database is represented as a value entered in a column of a row in a table. Columns are represented in the *UML Data Modeling Profile* as a stereotyped attribute; that is, an attribute with the "column" stereotype.

Create Columns

To create columns, follow the steps below:

1. Right click on the table in a diagram to open the context menu.
2. From the *Element Features* submenu, select *Attributes*.
3. The *Attributes* dialog will appear.
4. Enter a column *Name* and *Data Type* and press *Save*.

Tip: If the pulldown list of datatypes is empty, this means that you have not selected a target database for the table. Close the *Attributes* dialog and re-open the *Table Properties* dialog to set a *Database* type before continuing. To prevent this recurring, [set the default database type](#).

5. For each column you can optionally set:

- **Primary Key.** Tick this box if the column represents the [primary key](#) for this table.
- **Not Null.** Tick this box if empty values are forbidden for this column.
- **Unique.** Tick this box if it is forbidden for any two values of this column to be identical.
- **Initial Value.** Enter a value in here which will be used as a default value for this column, if required.
- **Access.** Select a scope of *Private*, *Protected* or *Public* from the pulldown list (this defaults to *Public*).
- **Alias.** Enter an alternative name for the field (for display purposes), if any.
- **Notes.** This field is available for you to enter any other information necessary to document the column.

Tip: Some datatypes, for example the Oracle **NUMBER** type, will require a precision and scale. These fields will appear where required and should be filled in as appropriate. For Oracle, create the **NUMBER** type by setting Precision = 0 and Scale = 0; create **NUMBER(8)** by setting Precision = 8 and Scale = 0; and create **NUMBER(8,2)** by setting Precision = 8 and Scale = 2.

Staff Attributes: StaffID

General Detail Constraints

Name: StaffID

Data Type: Integer Primary Key Not Null Unique

Stereotype: column

Initial: 0

Access: Public Alias:

Notes:

Columns

PK	Name	Type	Not ...	Unique
PK	StaffID	integer	Yes	No
	StaffName	Text	No	No
	Staff_Address	Text	No	No

Set the Column Order

To change the column order, follow the steps below:

1. Highlight a column name in the list of defined columns.
2. Press the upward pointing finger button to move the column up one position.
3. Press the downward pointing finger button to move the column down one position.

10.5 Primary Key

What is a Primary Key?

Keys are used to access tables, and come in two varieties: **Primary Keys** and **Foreign Keys**. A Primary Key uniquely identifies a record in a table, while a Foreign Key accesses data in some other related table via its Primary Key. This page describes Primary Keys; Foreign Keys are described [elsewhere](#).

Define a Simple Primary Key

If a primary key consists of a single column it is very easy to define.

1. Right click on the table in a diagram to open the context menu and from the *Element Features* submenu, select *Attributes*.
2. In the *Attributes* dialog, select the column which makes up the primary key.
3. Check the *Primary Key* checkbox and press the *Save* button.

It will be seen that a stereotyped operation is automatically created. It is this operation that defines the primary key for the table. To remove a primary key, simply delete this operation.

Define a Complex Primary Key

It will often be the case that a primary key needs to consist of more than one column. For example, a column "LastName" might not be unique within a table, so a primary key is created from the "LastName", "FirstName" and "DateOfBirth" columns. Perform the following steps to create a complex primary key:

1. Follow the steps above to create a Simple Primary Key. It doesn't matter which column you choose to do this.
2. Right click on the table in a diagram to open the context menu and from the *Element Features* submenu, select *Operations*.
3. Select the Primary Key operation (its name will begin "PK_") and then select the *Columns* tab.
4. To add a column to the primary key, press the *New* button, select a column from the *Column Name* list box, and then press the *Save* button.
5. Use the buttons with the up and down pointing hands to change the order of columns in the primary key, if necessary.

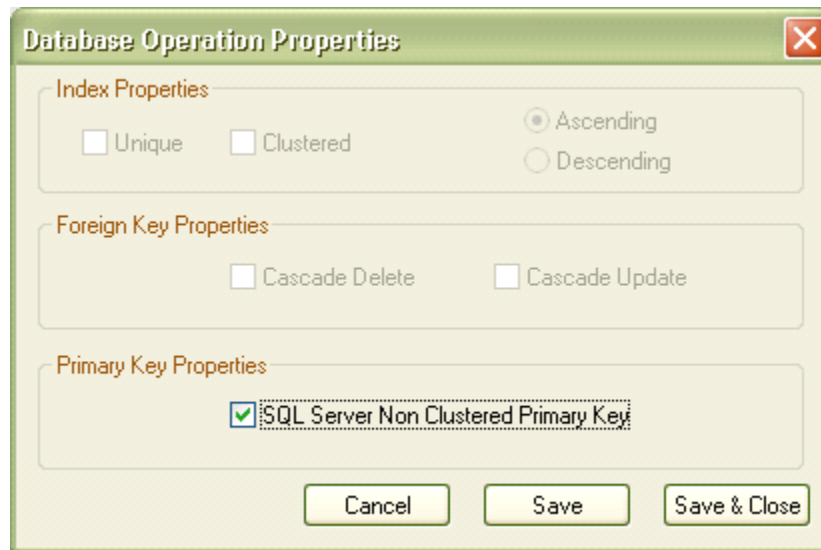
See also: [define a primary key as non-clustered for a SQL Server table](#)

10.5.1 Primary Key Extended Properties

Set SQL Server Clustered /Non Clustered Primary Keys

To define a primary key as non-clustered for a SQL Server table, follow the steps below:

1. Right click on the table in a diagram to open the context menu.
2. From the *Element Features* submenu, select *Operations*.
3. The *Table Operations* dialog will appear.
4. Highlight the Primary Key Operation and select *Extended Properties*.
5. Check the *SQL Server Non Clustered Primary Key* check box.



10.6 Foreign Keys

What is a Foreign Key?

Keys are used to access tables, and come in two varieties: Primary Keys and **Foreign Keys**. A Primary Key uniquely identifies a record in a table, while a Foreign Key accesses data in some other related table via its Primary Key.

Foreign keys are represented in Enterprise Architect UML using **stereotyped operations**. A Foreign Key is a collection of columns (attributes) that together have some operational meaning (they enforce a relationship to a Primary Key in another table). A foreign key is modeled as an operation stereotyped with the "**FK**" **stereotype** - the operation parameters become the columns involved in the key.

Note: It isn't necessary to define a Foreign Key in order to access another table through its Primary Key. Foreign Keys are a feature of some database management systems, providing "extras" such as **referential integrity checking** which prevents the deletion of a record if its primary key value exists in some other table's foreign key. The same thing can be achieved programmatically.

Create a Foreign Key

To create a foreign key, follow the steps below:

1. Locate the required tables in either a diagram or in the Project Browser.
2. Select an associate link in the *Class Toolbar*.
3. Click on the table that will contain the foreign key (source) and draw a connector to the other table (target).
4. Use the link context menu to open the *Foreign Key Constraint* dialog.

Foreign Key Constraint [X]

Name:

Source: Inventory Target: Warehouse

Key	Column	Type
PK	InventoryID	VARCHAR
	WarehouseID	VARCHAR
	CurrentStock	INTEGER
	OnOrder	INTEGER

Key	Column	Type
PK	WarehouseID	VARCHAR
	Location	VARCHAR
	Capacity	NUMERIC

Referential Integrity

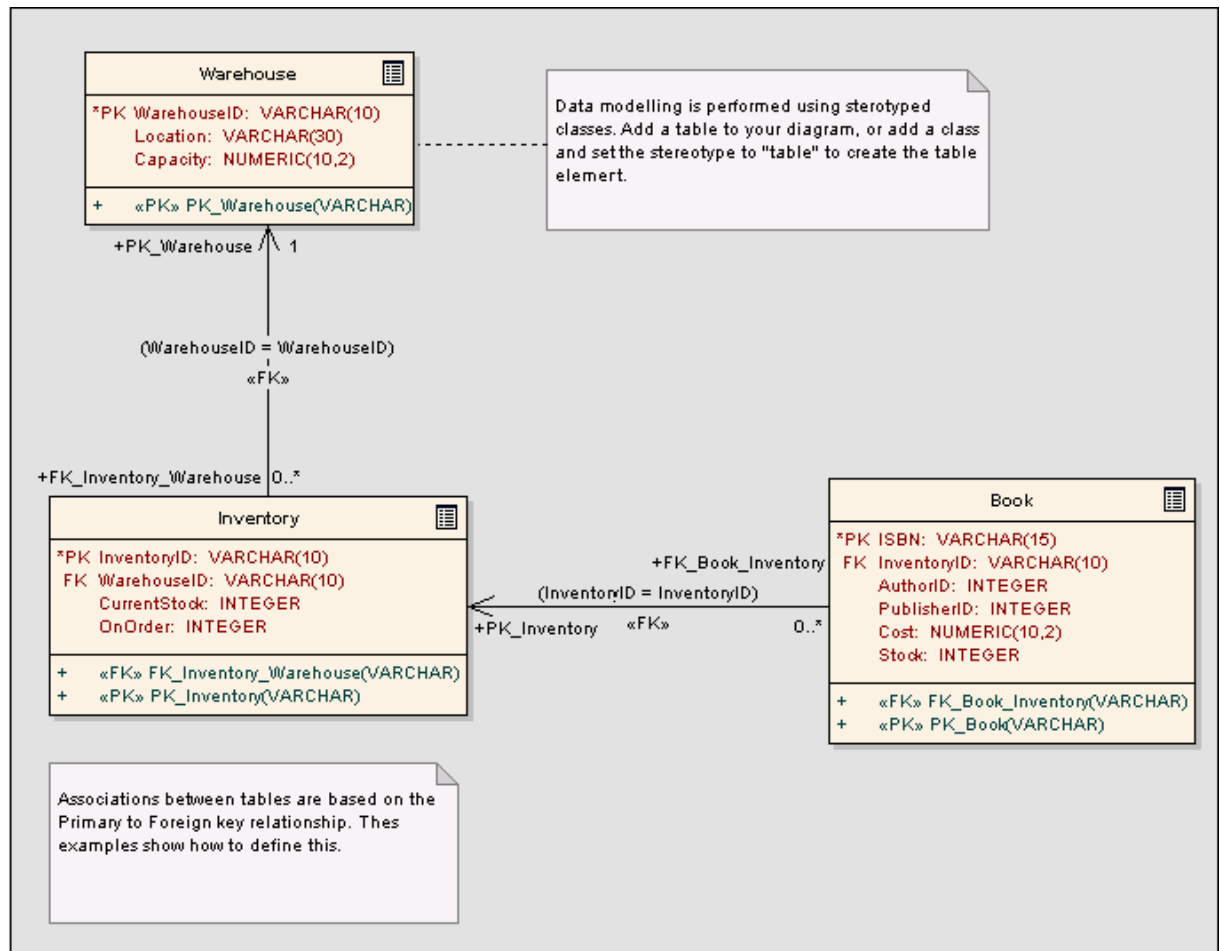
Delete Cascade Update Cascade

Column	Type
WarehouseID	VARCHAR

Column	Type
WarehouseID	VARCHAR

- The default name for the foreign key can be edited.
- Highlight the columns involved in the foreign key relationship.
- Press **Save** to automatically generate the foreign key operations.

You have created a Foreign Key. The example below shows how this will look in a diagram:



- To create a composite foreign key, select the appropriate columns and press **Save**. The foreign key columns will be sorted according to datatype to match the datatypes of the composite targeted primary key. If desired, the order of the key columns can be changed using the **move up** and **move down** buttons.

Foreign Key Constraint ✕

Name:

Source: Table2 Target: Table1

Key	Column	Type
	t2_date	datetime
	t2_id	int
PK	t2_pk	int
	t2_name	varchar

Key	Column	Type
PK	t1_id	int
PK	t1_name	varchar
PK	t1_date_cre...	datetime

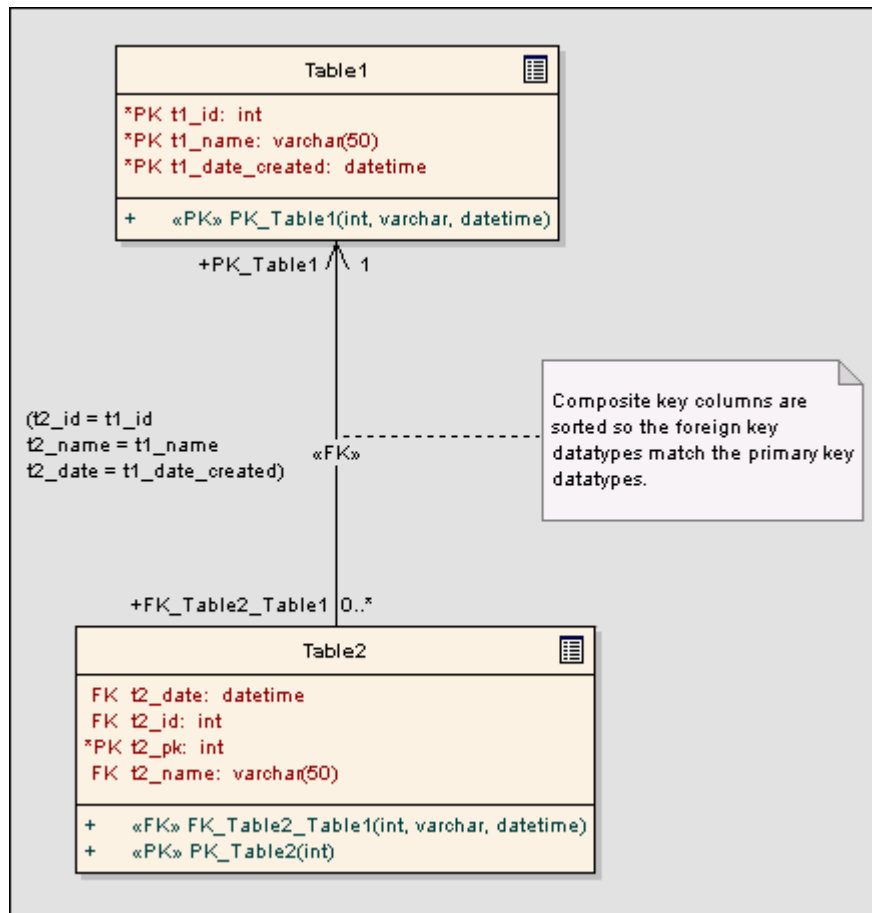
Referential Integrity

Delete Cascade Update Cascade

Column	Type
t2_id	int
t2_name	varchar
t2_date	datetime

Column	Type
t1_id	int
t1_name	varchar
t1_date_created	datetime

By using the feature a Foreign Key will be created. The example below shows how this will look in a diagram:



Hint: If you are defining a MySQL database and wish to use foreign keys, you will have to [set the table type](#) to allow this.

10.7 Indexes and Triggers

What is an Index?

An index is a sorted look-up for a table. When it is known in advance that a table needs to be sorted in a specific order, it is usually worth the small processing overhead to always maintain a sorted look-up list rather than sort the table every time it is needed. In EA, an index is modeled as a stereotyped operation. On generating DDL, the necessary instructions for generating indexes are written to the DDL output.

What is a Trigger?

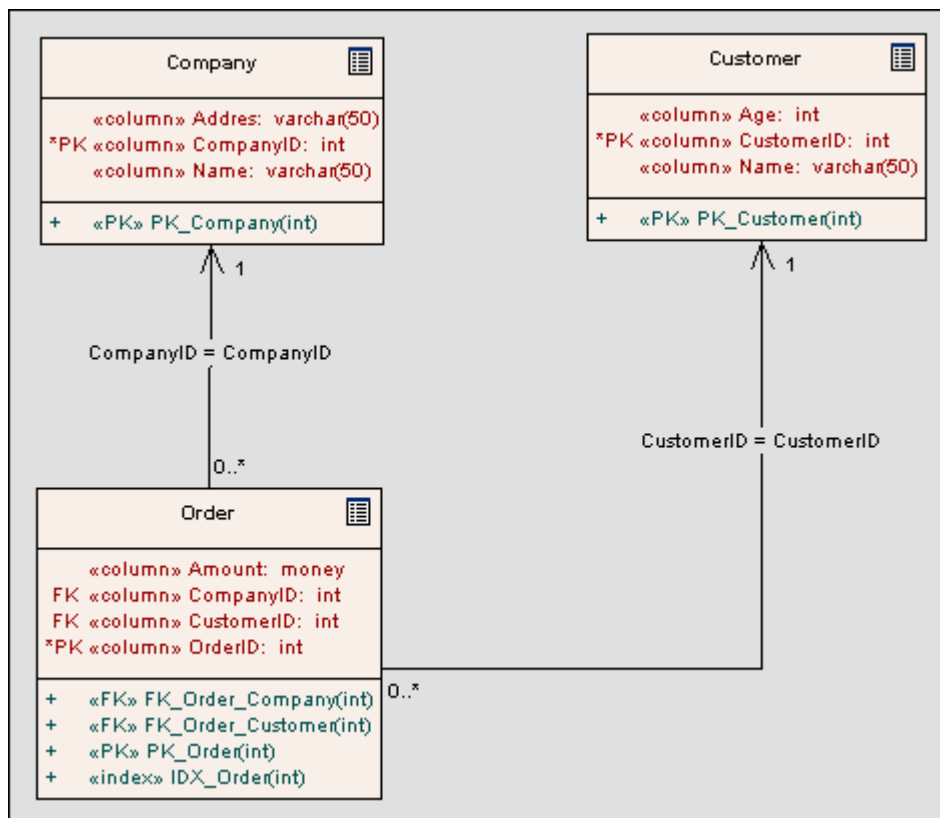
A trigger is an operation automatically executed as a result of the modification of data in the database, and will usually ensure consistent behavior of the database. For example, a trigger might be used to define validations that must be performed every time a value is modified, or might perform deletions in a secondary table when a record in the primary table is deleted. In EA, a trigger is modeled as a stereotyped operation. Currently EA will not generate DDL for triggers, but nonetheless they aid in describing and specifying the table structure in detail.

Create an Index or Trigger

1. Locate the required table in either a diagram or in the Project Browser.

2. Use the context menu to open the *Operations* dialog.
3. Add an operation (such as "IDX_CustomerID or TRG_OnCustomerUpdate" - the IDX_ and TRG_ prefixes are optional but help identify the operation).
4. Set the *Stereotype* for the Operation to "index" or "trigger" as appropriate - "check", "proc" and "unique" are also supported.
5. Enter the body of the trigger, procedure, or details of the check constraint in the Notes field of the Operation.
6. Select the *Operation* and go to the *Columns* tab.
7. Add the required columns in the desired order then press *Save* to save changes.

You have created an index or trigger. The example below shows how this will look in a diagram:



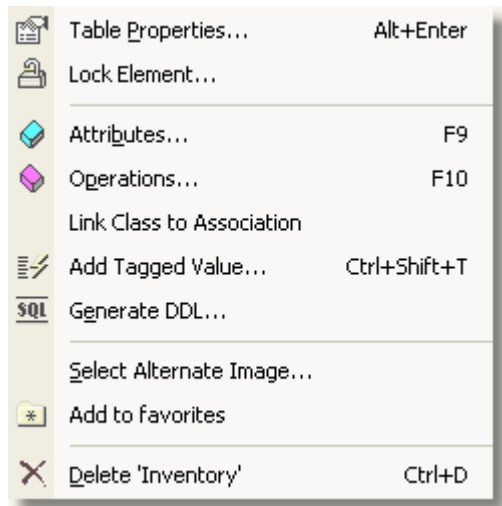
10.8 Generate DDL

Enterprise Architect will generate simple DDL scripts to create the tables in your model.

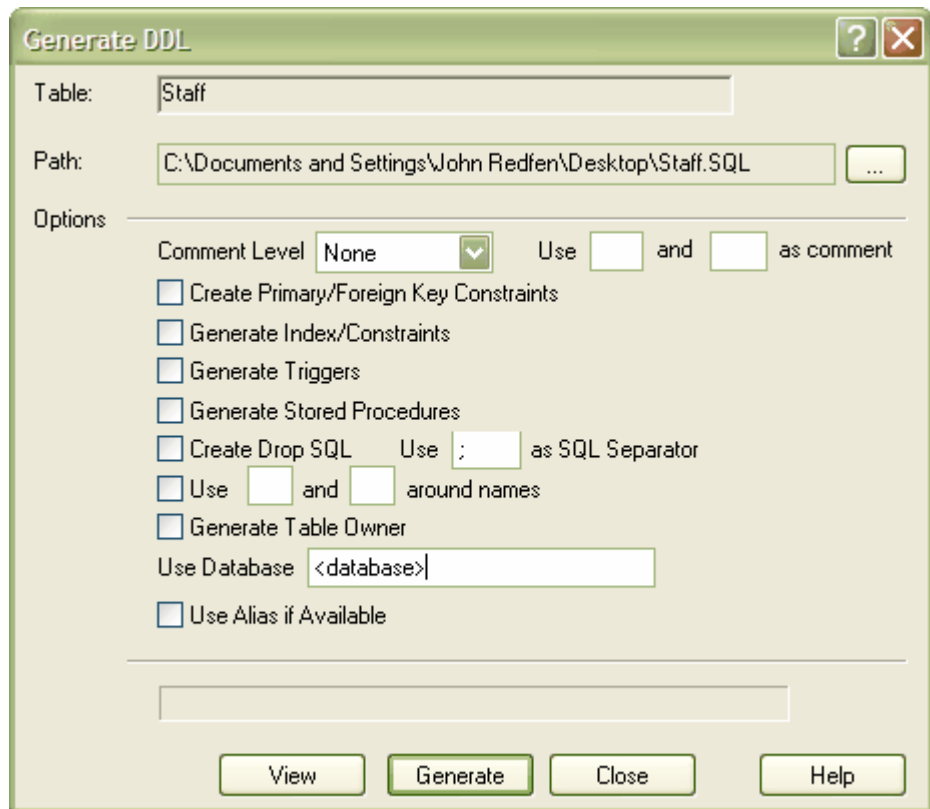
Generate DDL

To generate DDL, follow the steps below:

1. Select the table in the diagram to generate DDL for.
2. Right click to open the context menu and choose *Generate DDL*.



3. In the *Generate DDL* dialog, enter the *Filename* of the script to create.
4. Check the *Create Drop SQL* if you wish to include a 'drop table' command in the script.
5. Press *Generate* to create the DDL.
6. Press *View* to view the output (you will have to configure a DDL viewer in the Local Settings dialog first).



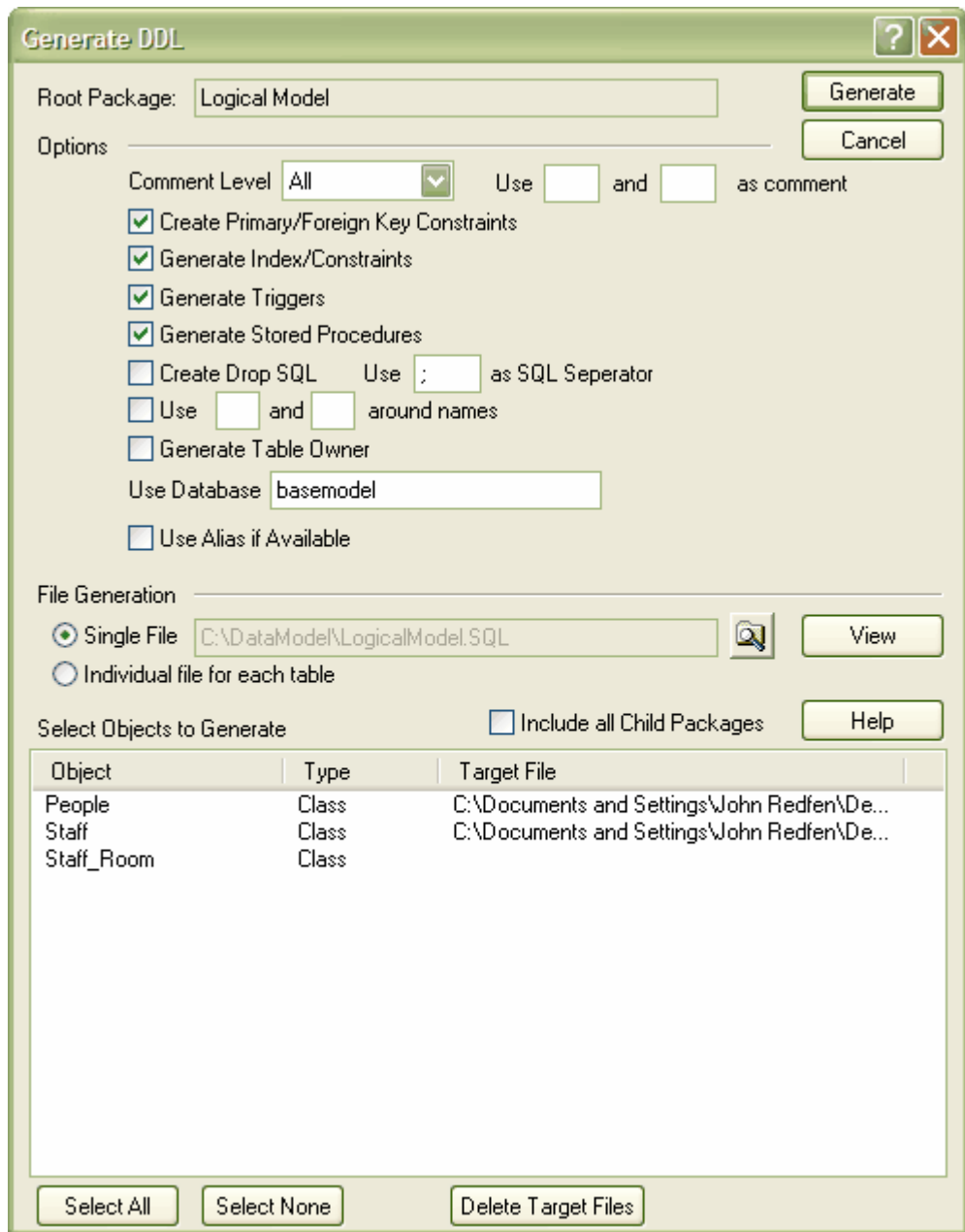
10.9 Generate DDL for a Package

To generate DDL for a package, follow the steps below:

1. Locate the required package in the Project Browser.
2. Right click to open the context menu.
3. From the *Code Engineering* submenu, select *Generate DDL*.

Note: Alternatively you can select *Generate Package DDL* from the *Project | Database Engineering* submenu.

4. The *Generate* dialog will appear.
5. You can check the *Include All Child Packages* to recursively generate DDL.
6. Press *Generate* to proceed. You will be prompted for file names as the process proceeds if required.



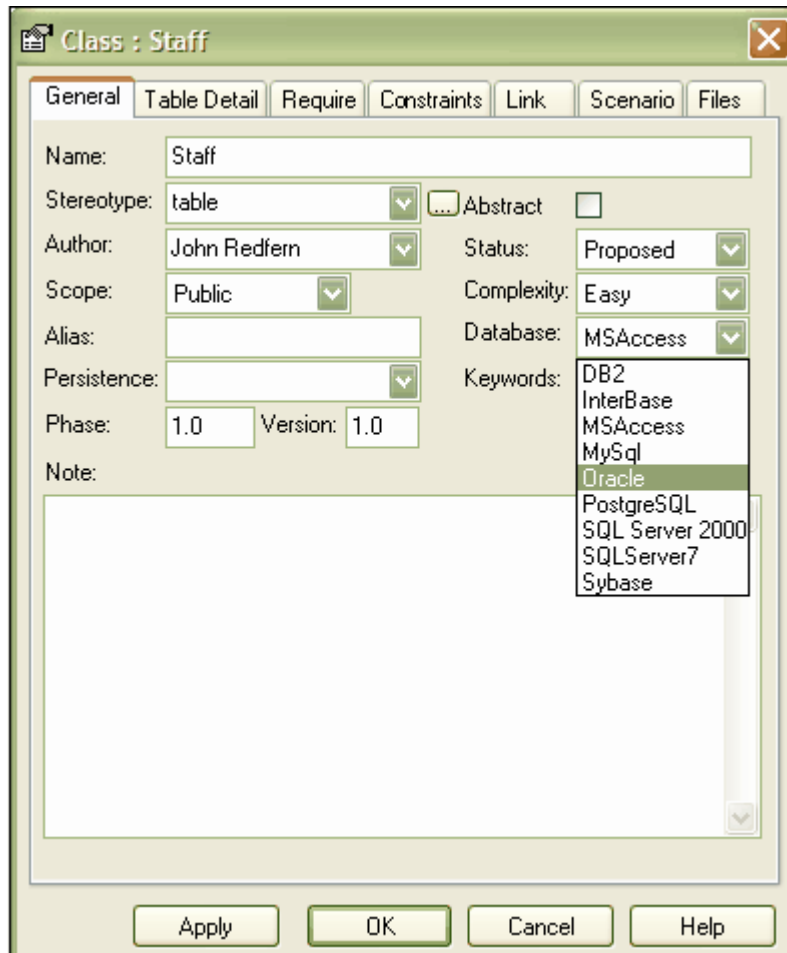
10.10 Data Type Conversion Procedure

Once a database schema has been set up on an EA diagram (either by importing through ODBC or manually setting up the tables), the DBMS can be changed to another type and the column datatypes will be mapped accordingly.

To Map the DBMS type of a table to another DBMS type:

1. Open the Table Properties Dialog. The Database drop down will show the current DBMS for this table.

2. To map the column datatypes to another DBMS, select the target from the Database drop down and press the **Apply** button.
3. The datatypes will be converted to match those of the new DBMS, and these will be reflected in any DDL generated from this table.



10.11 Data Type Conversion for a Package

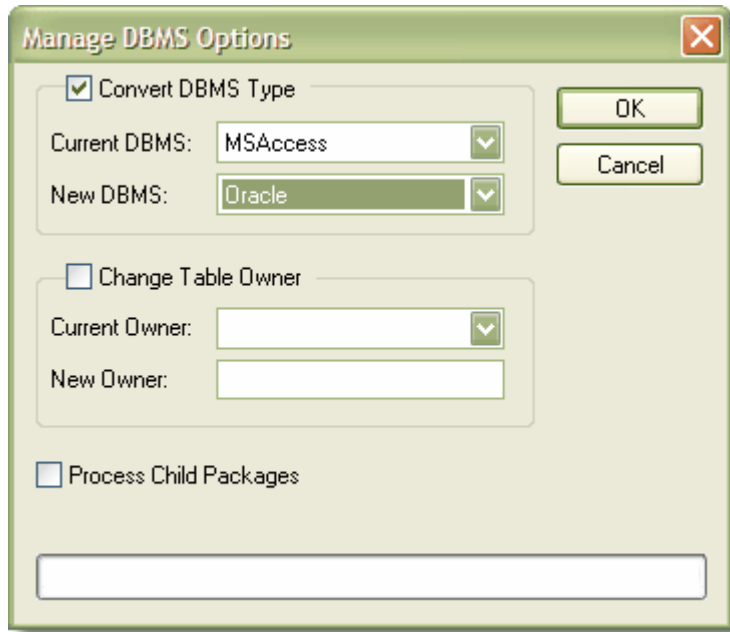
The DBMS Package procedure or mapper allows the user to convert a package of database tables from one DBMS type to another DBMS type as well as providing the ability to change the ownership of table owners .

To Map the DBMS types of a package to another DBMS type:

1. Right click on the package in the project tree, select **Code Engineering | Reset DBMS Options**.
2. In the resulting Manage DBMS Options dialog, select the current DBMS from the **Current DBMS** drop down menu and the target DBMS from the **New DBMS** drop down menu.
3. Ensure that the **Convert DBMS Type** checkbox is selected.
4. If there are child packages that also require changing, check the "**Process Child Packages**" check

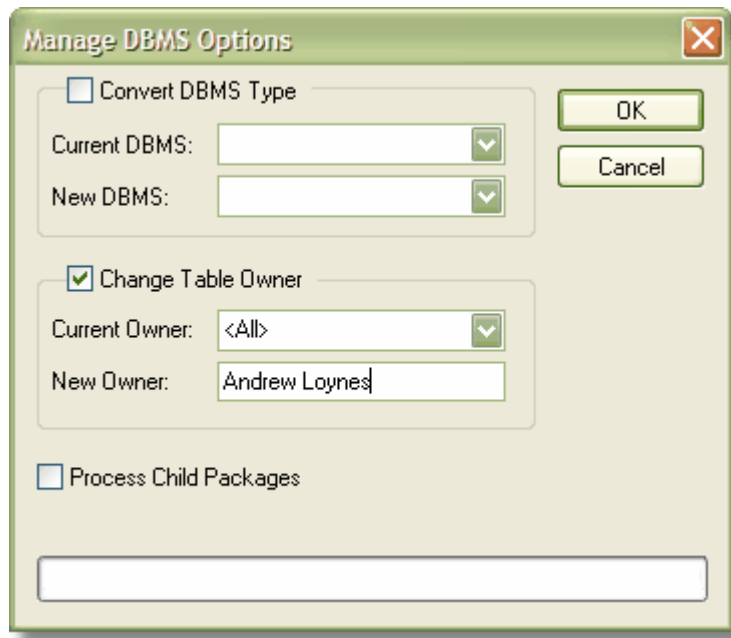
box.

5. Click **OK**. All Tables in the selected packages will be mapped to the new DBMS.



To change the owner of the table or all of the tables in a package use the following instructions:

1. Right click on the package in the project tree, select *Code Engineering | Reset DBMS Options*.
2. In the resulting Manage DBMS Options dialog, enter the name for the new table owner in the *New Owner* field
3. From the *Current Owner* drop down select the current owner that is to be changed or select *<All>* to change the ownership of all of the tables in the package to the name defined in the *New Owner* field.
4. Ensure that the *Change Table Owner* checkbox is selected.
5. If there are child packages that also require changing, check the "*Process Child Packages*" checkbox.
6. Click **OK**. All Tables in the selected packages will have their ownership changed.

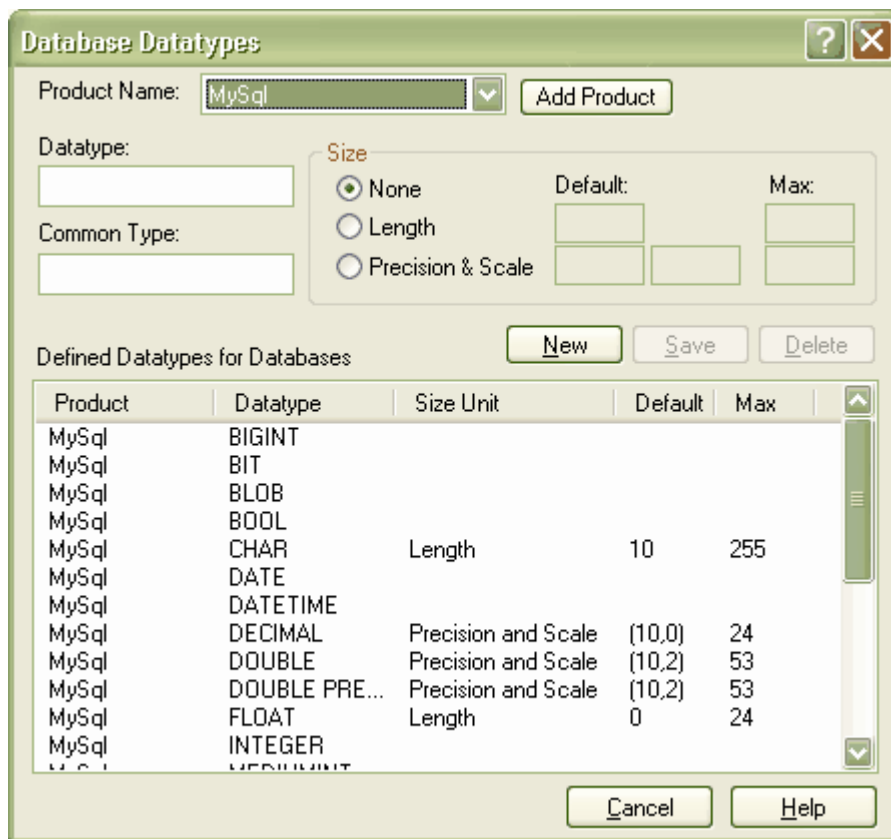


For more information relating to setting the table owner see the [Set Table Properties](#) topic, to display the [Table Owner](#) in the current diagram see the [Set Appearance Options](#) section.

10.12 DBMS Datatypes

When setting up your data modeling profile, you can customize the datatypes associated with a particular DBMS using the Database Datatypes screen. This screen allows you to add and configure custom data types. For some data types you will need to add the size and precision, defaults and maximum values.

The Database Datatypes screen is available by selecting *Database Datatypes* from the *Configuration* menu. You may also add a DBMS product and configure the inbuilt data types.



10.13 Import Database Schema from ODBC

Enterprise Architect supports importing database tables from an ODBC data source. Tables will be imported as stereotyped classes with suitable data definitions for the source DBMS.

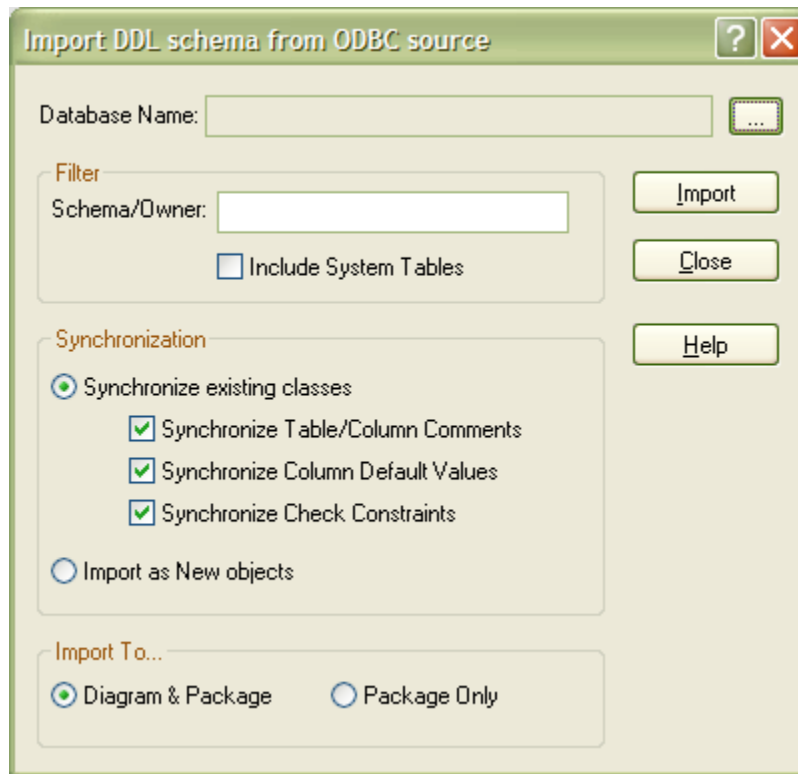
Import Database Tables

To import database tables, 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 bring up the context menu.
3. Select *Code Engineering | Import DDL schema from ODBC*.
4. To import into a diagram, select a suitable diagram in the selected package.
5. Right click on the diagram to open the context menu.
6. Select *Import DDL schema from ODBC*.

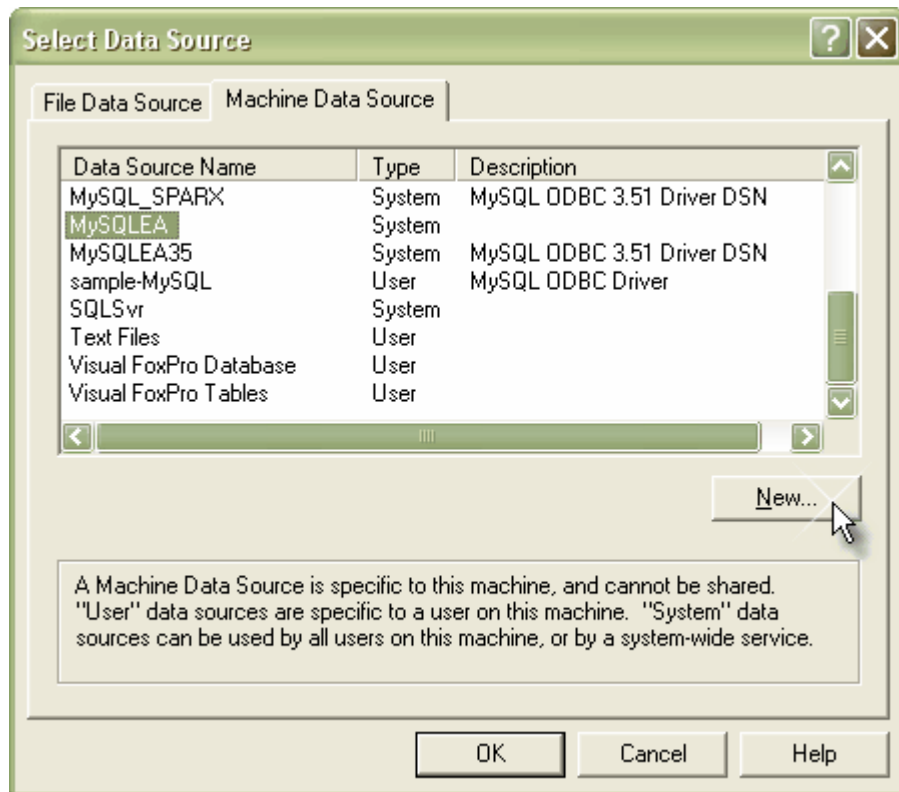
Note: Alternatively you can select *Import DDL from ODBC* from the *Project | Database Engineering* submenu.

7. Select an ODBC data source, then press *Import* to start the import. When synchronizing existing classes, select the appropriate check box to determine whether the model comments, default values or constraints are to be synchronized with the ODBC tables.



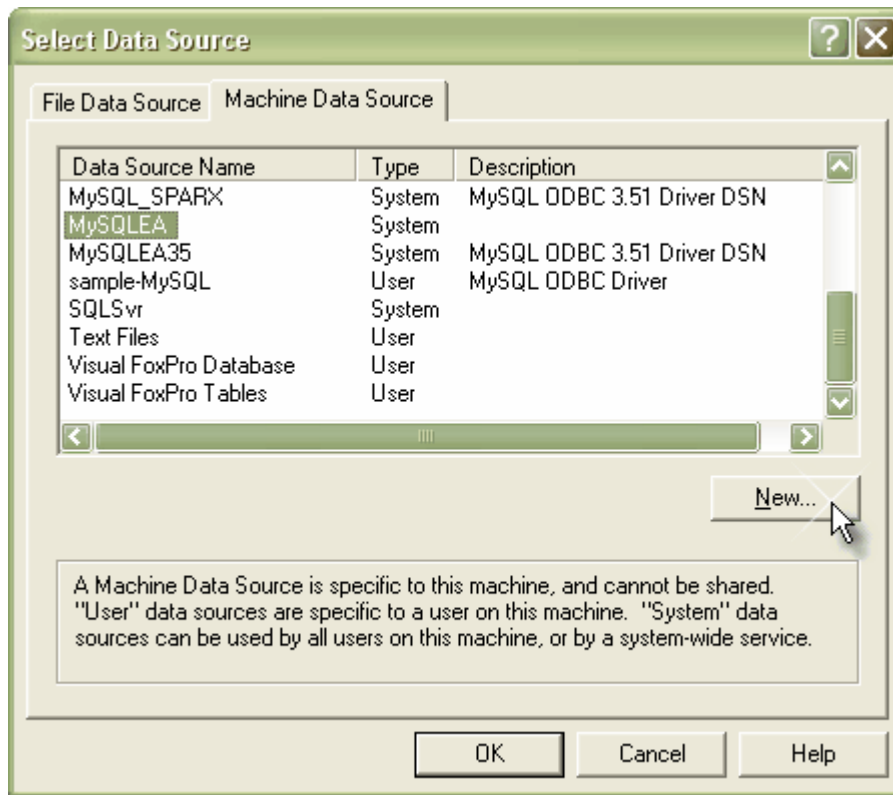
Note: It is only possible to import into a diagram if it is in the selected package. If a diagram from another package is open, a message will give the option to cancel the import or to continue importing into the package only. The Dialog includes options to import into **Diagram & Package**, or **Package Only**. If no diagram is open, the **Package Only** button will be selected and the options will be disabled. If the open diagram is in the selected package, either option can be selected.

8. Choose an ODBC data source. Select a suitable data source from the ODBC dialog (ODBC must be installed and configured on your machine for this to work correctly).



10.13.1 Select a Data Source

Importing DDL from existing data sources requires you to have installed and configured a suitable ODBC connection. From the Import DDL dialog you can select the ODBC data source using the standard windows ODBC set-up dialog.



10.13.2 Select Tables

Once you have opened the ODBC data source, EA will acquire a list of tables suitable for importing. This list is presented in a list form for you to select from. Highlight the tables you wish to import and clear those you do not require.

Selection shortcuts:

- To select all tables, press **Select All**
- To clear all tables, press **Select None**
- Hold down the **Ctrl** key while clicking on tables to multiple select
- Hold down **Shift** and click on table to select a range

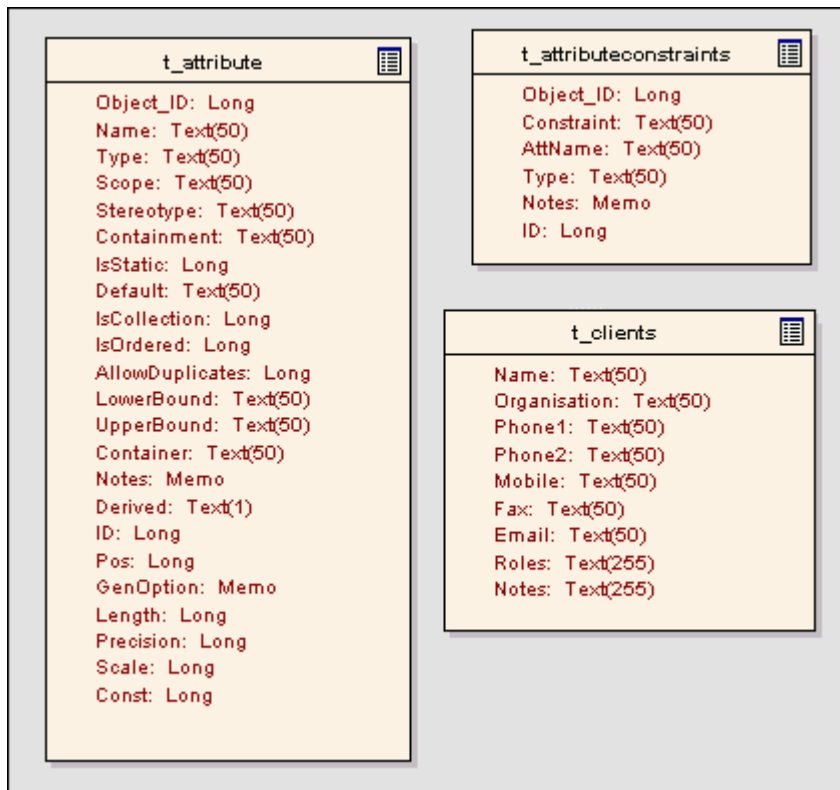
The selection dialog:



10.13.3 The Imported Class Elements

When you import DDL table definitions they are converted to stereotyped classes according the UML Data Modeling Profile.

The image below shows some example tables imported into the model using an ODBC data connection:



Part



11 Creating Documents

The production of documentation is essential to realizing the full benefit of EA. EA outputs high quality documentation in either [RTF](#) or [HTML](#) format. The RTF formatting can be modified directly with [RTF Style templates](#) to alter the look and feel of generated output. Using MS Word you can further enhance the output by connecting and interweaving model output in linked documents.

There are many ways to specify the EA content being documented:

- A package and/or its child packages can be documented by manually highlighting a package and selecting a documentation control.
- Embedded packages can be specified for [exclusion](#) if child packages are recursively documented.
- A package can be linked to an RTF document template to simplify generating consistent types of documentation (e.g., Use Case Reports) using the [Documents feature](#).
- Packages can be selected, grouped, and ordered together in a manner different from the project view by creating ["Virtual" Documents](#).

11.1 RTF Documents

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 the RTF format used by MS Word™, which provides the ability to link together a number of rich text documents into a single master document.

Typically you will create a Word master document, then some Enterprise Architect RTF reports. These are linked back into sub-sections of the master document and refreshed as required during project development. In this way the project document becomes an easy to manage and feature-rich work product.

By adding tables of contents, figure tables, header & footers and sections you can manage a complex document with relative ease. Simply update the Enterprise Architect RTF reports then refresh the links in MS Word.

Note: For more information on using MS Word in this manner see the Word help file.

Tip: You can create [RTF Style Templates](#) to customize your RTF output.

See:

- [The RTF Dialog](#)
- [Using MS Word](#)
- [Other Documents](#)
- [RTF Style Templates](#)
- [Custom Language Settings](#)

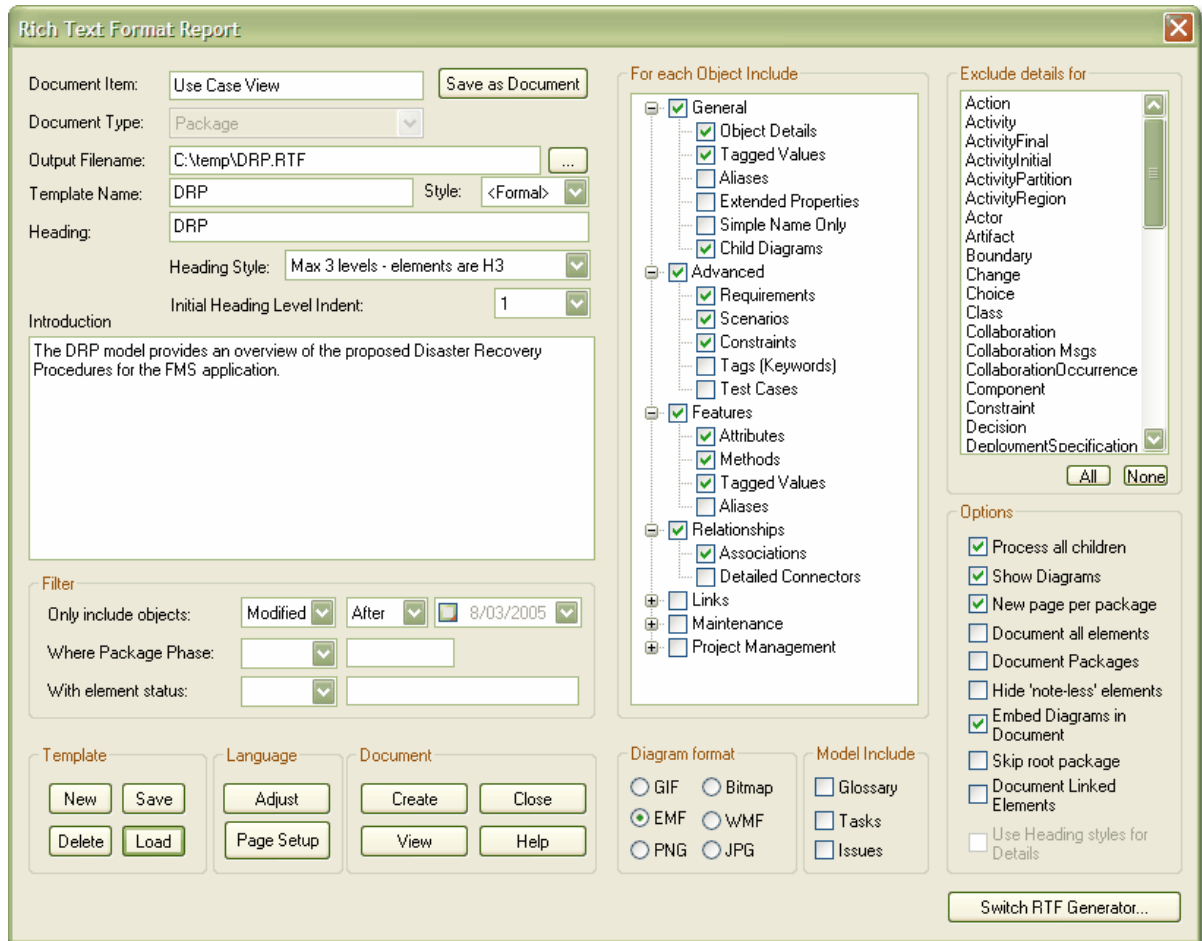
11.1.1 The RTF Dialog

The following topics provide assistance on using the RTF Report dialog to document your project.

See:

- [Create a Rich Text Document](#)
- [Document a Single Element](#)
- [The RTF Report Dialog](#)
- [Set the Main RTF Properties](#)
- [Apply a Filter](#)

- [Exclude Elements](#)
- [RTF Diagram Format](#)
- [Model Include](#)
- [RTF Options](#)
- [RTF Selections](#)
- [Generate the Report](#)
- [Diagram Only Report](#)
- [Report Templates](#)
- [Include or Exclude a Package from Report](#)
- [Save as Document](#)



11.1.1.1 Create a Rich Text Document

Creating a rich text document is a simple and flexible process. An RTF document is always based on a package in your project. To produce a document, you need to select the package you wish to report on in the Project Browser window.

Tip: Reports can be configured to include all packages within a parent package, or just the top level.

Once you have your package selected, you will use the context menu to open the RTF Report dialog and configure the details of your document. This next section will guide you through creating a rich text report.

How to Open the RTF Report Dialog

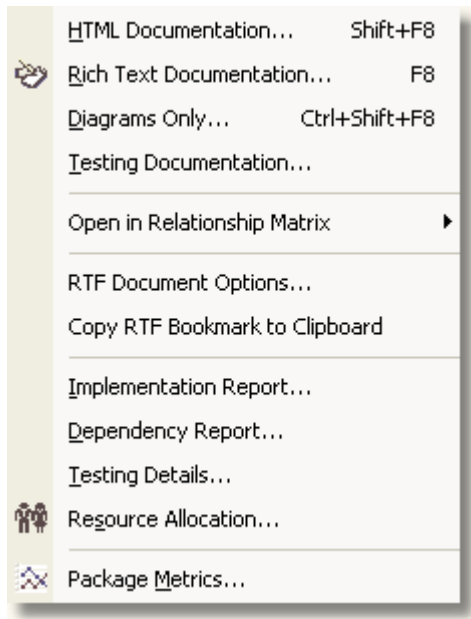
1. In the Project Browser, right click on the package you need to create documentation for, to open the context menu. Select *Rich Text Documentation* from the *Documentation* submenu.

-OR-

2. Select *Rich Text Format Report* from the *Project | Documentation* submenu.

-OR-

3. Use the keyboard shortcut *F8*.



See [The RTF Report Dialog](#) and related topics for more information.

11.1.1.2 Document a Single Element

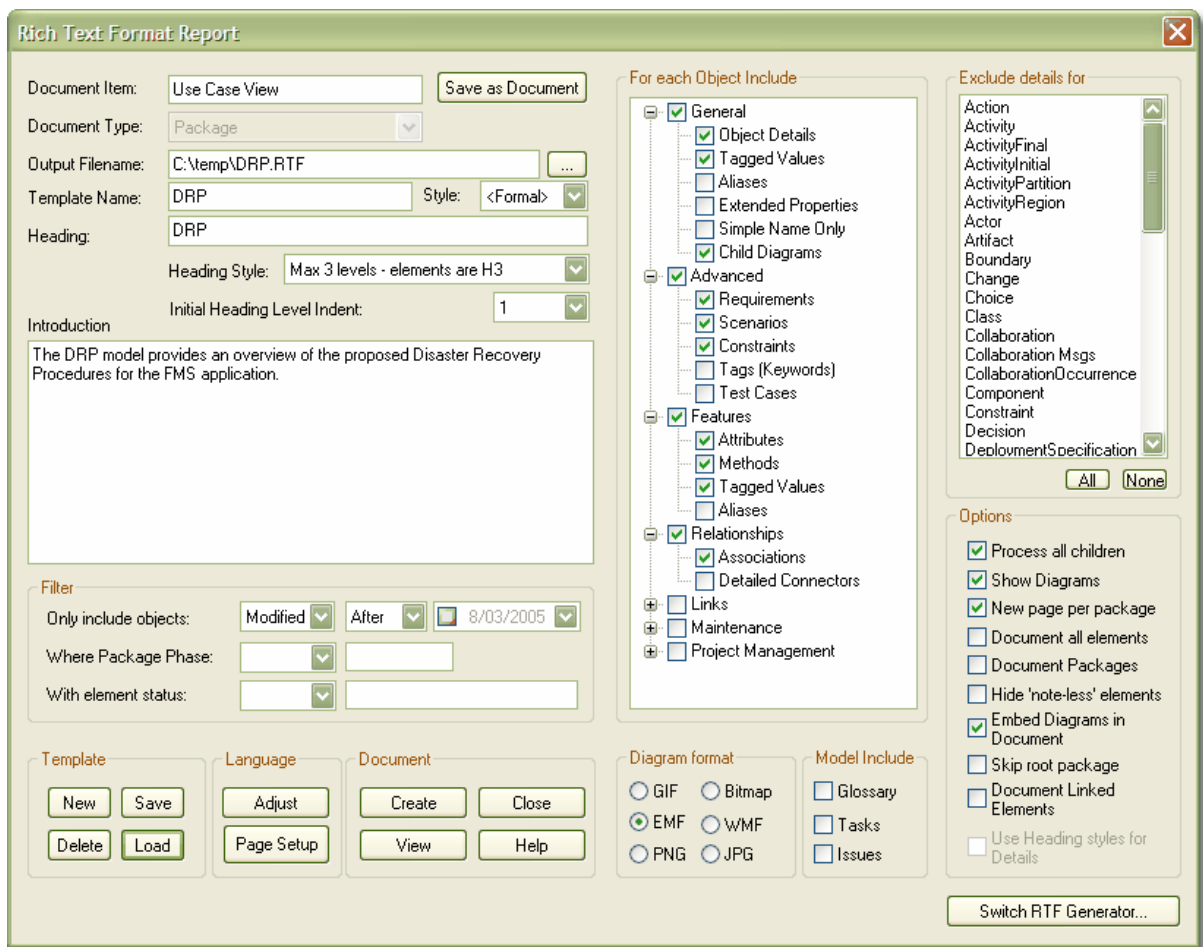
RTF documentation may also be generated for a single element.

Select the element you wish to generate the documentation for, and then select *Documentation* from the *Element* menu. This will open the *Rich Text Format Report* dialog. See [The RTF Report Dialog](#) and related topics for further information.

11.1.1.3 The RTF Report Dialog

The RTF Report dialog allows you to set the exact contents and look and feel of your report. Enter the file name of the report, a heading, additional notes, template name (for saving the set-up) and other options. You may also select the style of the report - either plain or formal.

Optionally, set up a filter, the details to include, element types to exclude, whether to process child packages, whether to show diagrams and the diagram format.



Note: The RTF dialog has a lot of options. Get to know them all to produce output at the level of detail suited to your project.

11.1.1.4 Set the Main RTF Properties

The main section of the RTF Report dialog allows you to set the output location and appearance of the final RTF document.

Setting Options for the RTF Document

1. Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The main section of this dialog is shown below.
2. Supply an **Output Filename** to save the report into - always include the extension .RTF as part of the filename.
3. Provide a **Template Name** if you wish to save this report set-up.
4. Select a report **Style** - Formal or Basic.
5. Enter a **Heading** for your report - appears as the first heading item in your output.

6. Select your desired *Heading Style* and *Initial Heading Level Indent* from the drop lists.

Current Package: Save as Document

Output Filename:

Template Name: Style:

Heading:

Introduction Heading Style:

The Use Case Model describes the proposed functionality of the new system. A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example creating a train, modifying a train and creating orders are all Use Cases. Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behaviour. Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.

Note: It is recommended that you enter a full path name for your report. The images in your report are saved externally in an images directory, and supplying the full directory path avoids confusion over the location of these images. Note also that if you move your report, you must also move the images directory.

11.1.1.5 Apply a Filter

You may apply a filter on the RTF Report dialog to include or exclude elements by date modified, phase or status. This helps to track changes and break a document into multiple delivery phases.

Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The *Filter* section of this dialog is shown below.

- To enable the Date filter, check the box in the drop down list. Set the selection criteria as required (modified/created, before/after).
- The Phase filter applies at the Package level (not the element level) - and ignores the phase of the root package that you are documenting. If you enable the phase filter by selecting an operator from the phase operator combo, EA will filter out all packages that do not meet the selection criteria. All elements in that package are ignored - regardless of their individual phase.
- The Status filter lets you limit the output by element status. Unlike the phase filter, this filter applies to every element. You can filter where a status is like or not like a criterion, eg. "like proposed" or using the in and not in operators eg. in approved, validated. When using the in and not in operators, enter a comma separated list of status types as your criteria expression.

Filter

Only include objects:

Where Package Phase:

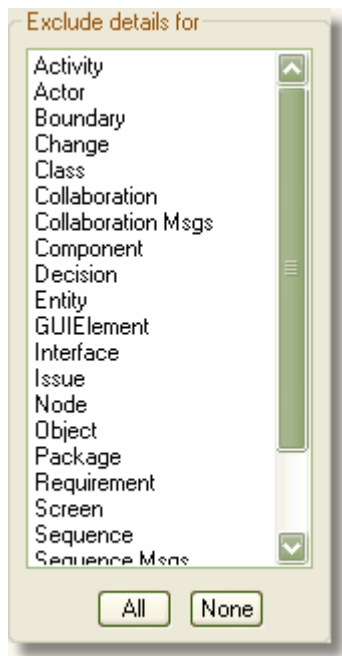
With element status:

11.1.1.6 Exclude Elements

The RTF Report dialog allows you to exclude any element types you desire from your final output. This is useful when you wish to highlight particular items and not clutter up a report with too much detail.

Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The *Exclude details* section of this dialog is shown below.

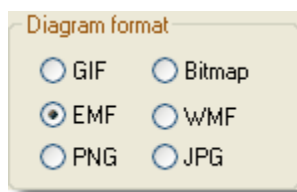
Left mouse click to select the elements you wish to exclude. Press *None* to clear your selections, or press *All* to select all elements.



11.1.1.7 RTF Diagram Format

Diagrams may be output to Bitmap files, GIF files or Windows Metafiles.

Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this). The *Diagram format* of this dialog is shown below. Select the format you prefer for your report



Bitmap files are raster images with a high level of detail but large size. They do not scale up or down very well. GIF files are raster images with reasonable detail and small size. They scale a little better than bitmaps. Metafiles are vector images with high detail and small size (but often have compatibility problems with some

printers or software). Metafiles scale very well.

PNG files are raster images with reasonable level of detail and smaller file sizes than GIF.

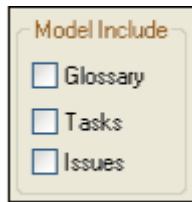
JPEG are lossy raster images with average levels of detail, JPEG does not work very well with line drawings and losses clarity when re sized, JPEG file sizes are typically very small.

Note: *Generally metafiles are the best option, although it sometimes pays to experiment.*

11.1.1.8 Model Include

The Model Include section of the RTF Report dialog has the following options, check the appropriate box to include the items in the RTF documentation:

- Check the **Glossary** item to include the [model glossary](#) in the generated RTF documentation.
- Check the **Tasks** item to include the [model tasks](#) in the generated RTF documentation.
- Check the **Issues** item to include the [model issues](#) in the generated RTF documentation.

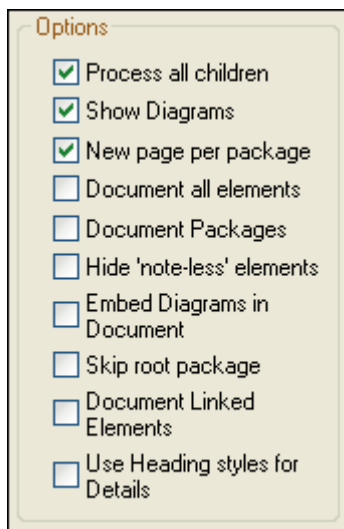


11.1.1.9 RTF Options

Additional **Options** you can select on the RTF Report dialog are shown below.

You can select whether to recursively document packages, to show diagrams or not and to add a page break before each new package or not.

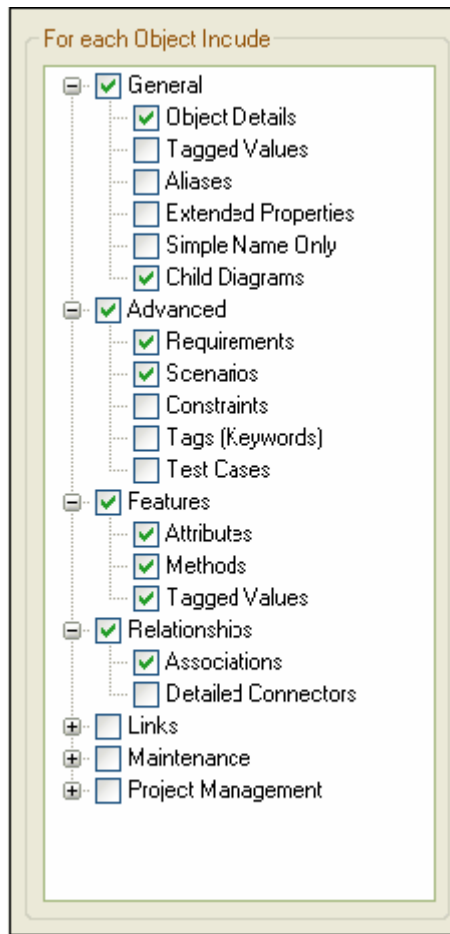
- Check the **Process all Children** item to recursively process all child packages within the main package.
- Check the **Show Diagrams** item to include diagrams in your document. Clear this item for no diagrams.
- Check the **New page per package** item to force a page break on each new package (excepting empty packages).
- Check the **Document all elements** item to include all elements included in the project.
- Check the **Document Packages** item to document the package as an element in addition to the documentation that would normally be produced for package documentation.
- Check the **Hide 'note-less' elements** item to exclude all elements without notes from the documentation.
- Check the **Embed Diagrams in Document** item to insure that the diagram images are contained within the RTF document rather than stored in a linked external file.
- Check the **Skip root package** item to exclude the parent package from the documentation and include only the child package/s.
- Check the **Document Linked Elements** item to include the object details for linked elements that do not originate from the selected package.
- Check the **Use Heading styles for Details** item to ensure that the details are formatted as Heading Styles rather than formatted text. This option is only available when the Heading Style in the [Main section](#) of the RTF Dialog is set to **Max 9 levels - elements are package + 1**.



11.1.1.10 RTF Selections

The *Objects to Include* section of the RTF Report dialog (shown below) allows you to select which documentation sections you want in your report. What you include or exclude will govern how simple or detailed your report is. You may wish to create multiple reports at different levels of detail for different audiences.

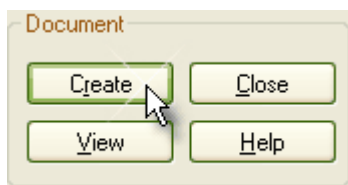
Experiment with these options to see what effect inclusion or exclusion has. Most items are self-explanatory and it is possible to expand the options by pressing the +symbol next to each category in the list. Checking an item in the list will select all of the options that are contained in the category, to exercise greater control over each category of options expand the top level item and then check the required individual items from the list. Sometimes an item will only apply to a certain type of element - eg. *Attributes* are only for classes and a few other element types. *Child diagrams* means show or hide any diagrams which are attached under a model element (eg. a Use Case may have a scenario diagram attached).



Note: Use this feature to produce the right level of detail for your audience. Technical readers may wish to see everything, while management may only wish the general outline.

11.1.1.11 ***Generate the Report***

Once you have set up your document properties as required, you can generate the report. To do this, press **Create**. Once you have generated the document, press **View** to open the RTF in MS Word.

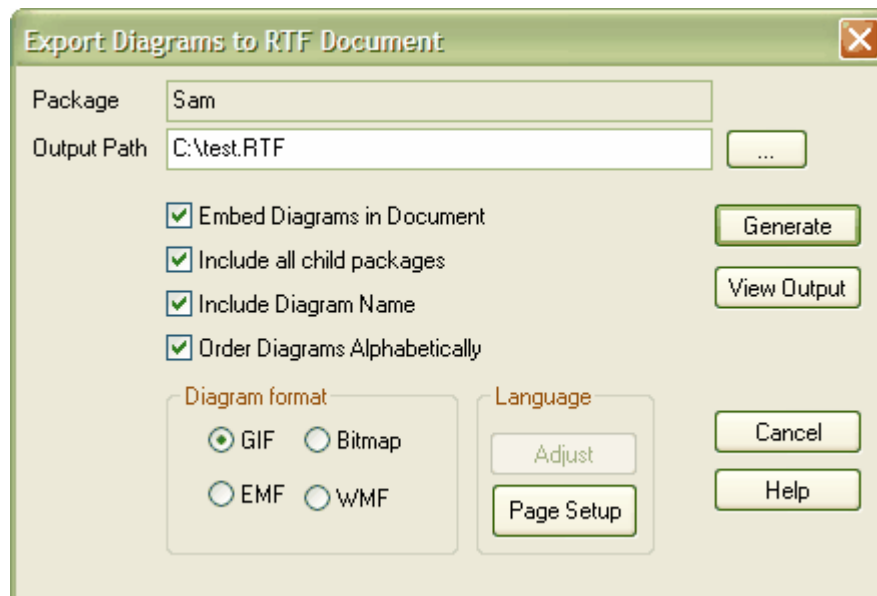


11.1.1.12 Diagram Only Report

You may 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 each one at a time or printing one at a time.

To Produce a Diagram Only Report

1. Right click on a package in the Project Browser to open the context menu.
2. From the *Documentation* submenu, select *Diagrams Only....* to open the *Export Diagrams to RTF Document* dialog.



3. Enter the required information and press *Generate* to run the report.
4. You can press *View* to show the RTF output once generated.

You can select following options to specify options to be included in the generated diagrams only report.

- Check the *Embed Diagrams in Document*, to ensure the diagrams are created within the rtf file not as linked image files.
- Check the *Include all child packages* to document all of the diagrams included in any child package.
- Check the *Include Diagram Name* item have the diagram name included within the generated documentation.
- Check the *Order Diagrams Alphabetically* item to have the documentation generated in alphabetical order.

11.1.1.13 Report Templates

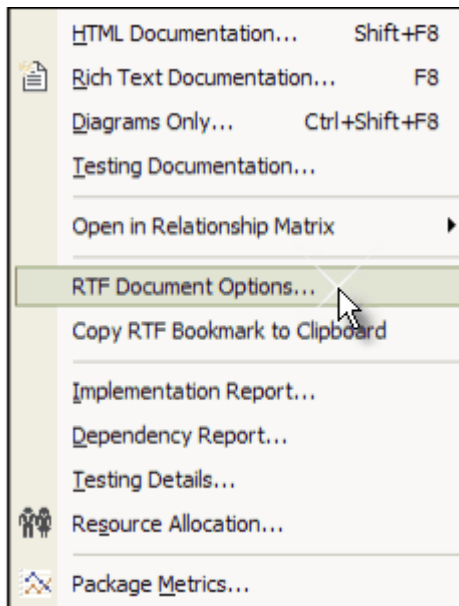
If you have previously defined and saved a template, select the *Load* option on the *RTF Report* dialog to open the list of defined templates. Select one in order to load it as the current template – all the features saved become the current features. This enables you to define a set of standard report types that streamline document production.

11.1.1.14 Include or Exclude a Package from Report

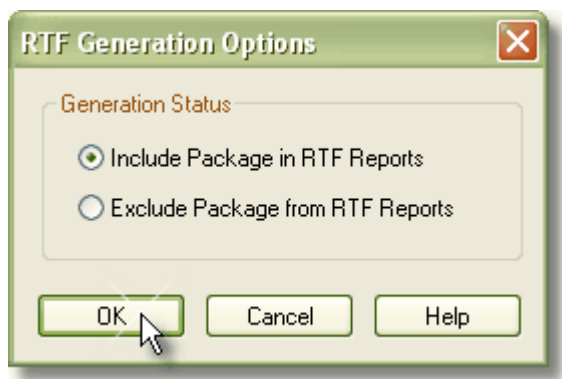
It is possible to exclude a particular package from any RTF reports by marking it for exclusion.

To Mark a Package for Exclusion

1. In the Project Browser, right click on the package to open the context menu.
2. From the *Documentation* submenu, select the *RTF Document Options* menu item.



3. In the *RTF Generation Options* dialog, select *Exclude Package from RTF reports*.
4. Press *OK* to save changes.



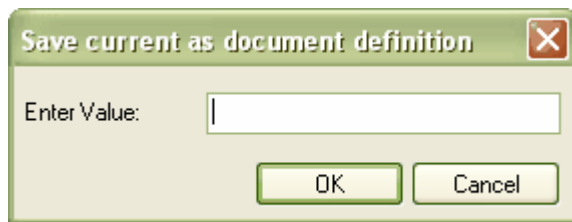
Note: By default, Packages are included in any RTF reports.

11.1.1.15 Save as Document

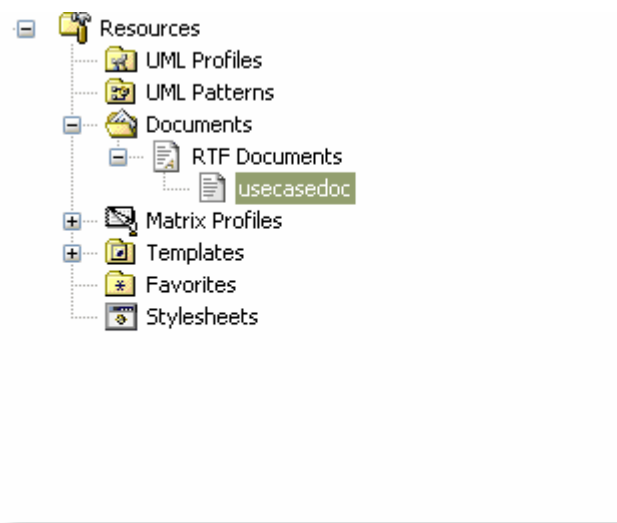
The Document feature allows the particular documentation configuration to be 'remembered', linking the loaded template within the RTF Report dialog to the current highlighted package. Perhaps a certain template is always used with a specific package, and there might exist multiple cases of documentation that need propagation; saving these as 'Documents' can ease document generation later.

To create and use Documents, follow these steps:

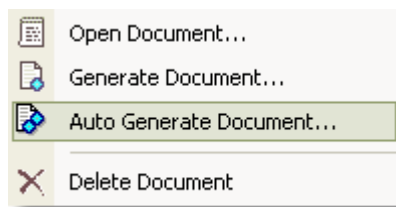
1. Open the RTF Report dialog (see [Create a Rich Text Document](#) for how to do this).
2. Press **Save as Document** to open the following dialog:



3. Supply a name for the document, and press **OK**. This document (the sample document was named 'usecasedoc') will be added to the resource view for easy future access.



3. To generate documentation from the resource view, right-click the desired document to see the following menu.



4. The available options are:
 - **Open Document**
Opens the corresponding .RTF file, as specified by the RTF template's 'Filename' property.

- **Generate Document**
Opens the RTF Report dialog, loaded with the specified template.
- **Auto Generate Document**
Documentation is generated, with the document located at the path specified by the template's 'Filename' property.
- **Delete Document**
Remove specified document.

11.1.2 Using MS Word

To further enhance and customize RTF documentation it is possible to create a custom master document, which can be used to add a table of contents, table of figures, headers and footers and to refresh linked files. In addition it is possible to create created with sustainable links to generated 'pieces' of EA output, pre-divided by EA using bookmarks.

See:

- [Open a Report in Microsoft Word](#)
- [Changing Linked Images to Embedded Images](#)
- [Bookmarks](#)
- [Other Features of Word](#)

11.1.2.1 Open a Report in Microsoft Word

To open an RTF file in MS Word, simply load Word and open the file as a normal document - Word will convert it for you. If Word is the default handler of RTF files, then double click on the output file to load up and view the report.

*Tip: If you have Word configured to view RTF files, you can also press **View** on the RTF Report dialog.*

11.1.2.2 Changing Linked Images to Embedded Images

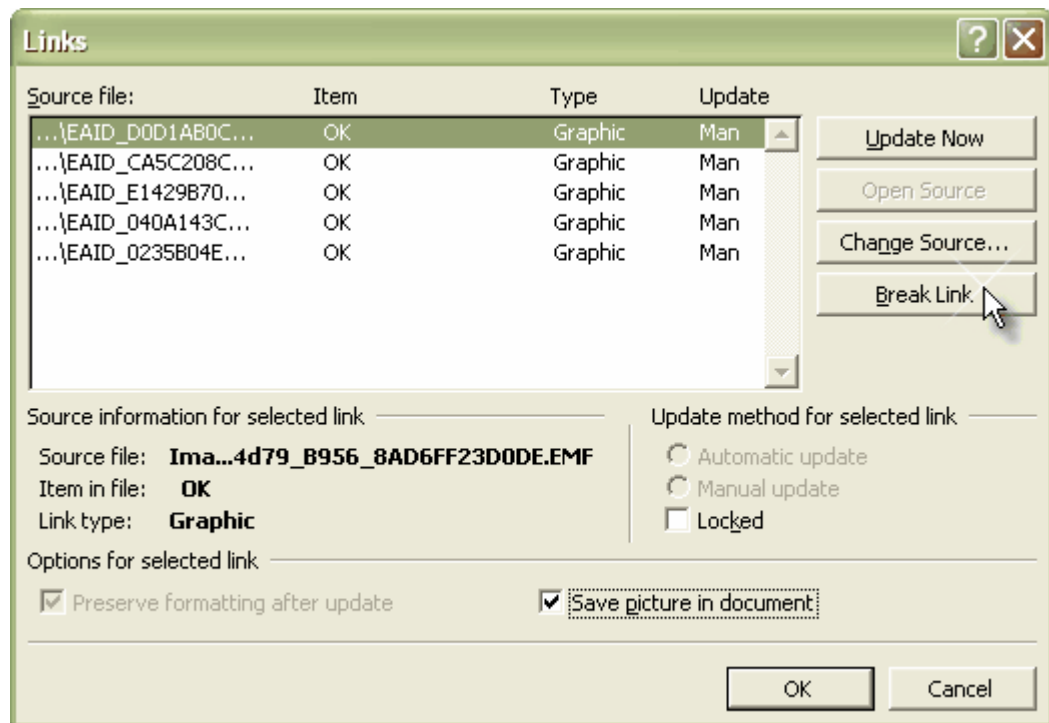
One of the options that is available when generating RTF documentation is the ability to store image files in a separate directory to the RTF document. If at a later stage it becomes desirable to embed the images into the RTF documentation, this is especially important when the document is to be distributed. If the images are stored in a separate directory recipients of document will see only the placeholder of images rather than the actual images.

If you import an RTF document into Word with the images not embedded into the document, you have the option of breaking the links to the images and saving the image in the document.

To Break Image Links in Word

1. Open an RTF file in Word.
2. Select **Links** from the **Edit** menu.
3. Highlight all links in the **Links** list.
4. Check the **Save Picture in Document** check box.
5. Press **Break Link**.
6. When prompted, press **Yes** to break links.

Word will break links and save copies of the images inside the document. You can distribute this document without the image directory.



11.1.2.3 Bookmarks

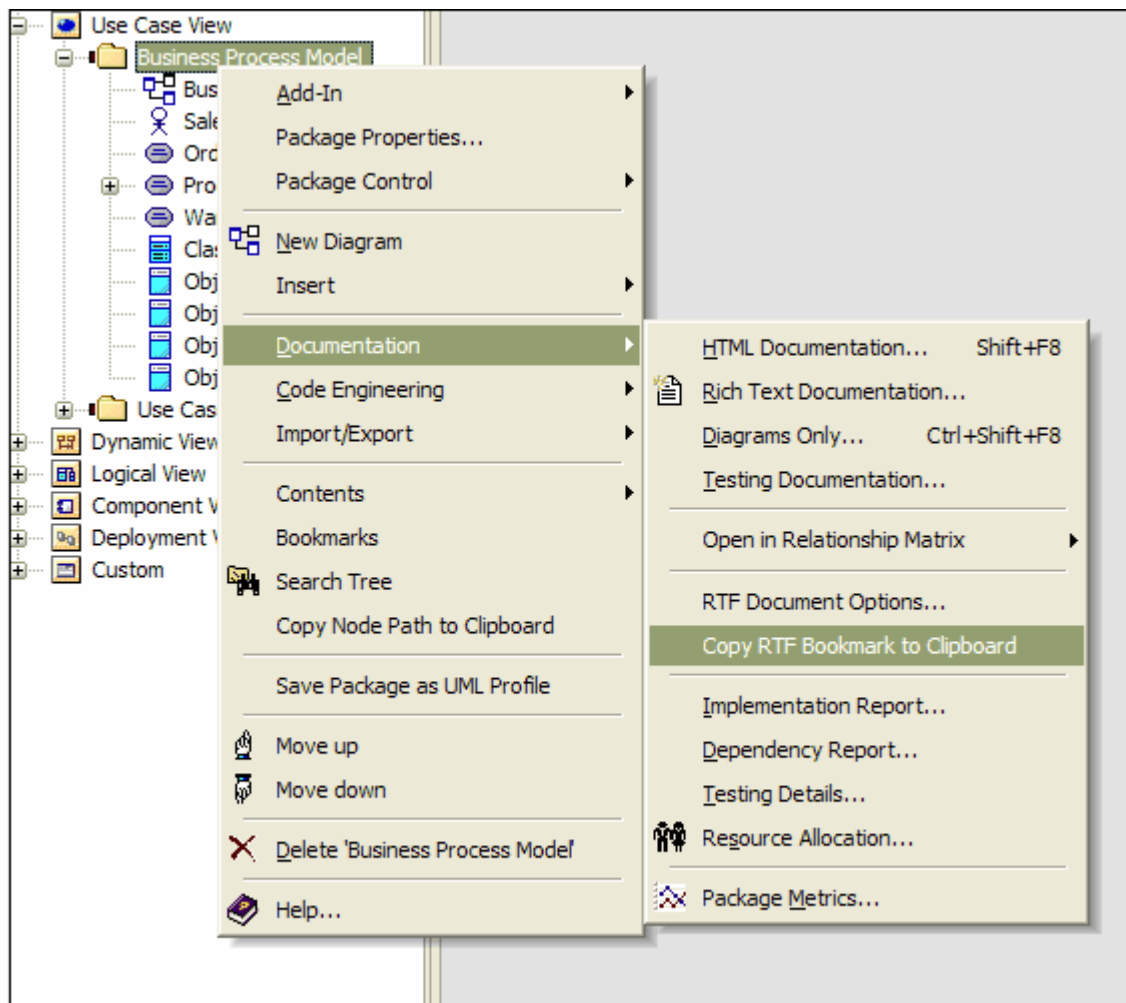
Bookmarks are markers that are automatically placed into your rich text document when it is generated. You can create a master document in Word, and link to sections of an Enterprise Architect report based on bookmarks. For an example, a Word document may have a section for a small part of your component model - using bookmarks you can generate a full component model, and then link into just one section of the report.

This way you can maintain a complex Word document from parts of EA reports. If you link into EA reports, then you can regenerate the report and refresh Word links to update the master document without having manually changed anything. More information relation to refreshing links can be found in the [Refresh Links](#) section

Bookmarks are created from package names. A bookmark applies from the beginning of a package to the end, and includes all child packages and elements underneath.

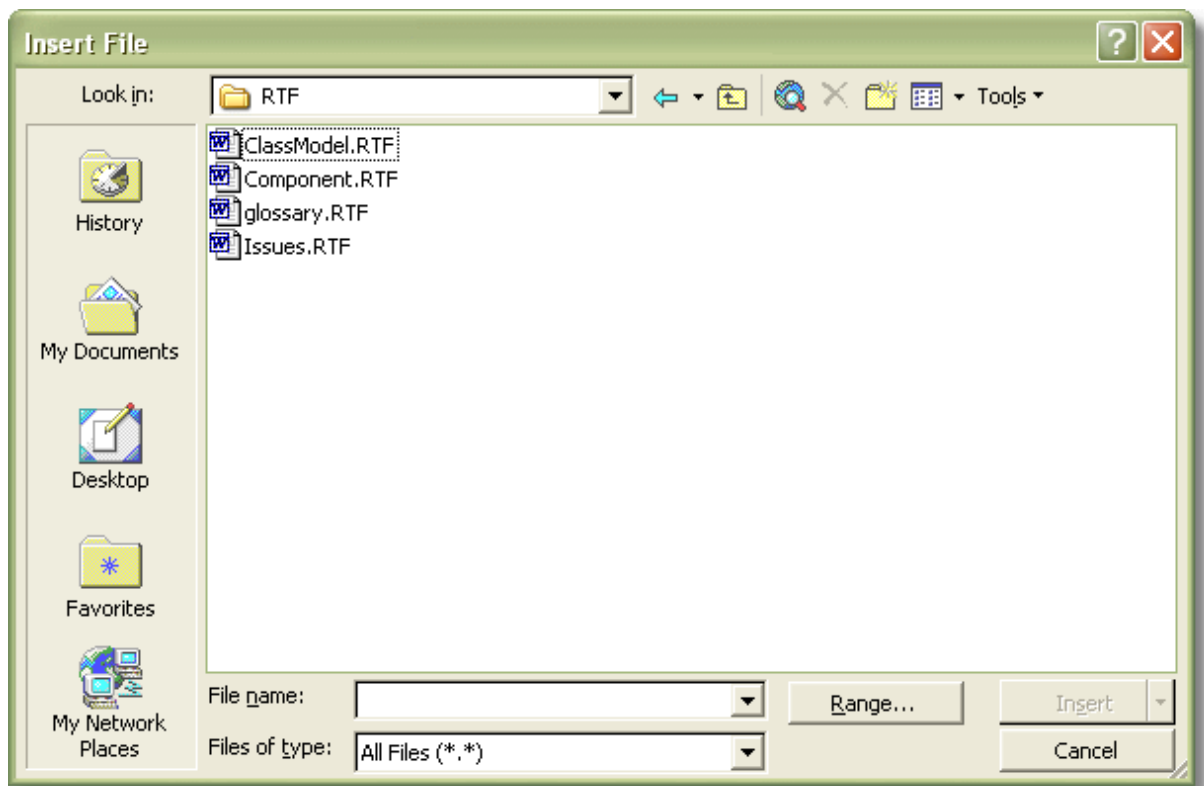
Bookmarking a Section of EA for RTF Documentation

1. In EA select the Package in the Project view which you wish to include into the documentation and right click it to bring up its context menu.
2. Mouse over the Documentation item and from the submenu select the Copy RTF Bookmark to Clipboard. This will paste your package into the clipboard to using in word.

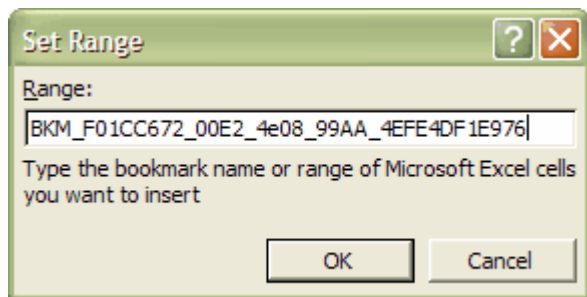


To Insert a Bookmarked Section of an EA RTF Document into Word

1. Open Word and position the cursor where you wish to insert the file.
2. Select *File* from the Word *Insert* menu.



3. In the *Range* cell type paste the information from the clipboard.



4. Press the *OK* button.

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"

11.1.2.4 Other Features of Word

Word offers a considerable number of document enhancement tools to complete your project documentation. Here are some of the things you can do with Word and EA generated RTF documentation:

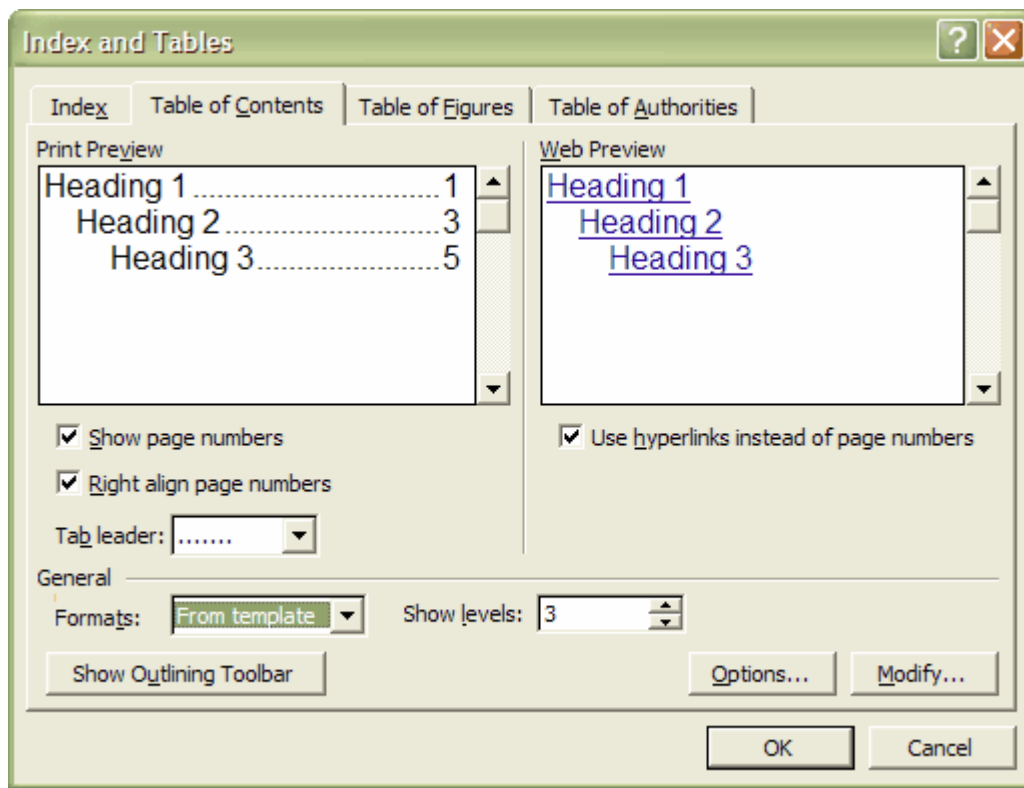
- [Add a Table of Contents](#)
- [Add a Table of Figures](#)
- [Add Headers and Footers](#)
- [Manipulating Tables in Word](#)
- [Refresh Linked Files](#)

Tip: Enterprise Architect provides the basic content for your document - use Word to add the presentation and linkages.

11.1.2.4.1 Add Table of Contents

Among the features of MS word which can be incorporated into generated EA reports is the option to include a table of contents. A table of contents can be used to aid navigation of documentation and enhance the readability of EA RTF reports. This option provide hyperlinks to the diagrams included in the RTF documentation in the electronic version and page numbers for both the printed and electronic documentation. To include a Table of Contents in the RTF documentation use the following instructions:

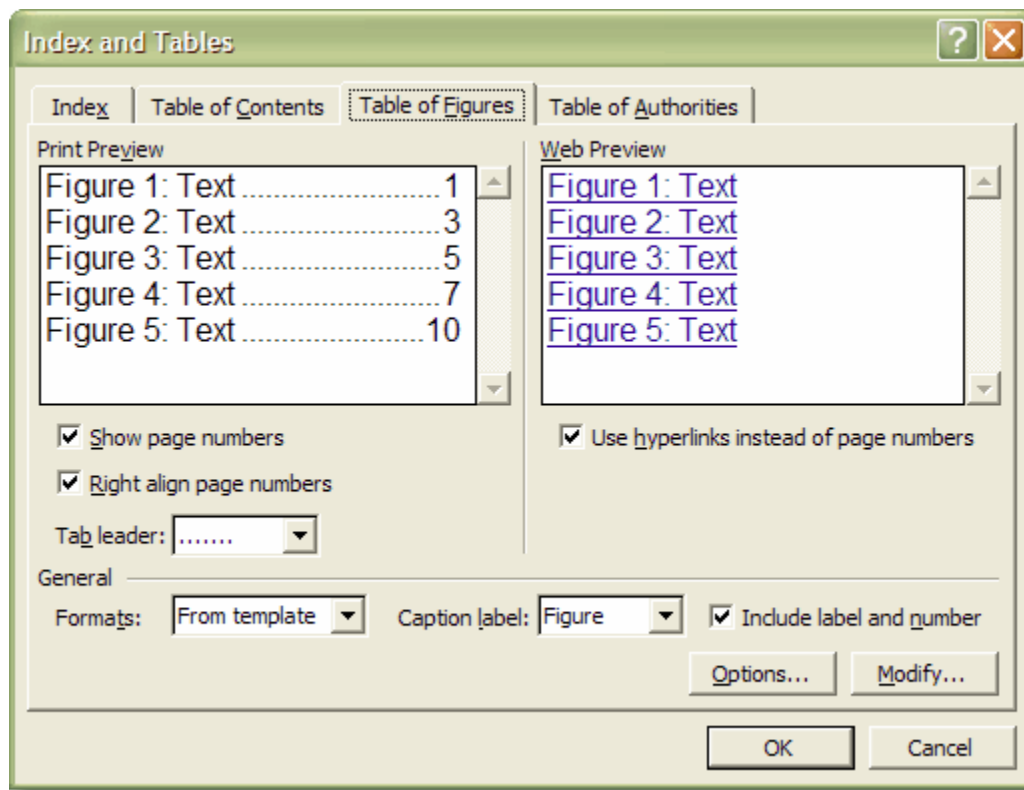
1. Open the EA RTF report that you want to add a Table of Contents in MS Word.
2. From the *Insert* menu mouse over the *Reference* option and from the submenu select the *Index and Tables* option.
3. Open the Table of Contents Tab to set the options that are available for formatting the table of contents. The format of the table of contents is dependant on the level/s of headings created when the RTF is generated, to set the heading style for details in EA RTF documentation refer to the RTF options section



11.1.2.4.2 Add Table of Figures

Among the features of MS Word which can be incorporated into generated EA reports is the option to include a table of figures. A table of figures can be used to aid the navigation of the documentation and enhance the readability of EA RTF reports. This option 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 Figures in the RTF documentation use the following instructions:

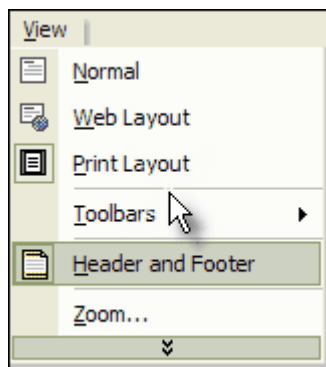
1. Open the EA RTF report that you want to add a Table of Figures in MS Word.
2. From the *Insert* menu mouse over the *Reference* option and from the submenu select the *Index and Tables* option.
3. Open the *Table of Figures* tab to set the options that are available for formatting the table of figures.



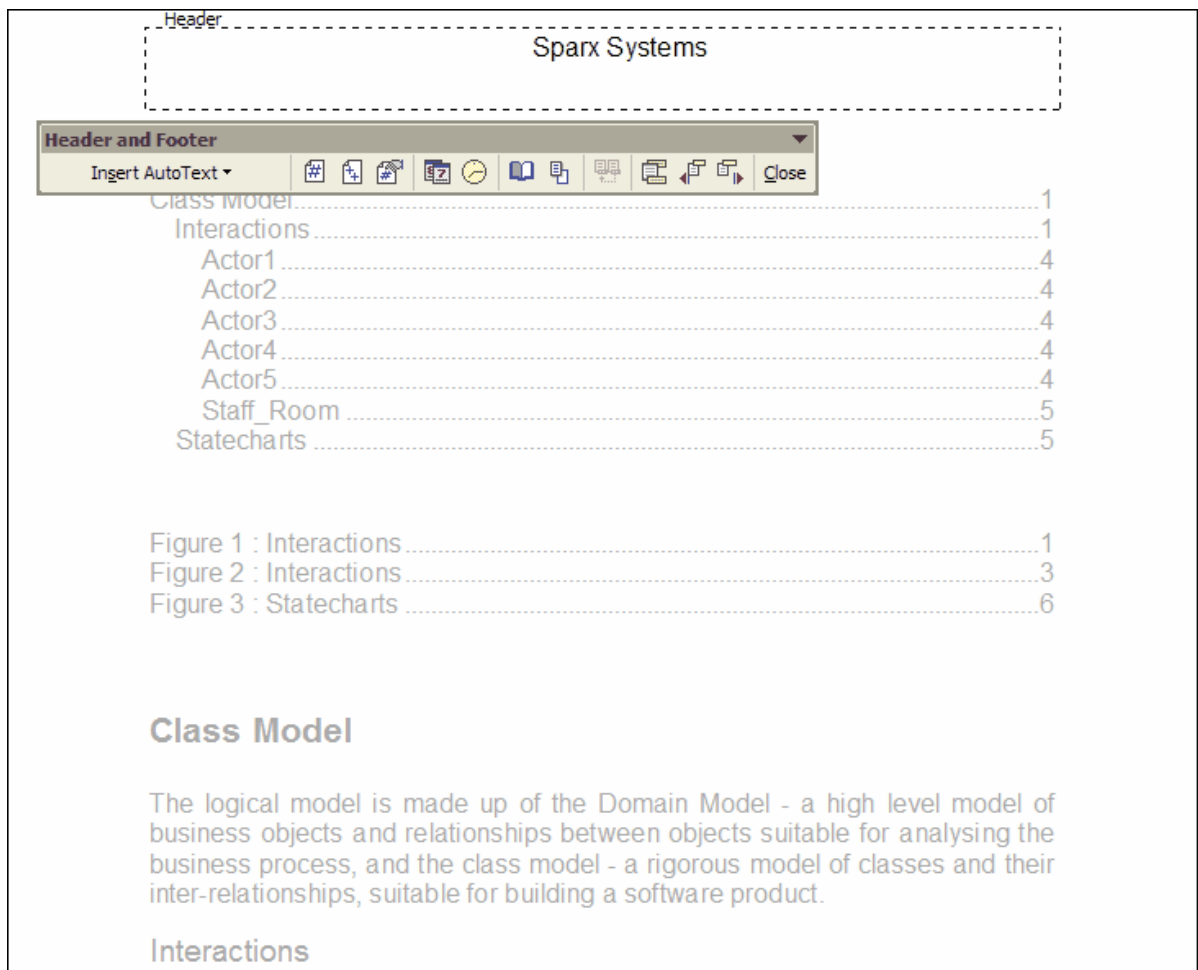
11.1.2.4.3 Add Headers and Footers

Among the features of MS word which can be used to enhance the appearance of EA RTF reports is the ability to add Headers and footers to the documentation. To include Headers and Footers in the RTF documentation use the following instructions:

1. Open the EA RTF report that you want to add Headers and Footers in MS Word.
2. From the *View* menu select the *Header and Footer* Option.



3. This will allow the user to enter information into the header section and the footer section of the RTF Documentation.



11.1.2.4.4 Manipulating Tables in Word

When generating RTF documentation from EA, tables are included when items such as *Attributes* and *Methods* are selected in the *For each Object* section in the Rich Text Format Report dialog. MS Word offers several levels of customisation for tables and can be used to tidy the formatting of the tables in situations where the margins of the table exceed the dimensions of the page size selected in Word for printing.

Resizing Tables

When the amount of detail for a documented item such as an attribute or operation exceeds the margins of the page in MS Word it is necessary to manually resize the table in order to view all of the details. To manually resize the table use the following instructions:

1. Select the table which exceeds the margin size.
2. Mouse over the border of the table until the mouse pointer changes into the icon shown below.

Message Attributes

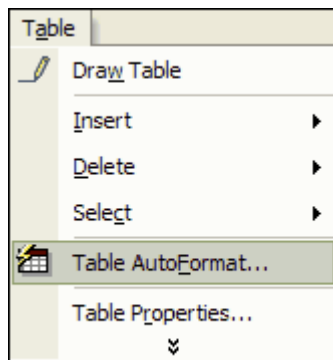
Attribute	Type	Notes
<u>sentTime</u>	protected : <i>Date</i>	
<u>receivedTime</u>	protected : <i>Date</i>	
body	protected : <i>String</i>	

3. Drag the cursor to the left to reduce the width of the table and then use the *File | Print Preview* to confirm that the table borders are within the page margins.
4. Resize all of the tables that overhang the margins of the page by using the steps detailed above.

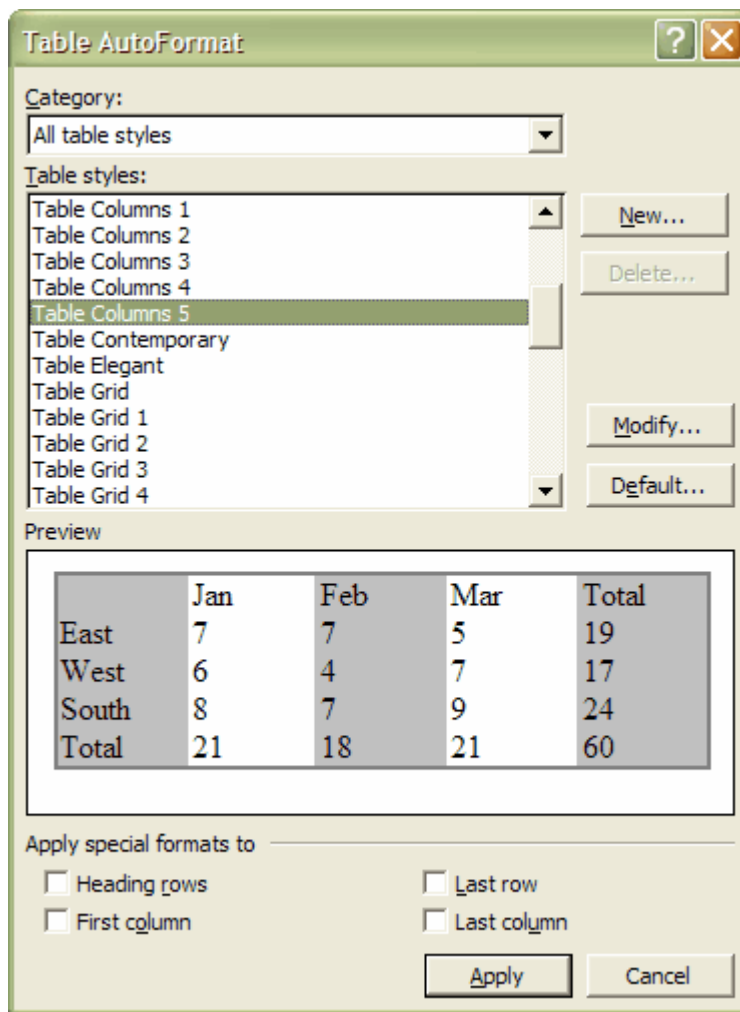
Applying Styles to Tables

One of the customizable properties of MS Word when working with tables is the ability to apply a style to a table which allows the user to rapidly change the appearance of the table to achieve this effect use the following instructions:

1. Open the EA RTF report that you want to change the table styles.
2. Locate the table that you wish to adjust the appearance and select the table.
3. From the *Table* Menu select *Table Auto Format...*



4. This will bring up the Table Autoformat dialog, from here the user may specify a predefined table style from the list of Table styles, or create a new style by pressing the *New* button. The Table styles defined in the Table Autoformat only applies to one table at a time so the user will need to apply the style to each table created individually.

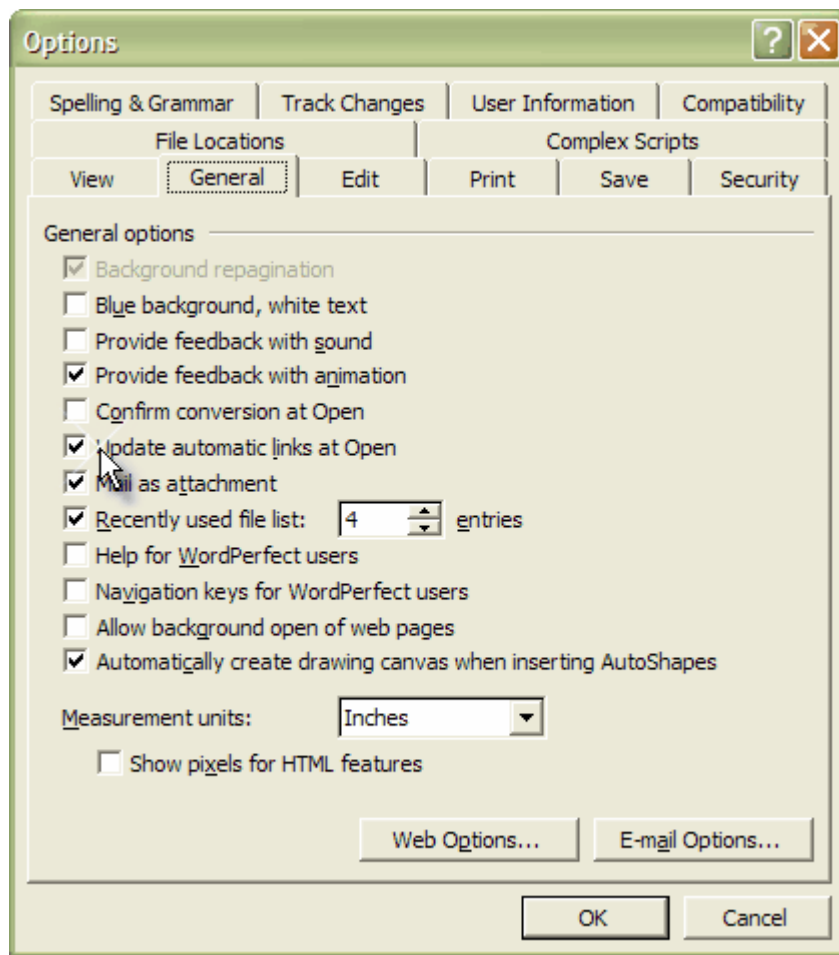


11.1.2.4.5 Refresh Links

If you link into EA reports, then you can regenerate the report and refresh Word links to update the master document without having manually changed anything.

To ensure that links are refreshed in the master document the Update automatic links at Open option must be checked in MS word. To ensure that this setting is established use the following instructions:

1. From within MS Word go to the *Tools* menu and select *Options*.
2. From the *General* tab ensure that the *Update automatic links at Open* is checked.



11.1.3 Other Documents

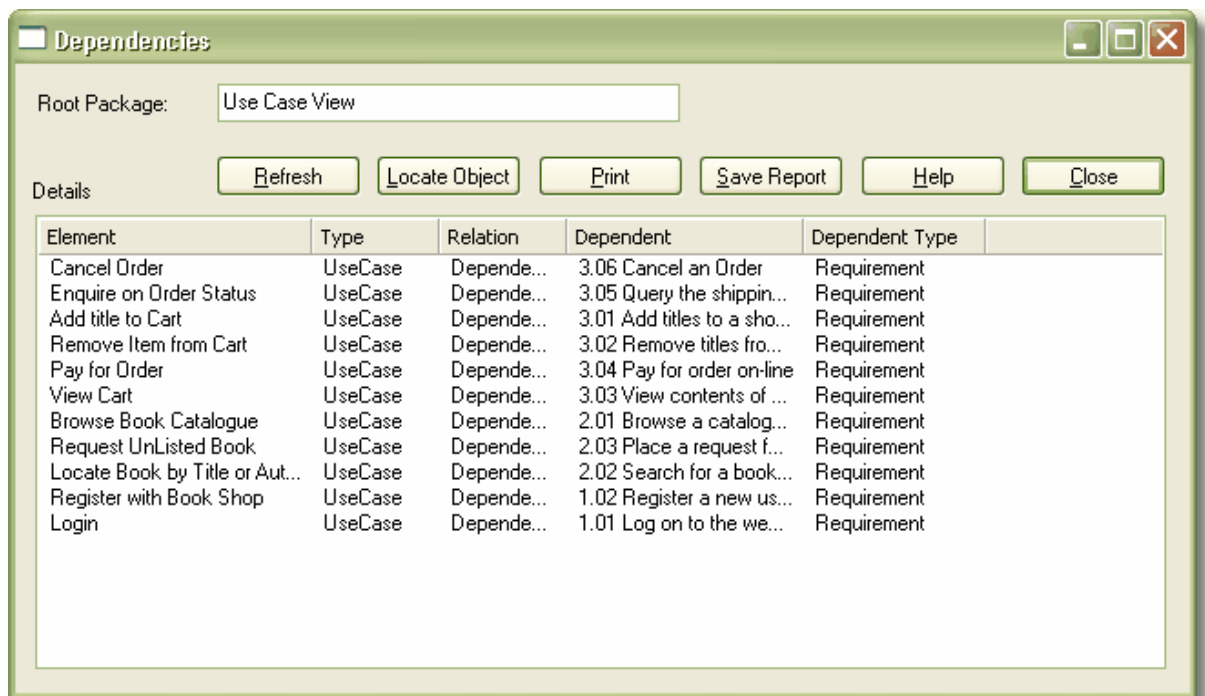
Enterprise Architect has other RTF based documentation that you can output:

- [Dependency Report](#)
- [Implementation Report](#)
- [Set Target Types Dialog](#)
- [Testing Report](#)

11.1.3.1 Dependency Report

To view a dependency report, follow these steps:

1. In the Project Browser, right click on the package you wish to report on (the report includes all sub-packages as well) to open the context menu.
2. From the *Documentation* submenu, select *Dependency Report...*
3. The *Dependencies* dialog displays a list of all elements that implement other elements in the provided list, together with the elements that are dependent. Save or print the results if required.

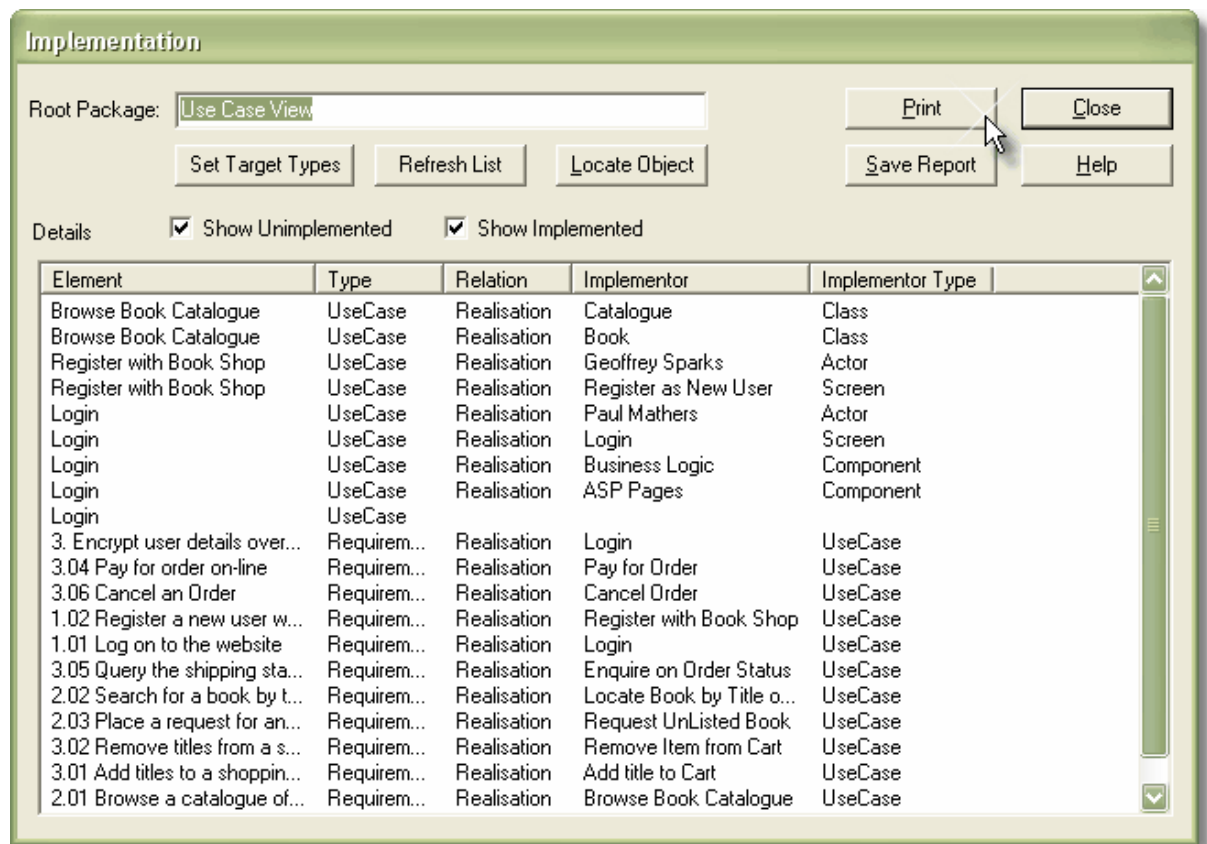


Control	Description
Root Package	The root package. All elements and packages under this will appear in the report.
Locate Object	Locate the element selected in the report list in the Project Browser.
Refresh	Run the report again.
Dependency Details	List of dependency details - lists elements in the current hierarchy and elements that implement them.
Print	Print the list.

11.1.3.2 Implementation Report

To view an implementation report, follow these steps:

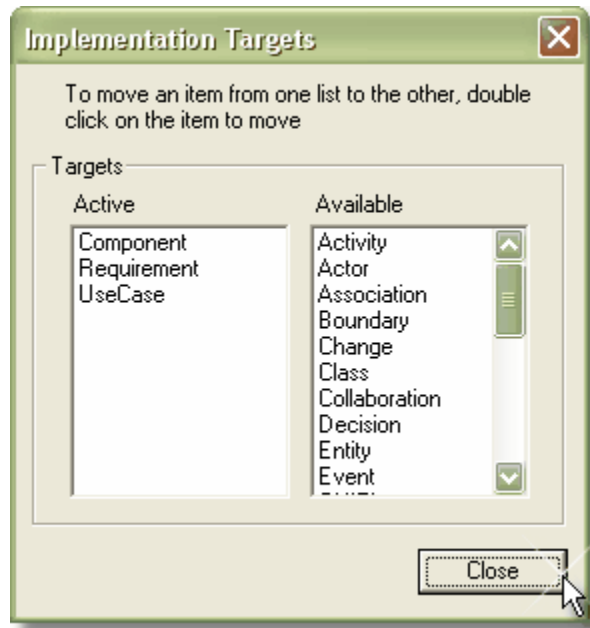
1. In the Project Browser, right click on the package you wish to report on (the report includes all sub-packages as well) to open the context menu.
2. From the *Documentation* submenu, select *Implementation Report...*
3. The *Implementation* dialog displays a list of all elements that require implementers in the provided list, together with the elements that have been connected to those elements by a "Realization" (Implementation) type link. Save or print the results if required.



Control	Description
Root Package	The root package. All elements and packages under this will appear in the report.
Show unimplemented	Show non-implemented elements. Implemented elements are those that don't have any other element to realize them (eg. a use case has no component or class to implement the use case behavior)
Show Implemented	Show implemented elements. These are elements that do have some element associated with them in a realization relationship. For example a use case has a component that implements it.
Locate Object	Locate the element selected in the report list in the Project Browser
Refresh	Run the report again
Implementation Details	List of implementation details - lists elements in the current hierarchy and elements that implement them
Print	Print the list
Set Target Types	By default EA only reports on Use Case, Requirement and a couple of other types. You can use this option to set the list of types you want to report on. See further information in the Set Target Types Dialog topic.

11.1.3.3 Set Target Types Dialog

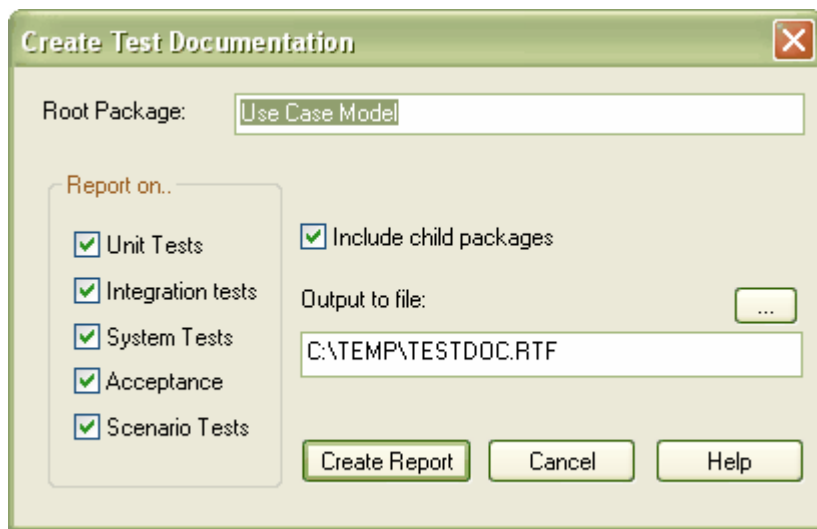
The *Set Target Types* dialog is accessed by pressing *Set Target Types* on the *Implementation* dialog. This dialog allows you to set the types of elements that will appear in the report as requiring implementation. Double click on an element in either list to move it to the other list.



11.1.3.4 Testing Report

To view a testing report, follow these steps:

1. In the Project Browser, right click on the package you wish to report on (the report can include all sub-packages as well, if 'Include child packages' is selected) to open the context menu.
2. From the *Documentation* submenu, select *Testing Documentation....*
3. The *Create Test Documentation* dialog specifies which test types will be documented in the report, and the output file location. The test data originates in the docked testing window, where tests are created and attached to a respective object.



11.1.4 RTF Style Templates

The *RTF Style Editor* allows you to edit the RTF associated with various sections of the RTF Report facility in Enterprise Architect. You would typically use this functionality to customize a report look and feel for your company or client.

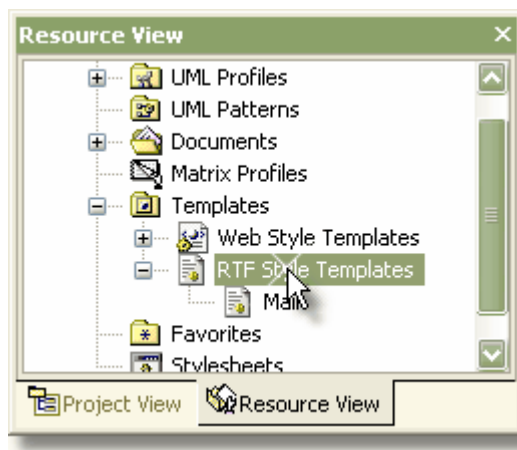
Create or Edit RTF Style Templates

1. In the *Resource View*, expand the *Templates* folder.
2. If you wish to edit an existing template, expand the *RTF Style Templates* tree, double click on the template name, or right click and select *Modify RTF Style Template* from the context menu. The *RTF Style Editor* will open. See below for further details.

-OR-

3. To create a new template, right click on *RTF Style Templates* and select *Create RTF Style Template* from the context menu.
4. Enter a name for the new template when prompted to do so. The *RTF Style Editor* will open. See below for further details.

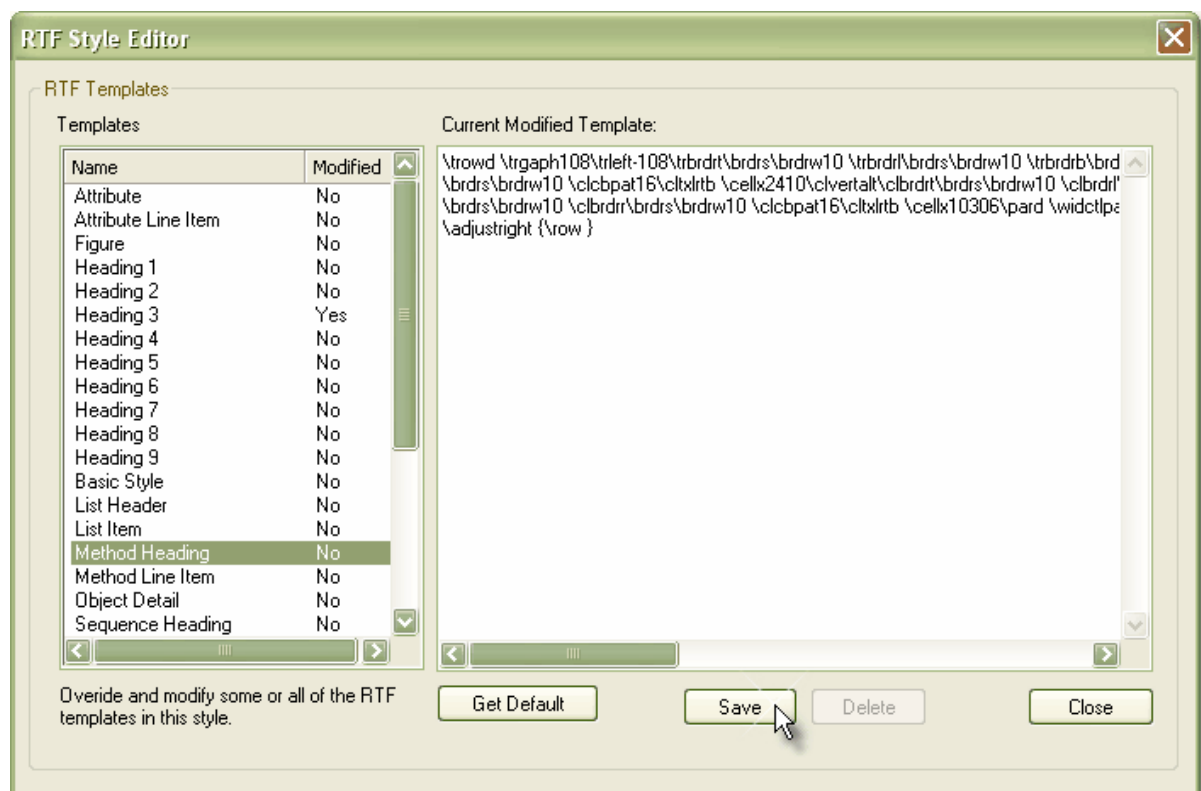
Tip: To delete a template, right click on it and select *Delete RTF Style Template* from the context menu.



The *RTF Style Editor* (shown below) contains a list of all available RTF fragments for modification and customization.

Each fragment typically contains RTF plus one or more special tag names that EA will replace with information during generation. Currently you cannot alter the content within the tag names, but you can omit a complete tag by removing it, or alter its basic display properties in the surrounding RTF.

Special tag names are delimited by "#" characters - eg. #NOTES#



- *Get Default* retrieves the default EA template for the currently selected template item in the left hand list.
- *Save* saves your version of the template for this style only.
- *Delete* removes your modified version of the template - which will cause EA to use the default template during report generation.

To select a template during generation - use the *Style* drop list on the *Rich Text Format Report* dialog. Once a style is selected, EA will apply that to the current report. Select <default> for the inbuilt style.



Tip: You can also [alter the custom language settings](#).

11.1.5 Custom Language Settings

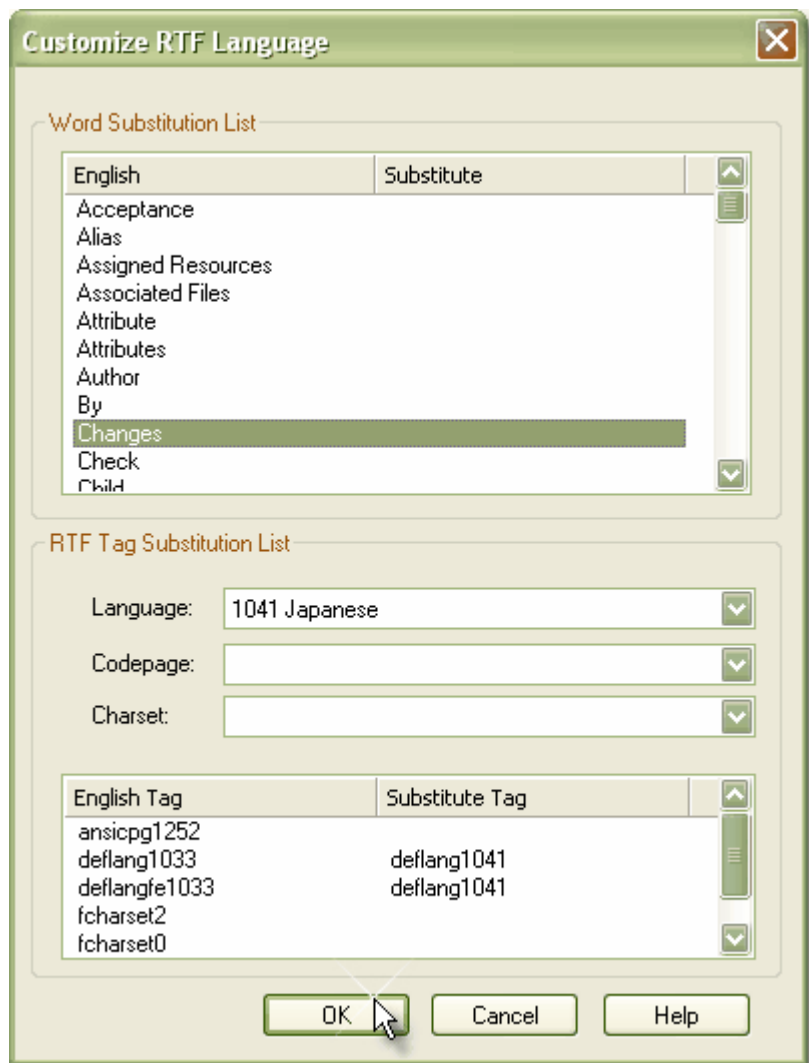
If you export documents in RTF format from EA in languages other than English, you can customize the standard set of keywords that EA uses when generating RTF. This makes it much easier to generate documentation that is readily acceptable in your country or locale.

You can also customize the codepage, default language ID and character set to use.

To do this - you will need to set up a list of word substitutions. For instance, where EA would include the word "Figure", you can specify another word to replace it that is either in your language or more meaningful to your readers.

To Set Up Substitutions

1. In the Project Browser, right click on a package to open the context menu.
2. Select *Rich Text Documentation* from the *Documentation* submenu.
3. In the *Rich Text Format Report* dialog, press *Adjust* (in the *Language* section). This opens the *Customize RTF Language* dialog (below)
4. Double click an item to set or clear its substitute word.
5. When you have finished, press *OK*.



To Set Up Codepage and Character Set

1. From the drop down lists, select the *Language*, *Codepage* and *Character set* that most closely matches your Locale.
2. If required, modify the *Substitute Tags* by double clicking on each and manually setting the value (for advanced use only).
3. To clear the substitution list, double click on each item in turn and delete the substitute value.
4. When you are happy with the settings, press *OK* to save.

Now when you generate RTF documents, the substitute tags will be used in the output.

Tip: Although intended for languages other than English, you can also tailor the look and feel of the RTF documents by substituting your keywords for the default ones used by EA.

11.2 HTML Reports

Enterprise Architect allows export of an entire model or a single branch of the model to HTML Web pages. The HTML report provides a convenient frames-based structure navigator and content explorer. In addition, much of the output is hyperlinked to related sections within the model.

The current implementation is based on internal templates and generated Java script. The ability to edit the templates will be added in a future version of Enterprise Architect.

Tip: You can create [Web Style Templates](#) to customize your HTML output.

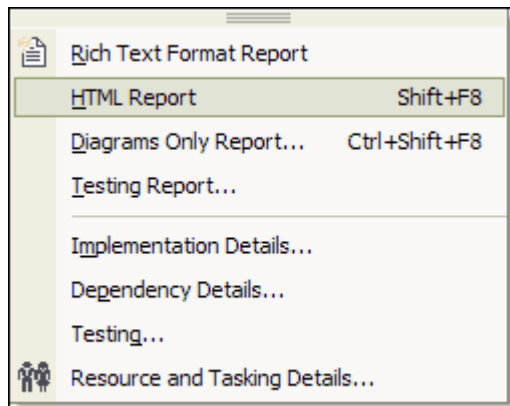
See:

- [Creating an HTML Report](#)
- [The Generate HTML Report Dialog](#)
- [Making the Output Available on the Web](#)
- [Web Style Templates](#)

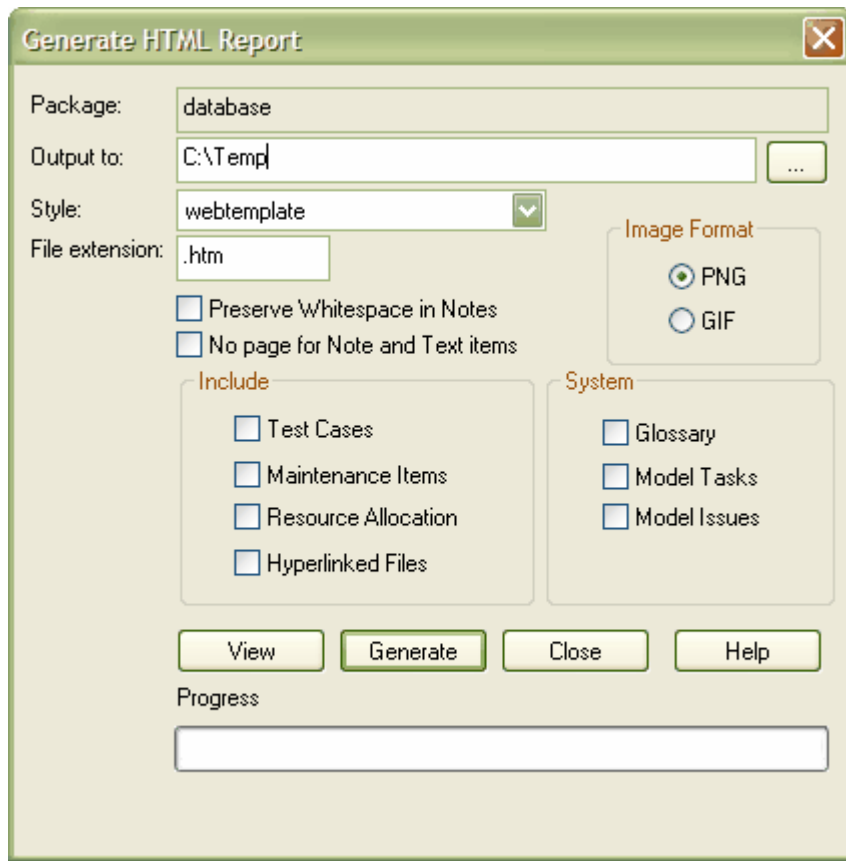
11.2.1 Creating an HTML Report

To Create a HTML Report

1. In the Project Browser, right click on the root package for the report - all child packages will be included in the output.
2. From the *Documentation* submenu within the context menu, select *HTML Documentation*. This opens the *Generate HTML Report* dialog.

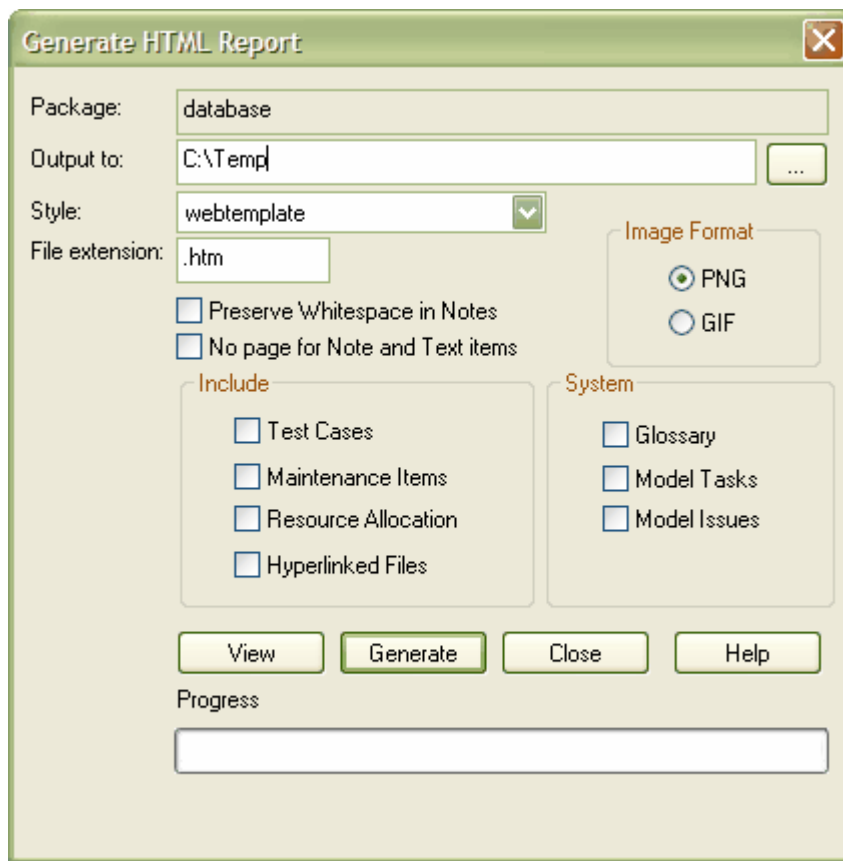


3. Select an *Output* directory for your report. Set any other required [options](#).
4. Press *Generate* to generate the report - the progress meter will track total percentage complete.
5. Once complete, press *View* to launch your default HTML viewer and view the web pages.



11.2.2 The Generate HTML Report Dialog

The *Generate HTML Report* dialog is used to generate documentation about your model in HTML format. There are various settings to choose from to control the output - see below.



Element	Description
Package	The package you are creating documentation for.
Output to	The directory your documentation will be saved to.
Style	Select a web style template to apply to your documentation (optional).
File extension	Specify the file extension you want your HTML documentation files to have - the default is .htm.
Image Format	Select the image file format you want your images to be saved as - either PNG or GIF.
Preserve White space in Notes	Select to preserve existing white space in your notes, deselect to remove white space.
No page for Note and Text items	Select if you do not wish to include a page for your notes and text items in the HTML report.
Include - <ul style="list-style-type: none"> • Test Cases • Maintenance Items • Resource Allocation • Hyperlinked Files 	Select to include any of these areas of your model in your report.
System - <ul style="list-style-type: none"> • Glossary • Model Tasks • Model Issues 	Select to include any of these areas of your model in your report.
View	View your report - note that your report can only be viewed after it has been generated (see below).
OK	Press to generate a report according to the settings you have selected.
Close	Close the dialog.

11.2.3 Making the Output Available on the Web

The web report produced is compatible with any standard web server - either on Unix or Windows platform. Simply bundle up the entire output directory and place within the context of your web server. All path names should be relative and case sensitive.

11.2.4 Web Style Templates

The *HTML and CSS Style Editor* allows you to edit the HTML associated with various sections of the HTML Report facility in Enterprise Architect. You would typically use this functionality to customize a report look and feel for your company or client.

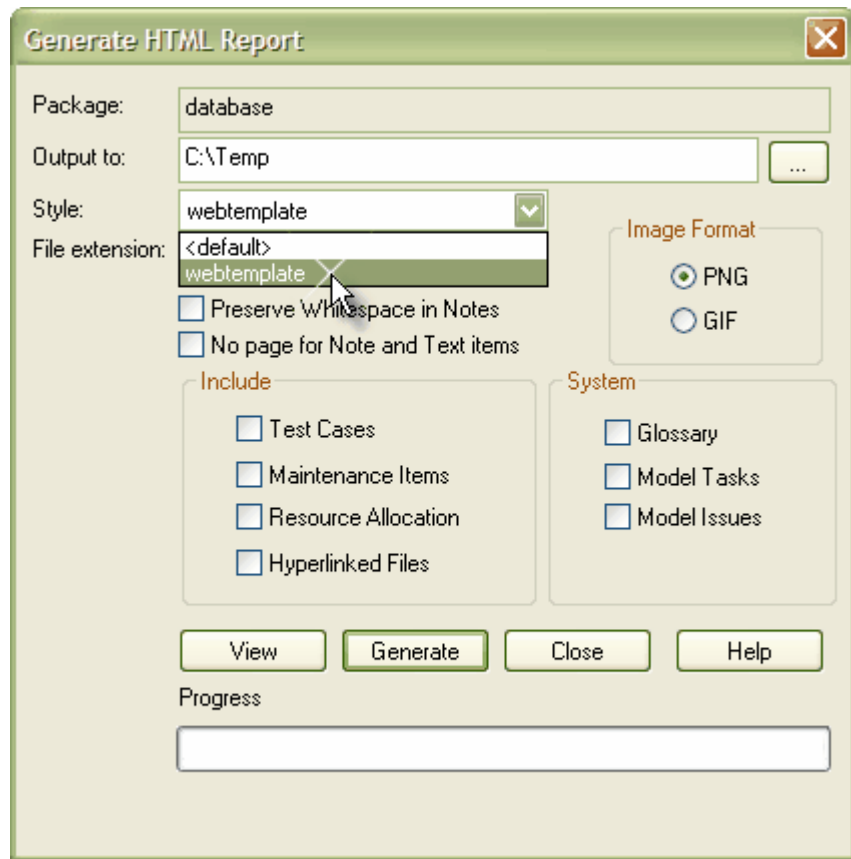
Create or Edit Web Style Templates

1. In the *Resource View*, expand the *Templates* folder.
 2. If you wish to edit an existing template, expand the *Web Style Templates* tree, double click on the template name, or right click and select *Modify HTML Style Template* from the context menu. The *HTML and CSS Style Editor* will open. See below for further details.
- OR-**
3. To create a new template, right click on *Web Style Templates* and select *Create HTML Template* from

the context menu.

4. Enter a name for the new template when prompted to do so. The *HTML and CSS Style Editor* will open. See below for further details.

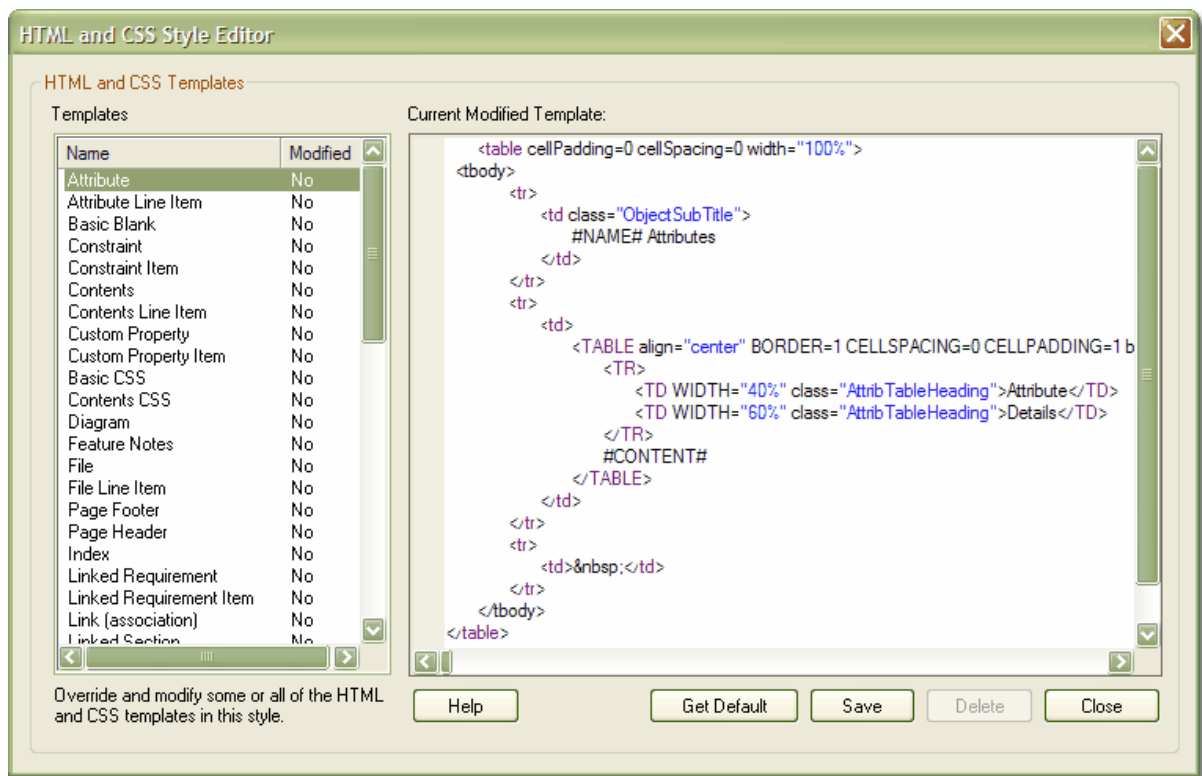
Tip: To delete a template, right click on it and select *Delete HTML Template* from the context menu.



The *HTML and CSS Style Editor* (shown below) contains a list of all available HTML fragments for modification and customization.

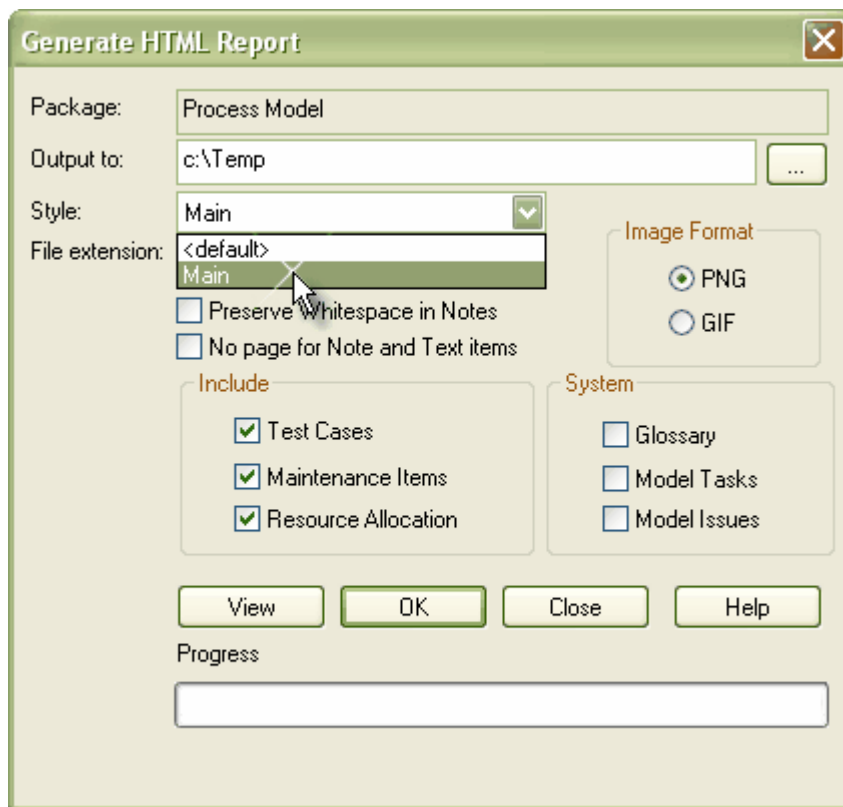
Each fragment typically contains HTML plus one or more special tag names that EA will replace with information during generation. Currently you cannot alter the content within the tag names, but you can omit a complete tag by removing it, or alter its basic display properties in the surrounding HTML.

Special tag names are delimited by "#" characters - eg. #NOTES#



- **Get Default** retrieves the default EA template for the currently selected template item in the left hand list.
- **Save** saves your version of the template for this style only.
- **Delete** removes your modified version of the template - which will cause EA to use the default template during report generation.

To select a template during generation - use the **Style** drop list on the **Generate HTML Report** dialog. Once a style is selected, EA will apply that to the current report. Select <default> for the inbuilt style.



Note: Each time EA generates the web report it overwrites these files, so you will need to back up your modified versions and copy them back in after every update.

11.3 Virtual Documents

You can create "virtual documents" in EA. This allows you to set up document objects and insert any packages you wish into the document. For example, you could create a document which includes a Sequence Diagram package and a Code Engineering packages (these packages are included in the [EA example model file](#)).

To create a virtual document, you first need to [Create a document object](#), [Add Packages to Your Document Object](#), then [Generate the Document](#).

You can include any combination of packages, and [add](#) or [delete](#) packages as required. You can also [Rearrange the Package Order](#) within the documentation.

The document will obtain its contents dynamically - ie. you don't need to update the document if you make a change to one of the packages included in it.

Tip: You can create as many document objects as you wish, for as many combinations of packages as required. You can include the same package in multiple objects.

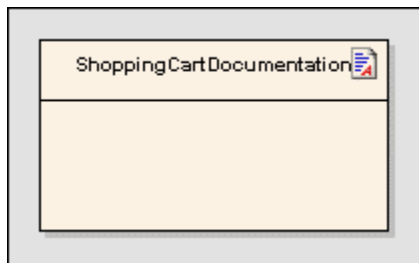
11.3.1 Create a Document Object

To create a "virtual document" you need to create a document object. Follow the steps below:

1. Create a new class diagram.
2. From the UML Toolbox, drag the **Class** icon onto the diagram to create a new class. Name the class something appropriate - in this example we will be including documentation relevant to the shopping cart components of the model, so we will call it "ShoppingCartDocumentation".
3. In the **Stereotype** field of the Class Properties dialog, enter **Model Document**. This is crucial to the virtual documentation process.

Note: The stereotype must be **Model Document** in order for the process to work correctly.

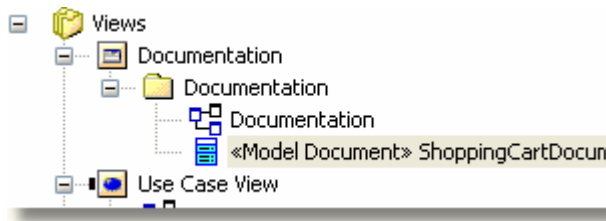
4. Press **OK** - this will create your document class.



Notice the small icon on the top right-hand corner of your class which indicates that this is a document object.

Tip: Resize your *ShoppingCartDocumentation* as required for neatness.

Your document object will appear in the Project Browser as shown below:

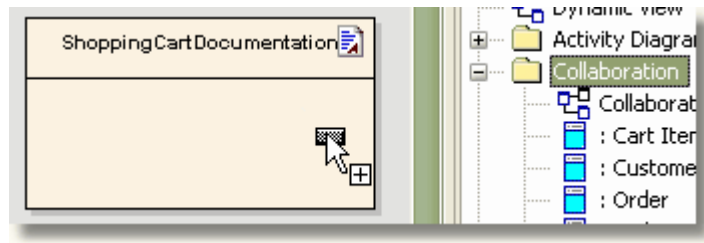


The next step is to [add packages to your document](#).

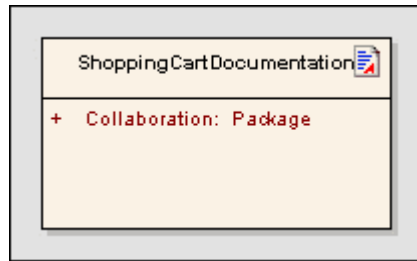
11.3.2 Add Packages to Your Document Object

To add packages to your document object, follow the example below. As this document object is called *ShoppingCartDocumentation*, this example will explain how to add shopping cart related packages to the object.

1. Keeping your Documentation diagram open, find a package you want to add to the documentation in the Project Browser - for example, the Collaboration package in the Dynamic view from the [EA example model file](#).
2. Drag and drop the Collaboration package from the Project Browser onto the document object as shown below:



The title of the package will appear in the document object, as shown below:



This means that the Collaboration package will now be included in the document when you [generate it](#). You can add as many packages from as many different views as desired using the above method.

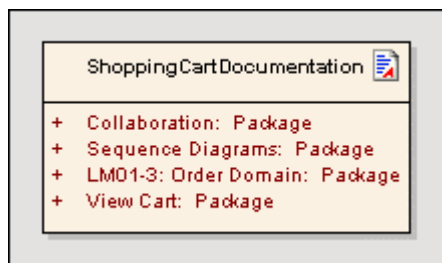
The next step is to [generate your document](#). You can also [rearrange](#) or [delete packages](#) if you wish.

11.3.3 Rearrange the Package Order

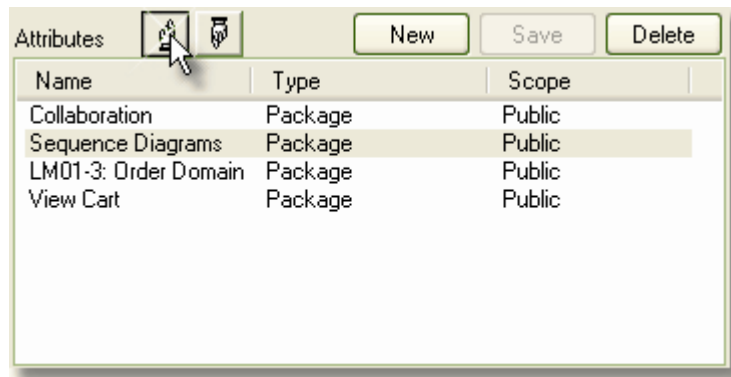
If you have more than one package in your document object, you can rearrange the order by following this example.

Note that the example now has 4 packages included from the [EA example model file](#):

- Collaboration Package (from the Dynamic View)
- Sequence Diagrams Package (from the Dynamic View)
- LM01-3: Order Domain Package (from the Logical View)
- View Cart Package (from the Logical View)



1. Right click on the document object and select **Attributes** from the context menu. This will open the **Attributes** dialog.
2. On the **Attributes** list, use the **Up** and **Down** buttons to change the order the packages will be included in the documentation.



3. When you are satisfied with the order of your packages, press **OK**.

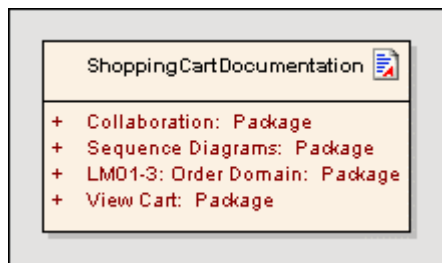
See also: [Delete a package from your document object](#) and [Generate the Document](#).

11.3.4 Delete a Package from Your Document Object

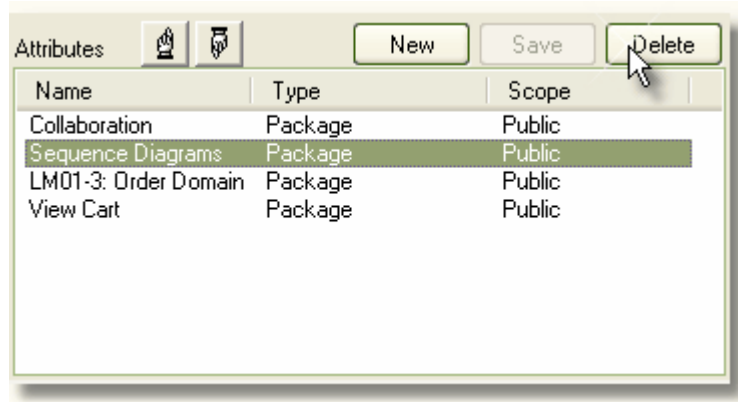
You can delete a package from your document object by following this example:

Note that our example now has 4 packages included from the [EA example model file](#):

- Collaboration Package (from the Dynamic View)
- Sequence Diagrams Package (from the Dynamic View)
- LM01-3: Order Domain Package (from the Logical View)
- View Cart Package (from the Logical View)



1. Right click on the document object and select **Attributes** from the context menu. This will open the **Attributes** dialog.
2. On the **Attributes** list, select the package you wish to delete.
3. Press **Delete** to remove the package from the document.

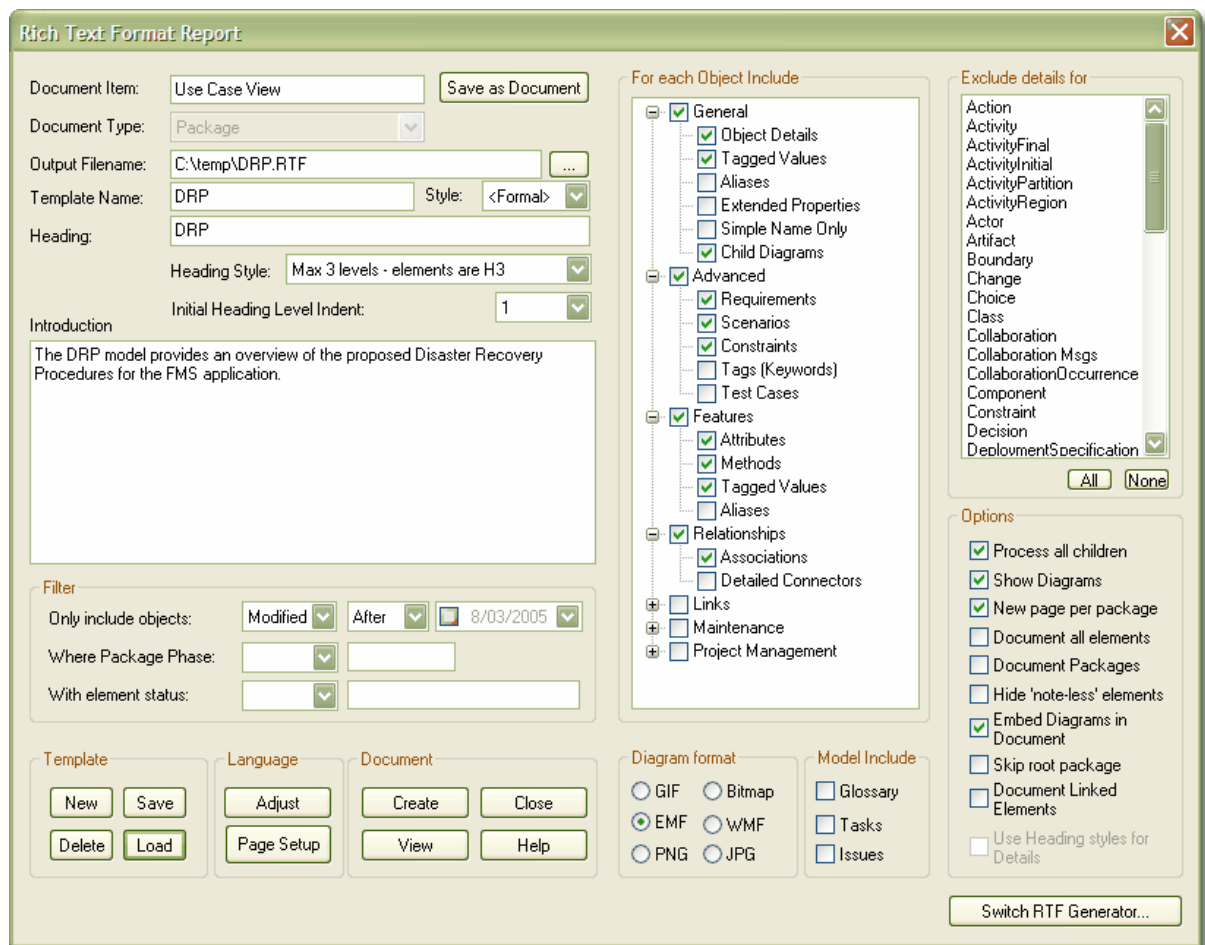


See also: [Rearrange the Package Order](#) and [Generate the Document](#).

11.3.5 Generate the Document

To generate the documentation listed in the document object, follow the steps below:

1. Select the element on the diagram.
2. From the *Element* menu, select *Documentation*. This will open the *Rich Text Format Report* dialog (shown below).



3. Set the options for this as required - see [The RTF Report Dialog](#) and related topics for further information on these settings.
4. Press **Create** to create the documentation.
5. Press **View** to view the documentation. For the example given in this section, the document will include the following packages from the [EA example model file](#):
 - Collaboration Package (from the Dynamic View)
 - Sequence Diagrams Package (from the Dynamic View)
 - LM01-3: Order Domain Package (from the Logical View)
 - View Cart Package (from the Logical View)

See also: [Rearrange the Package Order](#) or [Delete a Package from Your Document Object](#).

Part



12 Automation and Scripting

Automation provides a way for other applications to access the information in an EA model using Windows OLE Automation (ActiveX). Typically this will involve scripting clients such as MS Word or Visual Basic. This section describes how to access the Enterprise Architect automation interface.

See:

- [The Automation Interface](#)
- [The Project Interface \(XML\)](#)
- [Add-ins](#)

For the key methods available see:

- [The Repository Functions](#)
- [The Project Functions](#)

For Details of the main elements within the Automation Interface see:

- The [Repository](#) which represents the model as a whole and provides entry to model packages and collections;
- [Elements](#) which are the basic structural unit (eg. Class, Use Case and Object)
- [Element Features](#) which are attributes and operations defined on an element
- [Diagram](#) - the visible drawings contained in the model
- [Connectors](#) - relationships between elements

12.1 The Automation Interface

Introduction

The Automation Interface provides a way of accessing the internals of EA models. Here are some examples of things you might want to do using the Automation Interface:

- Perform repetitive tasks, such as update the version number for all elements in a model.
- Generate code from a state machine diagram.
- Produce custom reports.
- Ad hoc queries.

and much much more.

Connecting to the Automation Interface

All development environments capable of generating ActiveX Com clients should be able to connect to the Enterprise Architect Automation Interface. This guide provides detailed instructions on [connecting to the interface](#) using Microsoft Visual Basic 6.0, Borland Delphi 7.0 and Microsoft C#. There is also more detailed instruction on how to [set-up Visual Basic](#); the principles should be applicable to other languages.

Examples and Tips

Instruction on how to use the Automation Interface is provided by means of sample code. See [pointers to the samples](#) and other [available resources](#). Also, you will probably want to consult the extensive [Reference Section](#).

Calling Executables from EA

EA can be set up to call an external application. You can pass parameters on the current position selected in the Project View to the application being called. For instructions, go to [Calling Executables from EA](#). A more sophisticated method is to create [Add-Ins](#), which are discussed in a separate section.

12.1.1 Using the Automation Interface

This chapter provides instructions on how to connect to and use the interface.

See:

- [Connecting to the Interface](#)
- [Set Up VB](#)
- [Examples and Tips](#)

12.1.1.1 Connecting to the Interface

All development environments capable of generating ActiveX Com clients should be able to connect to the Enterprise Architect Automation Interface.

By way of example, the following text describes how to connect using several such tools. The procedure may vary slightly with different versions of these products.

Microsoft Visual Basic 6.0

1. Create a new project.
2. Click on the *Project* menu and select *References*.
3. Select *Enterprise Architect Object Model 2.0* from the list. (If this does not appear, go to the command line and re-register EA using ea.exe /unregister then ea.exe /register).
4. Refer to the general library documentation on the use of classes. The following example creates and opens a repository object:

```
Public Sub ShowRepository()  
    Dim MyRep As New EA.Repository  
    MyRep.OpenFile "c:\eatest.eap"  
End Sub
```

Borland Delphi 7.0

1. Create a new project.
2. Click on the *Project* menu and select *Import Type Library*.
3. Select *Enterprise Architect Object Model 2.0* from the list. (If this does not appear, go to the command line and re-register EA using ea.exe /unregister then ea.exe /register).
4. Click the button *Create Unit*.
5. Include EA_TLB in Project1's Uses clause.
6. Refer to the general library documentation on the use of classes. The following example creates and opens a repository object:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    r: TRepository;  
    b: boolean;  
begin  
    r := TRepository.Create(nil);  
    b := r.OpenFile('c:\eatest.eap');  
end;
```

Microsoft C#

1. Select *Enterprise Architect Object Model 2.0* from the list. (If this does not appear, go to the command line and re-register EA using `ea.exe /unregister` then `ea.exe /register`).
2. Create a new *Windows Application* project.
3. From the Solution Explorer, right-click on *References* and select *Add References* from the menu.
4. Click the *COM* tab.
5. Click the button *Select* then *OK*.
6. Refer to the general library documentation on the use of classes. The following example creates and opens a repository object:

```
private void button1_Click(object sender, System.EventArgs e)
{
    EA.Repository r = new EA.RepositoryClass();
    r.OpenFile("c:\\eatest.eap");
}
```

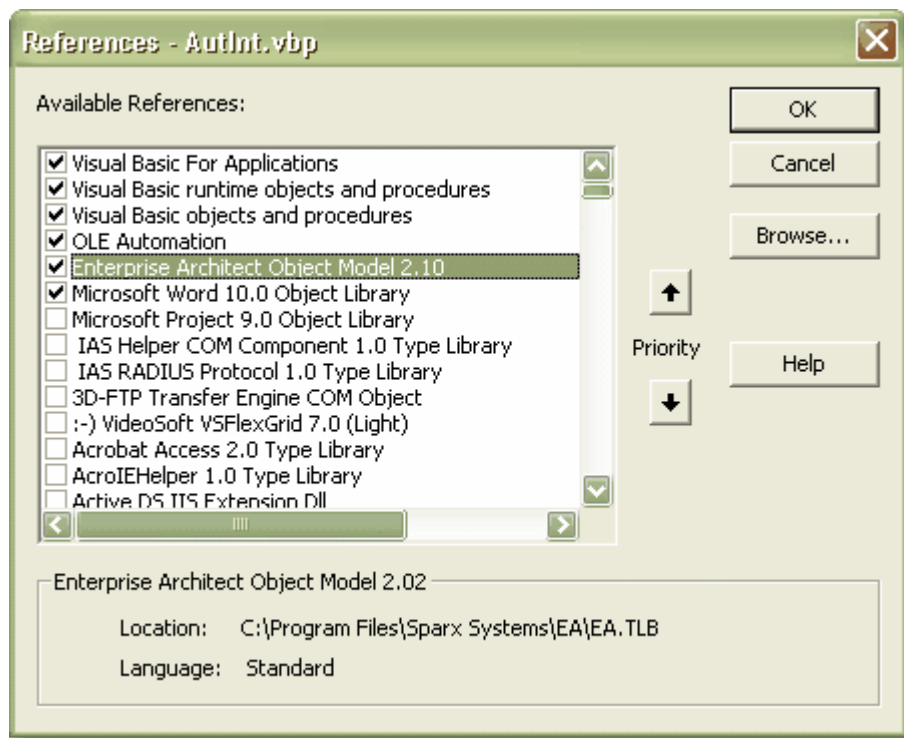
12.1.1.1.1 Set Up VB

Setting References in Visual Basic

The following describes how to use the Enterprise Architect ActiveX interface with Visual Basic. Usage is ensured for version 6. This may vary slightly on versions other than these. It is assumed that VB has been accessed through a Microsoft Application such as VB 6.0, MS Word or MS Access. If the code is not called from within Word, the Word VB reference must also be set.

On creating a new VB project, set a reference to an **Enterprise Architect Type Library** and a **Word Type Library** as follows:

1. Select References on the *Tools* menu.



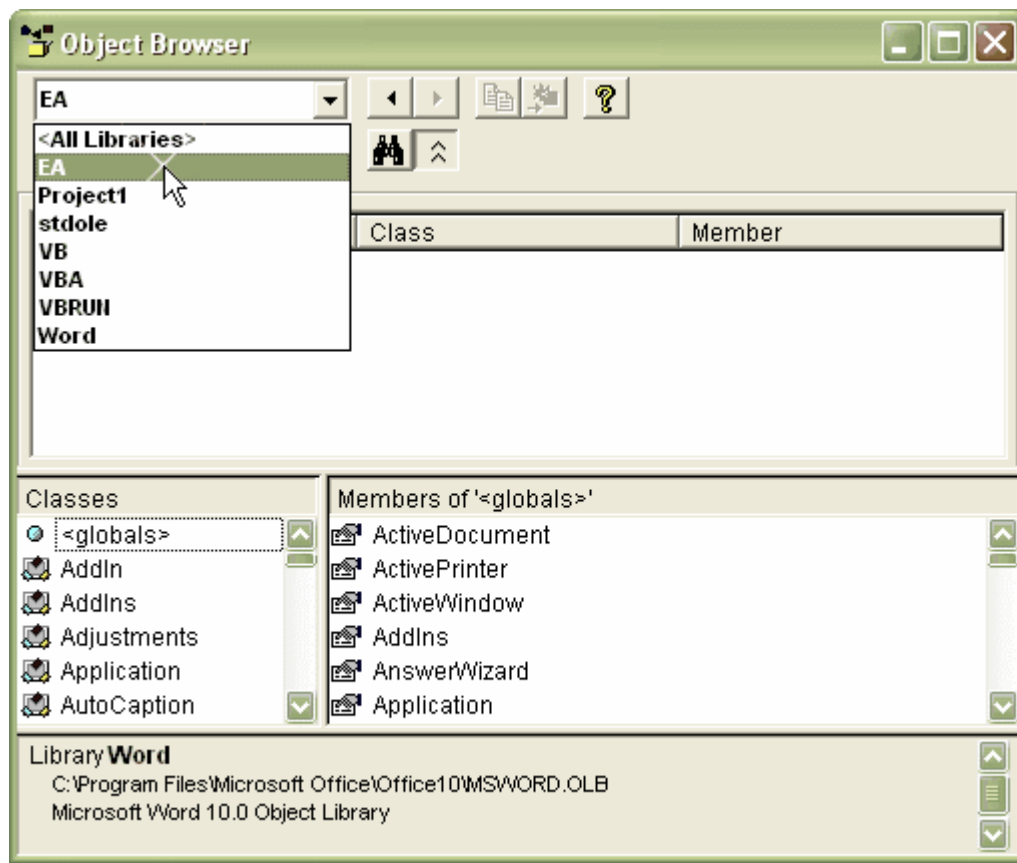
2. Select the check box for *Enterprise Architect Object Model 2.1* from the list.
3. Do the same for **VB** or **VB Word** - select the check box for the *Microsoft Word 10.0 Object Library*.
4. Press **OK**.

Note: If *Enterprise Architect Object Model 2.0* does not appear in the list, go to the command line and manually re-enter EA using the following:

- To unregister EA: **ea.exe /unregister**
- To register EA: **ea.exe /register**

Visual Basic 5/6 users should also note that the version number of the EA interface is stored in the VBP project file in a form similar to the following:
 Reference="G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\..\Program Files\Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02

If you experience problems moving from one version of EA to another, open the VBP file in a text editor and remove this line. Then open the project in Visual Basic and use Project-References to create a new reference to the EA Object model.



Reference to objects in EA and Word should now be available in the Object Browser. This can be accessed from the main menu by selecting *View* then *Object Browser*, or by pressing **F2**.

The drop down list on the top-left of the window should now include EA and Word (if MS-Project is installed this will also need to be set up).

12.1.1.2 Examples and Tips

Instructions for using the interface are provided through sample code. There are several sets of examples:

- VB 6 and C# samples are available in the Samples folder of your EA installation – normally "C:\Program Files\Sparx Systems\EA\Code Samples".
- A series of VB.NET examples is provided in the [reference section](#).
- A comprehensive example of using Visual Basic to create MS Word documentation is available from the internet at www.sparxsystems.com.au/AutIntVB.htm.

Additionally, the following list of tricks and traps should be noted:

- The EA application will appear when you access the interface, you will not be able to close it but you can minimize it if it gets in the way.
- The EA ActiveX Interface is a functional interface rather than a data interface. When you load data through the interface you will encounter a noticeable delay as EA user interface elements (eg. Windows, menus) are loaded along with the data.
- Collections are zero-based. Repository.Models(0) represents the first model in the repository.
- You can create multiple Repository objects - but don't do it. They will manipulate the same data. It is not currently possible to open two .EAP files at once.
- Sometimes during the development of your client software your program may terminate unexpectedly. It

may be that EA.exe is left running in such a state that it is unable to support further interface calls. If you encounter inexplicable problems ensure that EA is not running (see Windows Task Manager / Process Tab).

EA Not Closing

If your automation controller was written using the .NET framework, EA will not close even after you release all your references to it. To force the release of the COM pointers, call the memory management functions as shown below:

```
GC.Collect();  
GC.WaitForPendingFinalizers();
```

There are additional concerns when controlling a running instance of EA which loads Add-ins - see the [Tricks and Traps](#) topic for details.

See also:

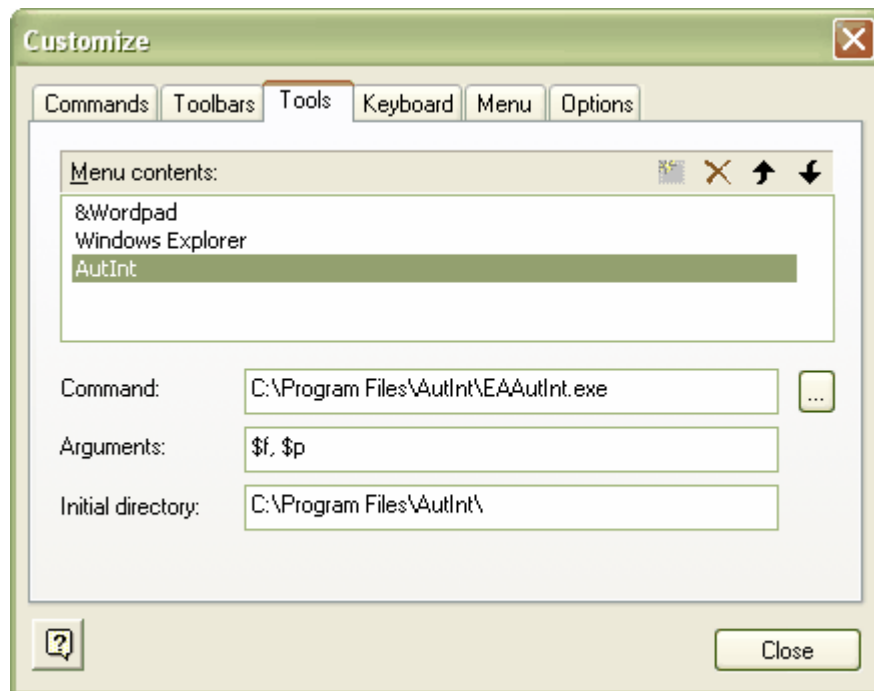
- [Call from EA](#)
- [Available Resources](#)

12.1.1.2.1 Call from EA

Automation Interface Examples: Calling Applications from EA

EA can be set up to call an external application. You can pass parameters on the current position selected in the Project View to the application being called.

To define an application runnable from EA - select from the main menu the **Tools | Customize | Tools** tab option.



With this you can:

- Add a command line for an application
- Define parameters to pass to this application

The parameters required for running the AutInt executable are:

1. The EA file parameter \$f and
2. The current PackageId \$p

Hence the arguments should simply contain: \$f, \$p

The available parameters for passing information to external applications are:

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

Once this has been set up, the application can be called from the main menu in EA using Tools | YourApplication.

12.1.1.2.2 Available Resources

Other available resources include:

Resource	Download Link
Examples - Some PDF copies of documents produced by this application	www.sparxsystems.com.au/AutIntVBExamples.htm
The Application - A brief on the application	www.sparxsystems.com.au/AutIntVBApplication.htm
Details - Background on the code	www.sparxsystems.com.au/AutIntVBDetail.htm
Download - Download the code and the executable	www.sparxsystems.com.au/AutIntVBDownload.htm
References - Guidelines on the use of the AI	www.sparxsystems.com.au/AutIntVBReferences.htm

12.1.2 Reference

This chapter provides detailed information on all the objects available in the object model provided by the Automation Interface.

See:

- [Interface Overview](#)
- [App](#)

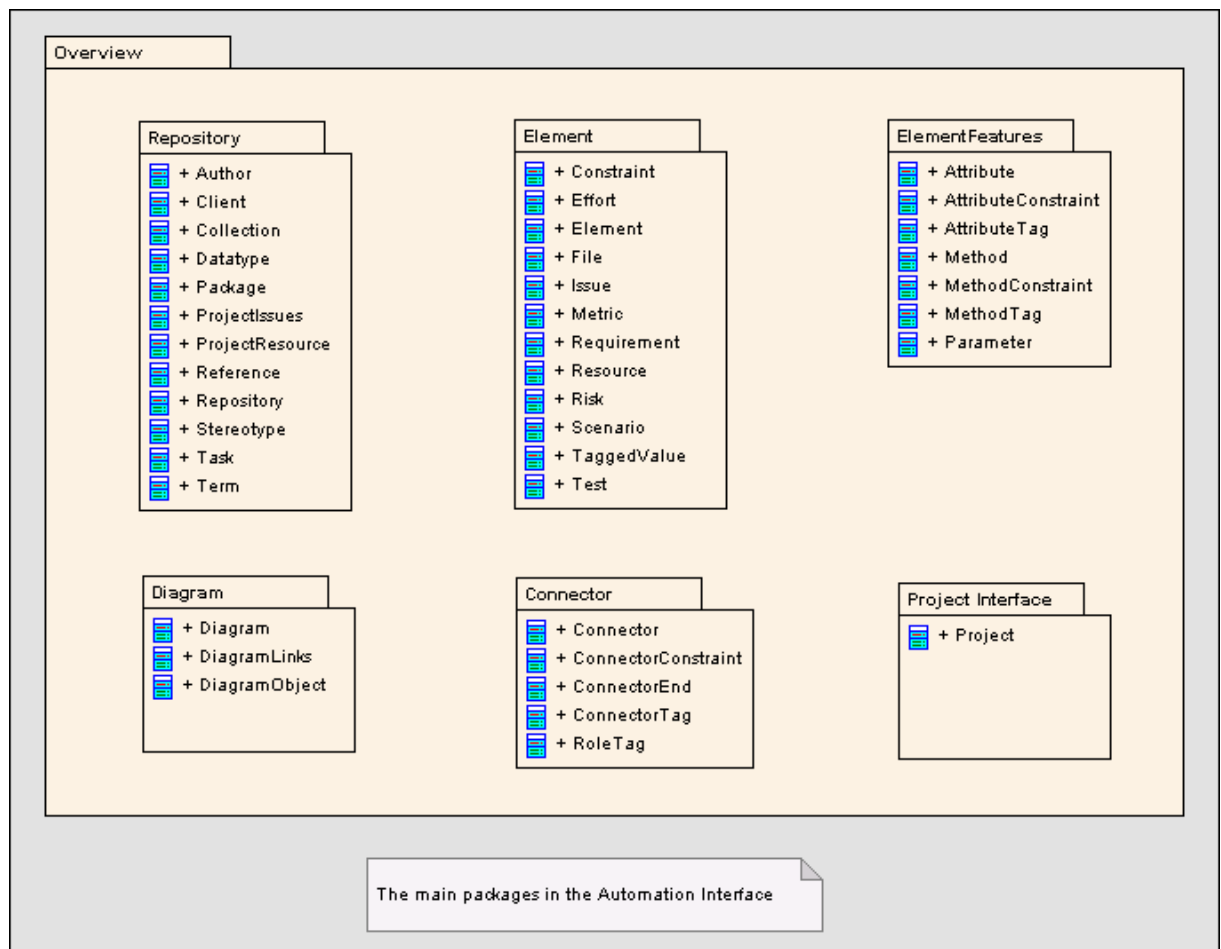
- [Enumerations](#)
- [Repository](#)
- [Element](#)
- [Element Features](#)
- [Connector](#)
- [Diagram](#)
- [Project Interface](#)
- [Code Samples](#)

12.1.2.1 Interface Overview

public Package

This diagram illustrates the main interface elements and their associated contents. Each element in this document is creatable by Automation and may be accessed directly in some cases - or through the various collections that exist.

Figure 1 : Automation Interface



Overview

public Package

This package provides an overview of the main elements within the Automation Interface. These are:

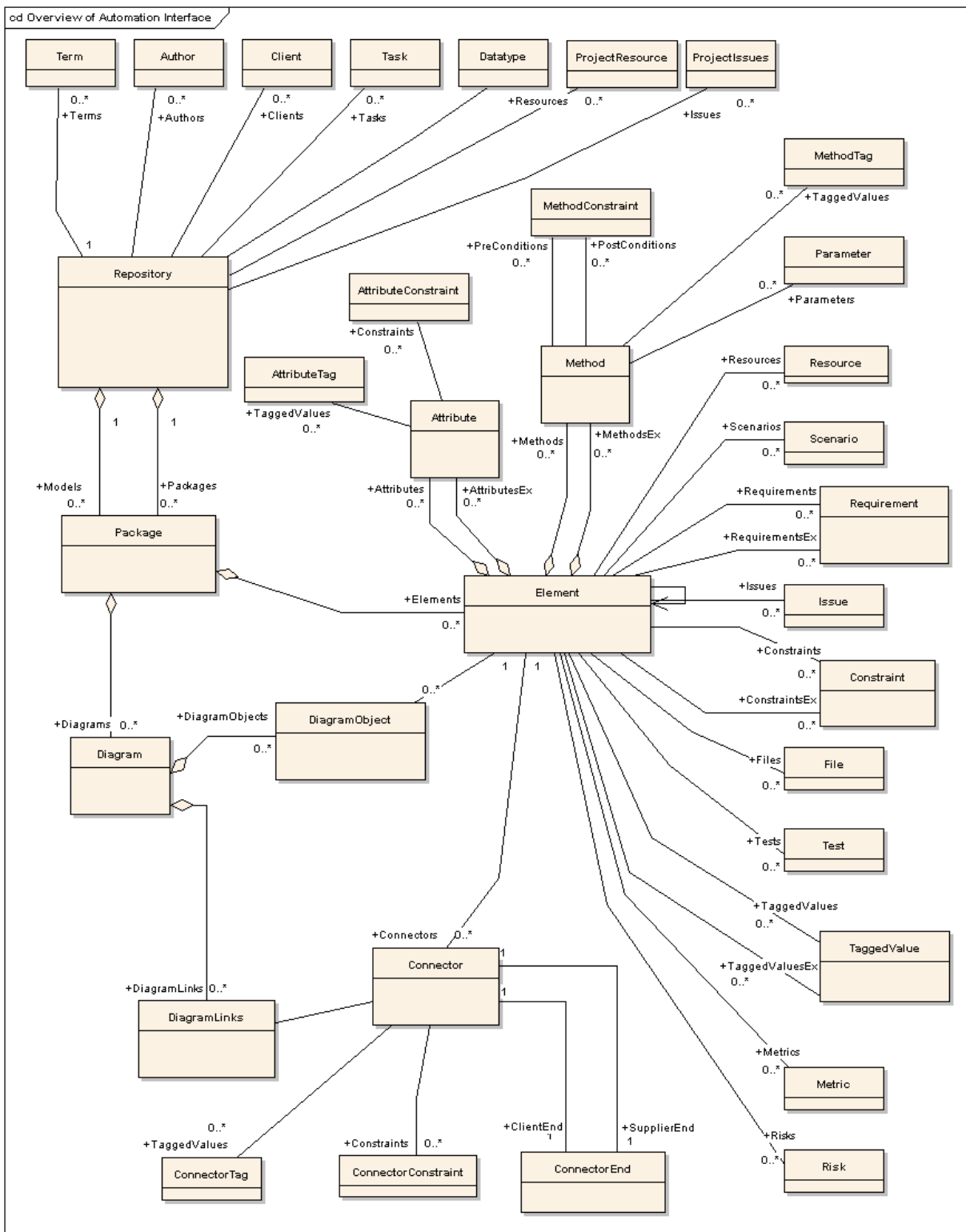
- The [Repository](#) which represents the model as a whole and provides entry to model packages and

collections;

- [Elements](#) which are the basic structural unit (eg. Class, Use Case and Object)
- [Element Features](#) which are attributes and operations defined on an element
- [Diagram](#) - the visible drawings contained in the model
- [Connectors](#) - relationships between elements

This diagram provides a high level overview of the Automation Interface for accessing, manipulating, modifying and creating EA UML elements. The top level object is the Repository - which contains collections for a variety of system level objects, as well as the main Models collection which provides access to the UML elements, diagrams, packages etc. within the project. In general the Role names applied at the Target end of associations indicate the name of the Collection which is used to access instances of that object.

Figure 2 : Overview of Automation Interface



Internal Links

- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface
- Logical diagram :: Automation Interface
Package :: Automation Interface

Connections

Connector	Source	Target
<i>Nesting</i> source > target	<i>Connector</i> Contained Element	<i>Overview</i> Containing Element
<i>Nesting</i> source > target	<i>Repository</i> Contained Element	<i>Overview</i> Containing Element
<i>Nesting</i> source > target	<i>Diagram</i> Contained Element	<i>Overview</i> Containing Element
<i>Nesting</i> source > target	<i>Element</i> Contained Element	<i>Overview</i> Containing Element
<i>Nesting</i> source > target	<i>Project Interface</i> Contained Element	<i>Overview</i> Containing Element
<i>Nesting</i> source > target	<i>ElementFeatures</i> Contained Element	<i>Overview</i> Containing Element

12.1.2.2 App

The App object represents a running instance of EA. Its object provides access to the Automation Interface.

Attribute	Type	Notes
Repository	<i>Repository</i>	Read Only. Provides a handle to the Repository object.
Project	<i>Project</i>	Read Only. Provides a handle to the Project Interface.
Visible	<i>Boolean</i>	Read/Write. Whether or not the application is visible.

GetObject() Support

The App object is creatable and a handle may be obtained by creating one.

In addition, clients may use the equivalent of VB's GetObject() to obtain a reference to a currently running instance of EA.

Use this method to more quickly test changes to add-ins and external clients, since the EA application and data files do not need to be constantly re-loaded.

For example:

```
Dim App as EA.App
Set App = GetObject(, "EA.App")
MsgBox App.Repository.Models.Count
```

12.1.2.3 Enumerations

These enumerations are defined by the Automation Interface:

- [MDGMenus Enum](#)
- [EnumRelationSetType Enum](#)
- [ConstLayoutStyles Enum](#)
- [ObjectType Enum](#)
- [XMIType Enum](#)

12.1.2.3.1 MDGMenus Enum

Use this enumeration when providing HiddenMenus property to [MDG_GetProperty](#).

These options are exclusive of one another and may be read or added to hide more than one menu.

See [MDG_GetProperty](#) for an example of use.

mgMerge	Hide "Merge" menu option.
mgBuildProject	Hide "Build Project" menu option.
mgRun	Hide "Run" menu option.

12.1.2.3.2 EnumRelationSetType Enum

This enumeration represents values returned from the GetRelationSet method of the [Element](#) object.

Method	Notes
rsGeneralizeStart	List of elements which the current element generalizes.
rsGeneralizeEnd	List of elements which are generalized by the current element
rsRealizeStart	List of elements which the current element realises.
rsRealizeEnd	List of elements which are realised by the current element.
rsDependStart	List of elements which the current element depends on.
rsDependEnd	List of elements which depend on the current element.
rsParents	List of all parent elements of the current element.

12.1.2.3.3 ConstLayoutStyles Enum

The enum values defined here are used exclusively for the [Layout a Diagram](#) method. They allow for the ability to define the layout options as depicted in the Layout a Diagram menu command (refer to [Layout a Diagram](#) for further information):

Method	Notes
IsDiagramDefault	
IsProgramDefault	
IsCycleRemoveGreedy	Use the Greedy Cycle Removal algorithm
IsCycleRemoveDFS	Use the Depth First Cycle Removal algorithm
IsLayeringLongestPathSink	Layer the diagram using the Longest Path Sink algorithm
IsLayeringLongestPathSource	Layer the diagram using the Longest Path Source algorithm
IsLayeringOptimalLinkLength	Layer the diagram using the Optimal Link Length algorithm
IsInitializeNaive	Initialize the layout using the Naïve Initialize Indices algorithm
IsInitializeDFSOut	Initialize the layout using the Depth First Search Outward algorithm
IsInitializeDFSIn	Initialize the layout using the Depth First Search Inward algorithm
IsCrossReduceAggressive	Perform aggressive Cross-reduction in the layout process (time consuming)
IsLayoutDirectionUp	Direct links to point upwards
IsLayoutDirectionDown	Direct links to point downwards
IsLayoutDirectionLeft	Direct links to point leftwards
IsLayoutDirectionRight	Direct links to point rightwards

12.1.2.3.4 Object Type Enum

The Object Type enumeration identifies EA object types even when referenced through a dispatch interface.

For example:

```
Object ob = Repository.GetElementByID(13);
if ( ob.ObjectType == otElement )
;
else if( ob.ObjectType == otAuthor )
...

```

12.1.2.3.5 XMIL Type Enum

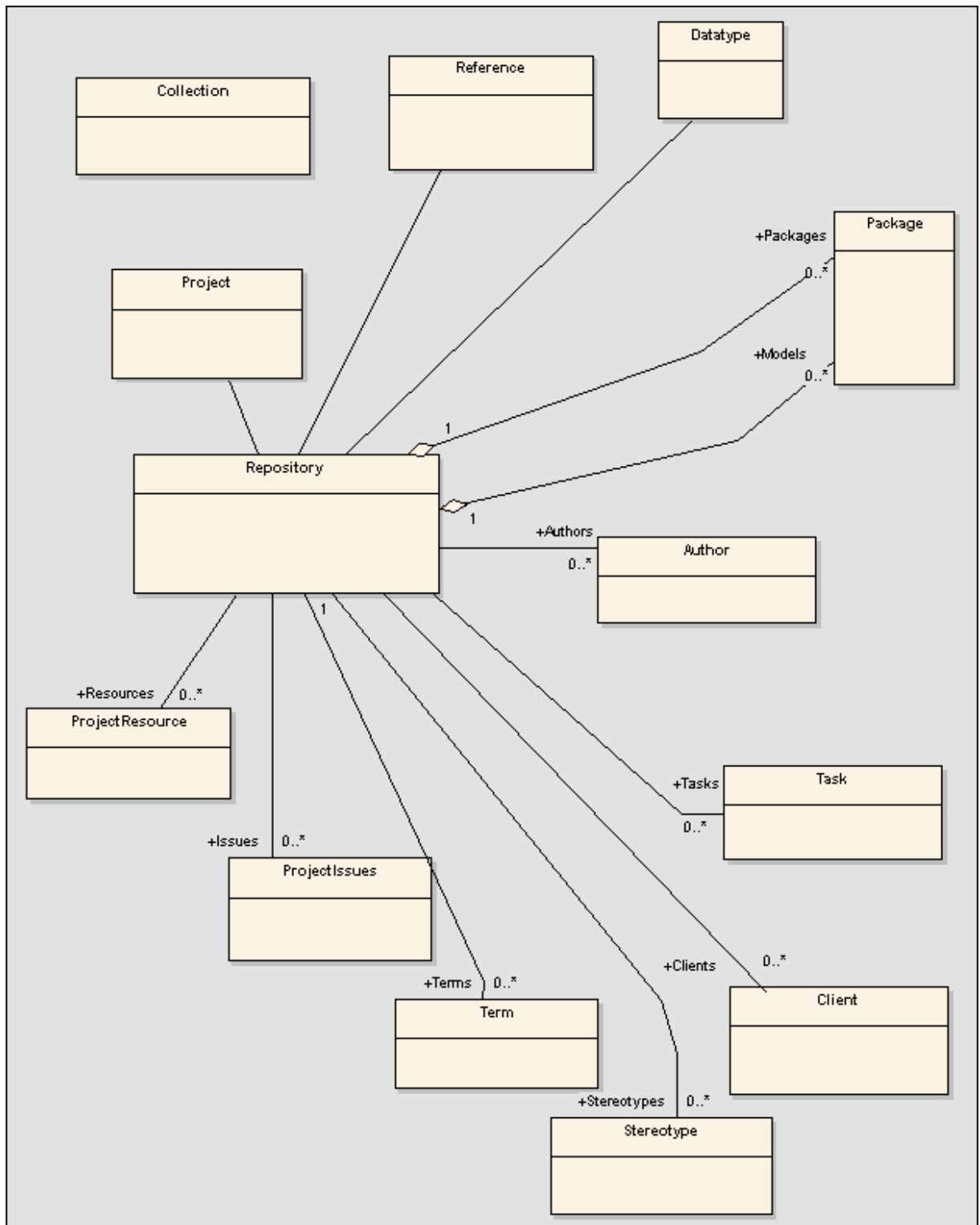
The following enumeration values are used Project.ExportPackageXMI() method. They allow for the specification of the XMI export type :

- xmiEADefault
- xmiRoseDefault
- xmiEA10
- xmiEA11
- xmiEA12
- xmiRose10
- xmiRose11
- xmiRose12

12.1.2.4 Repository

This diagram illustrates the [Repository](#) and its first level functions and collections. From here, the rest of the model may be accessed - using the Models collection and the other system level collections.

Figure 3 : Repository



Repository

public Package

The Repository package contains the high level system objects and entry point into the Model itself using

the Model's collection.

12.1.2.4.1 Repository

Repository

public Class

The Repository is the main container of all models, packages, elements, etc. You can iteratively begin accessing the model using the Models collection. Also has some convenience methods to directly access elements, packages, etc. without having to locate them in hierarchy first.

Associated table in .EAP file: <none>

Repository Attributes

Attribute	Type	Notes
Models	<i>Collection of type Package</i>	Read only. Models of type package and belong to a collection of packages. This is the top level entry point to an EA project file. Each model is a Root Node in the Project Browser and may contain views/packages etc. A Model is a special form of a Package - it has a ParentID of '0'. By iterating through all models, you can access all the elements within the project hierarchy. You may also use the AddNew function to create a new Model. A Model may also be deleted, but remember that everything contained in the Model will be deleted as well!
Terms	<i>Collection</i>	Read only. The project Glossary. Each Term object is an entry in the glossary. Add, modify and delete Terms to maintain the Glossary.
Issues	<i>Collection</i>	Read only. The System Issues list. Contains Issue objects - each detailing a particular issue as it relates to the project as a whole.
Authors	<i>Collection</i>	Read only. The system Authors collection. Contains 0 or more Author objects - each of which can be associated with elements, diagrams etc. as the item author or owner. Use AddNew, Delete and GetAt to manage Authors
Clients	<i>Collection</i>	Read only. A list of Clients associated with the Project. You can add new, modify and delete Client objects using this collection.
Tasks	<i>Collection</i>	Read only. A list of system tasks (to do list). Each entry is a Task Item - you may add new, modify and delete Tasks.
Datatypes	<i>Collection</i>	Read only. The Datatypes collection. Contains a list of Datatype objects - each representing a data type definition for either data modeling or code generation purposes.
Resources	<i>Collection</i>	Read only. A list of available resources to assign to work items within the project. Use the add new, modify and delete functions to manage Resources.
Stereotypes	<i>Collection</i>	Read only. The stereotypes collection. A list of Stereotype objects which contain information in a stereotype and what elements it may be applied to.
PropertyTypes	<i>Collection</i>	Read only. Collection of Property Types available to the Repository.
LibraryVersion	<i>Long</i>	Read only. The build number of the EA runtime.
LastUpdate	<i>String</i>	Read only. The identifier string identifying the EA runtime session and the timestamp it was set.
FlagUpdate	<i>Boolean</i>	Read/Write. Instructs EA to update the Repository with the LastUpdate value.
InstanceGUID	<i>String</i>	Read only: The identifier string identifying the EA runtime session.
ConnectionString	<i>String</i>	Read only. The filename/connection string of the current Repository
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface

Repository Methods

Method	Type	Notes
OpenFile (String)	<i>Boolean</i>	param: Filename [String - in] The filename of the EA project to open. This is the main point for opening an EA project file from an automation client. Provide the filename of a valid EA project and call this to open it and work with the contained objects.
GetElementByID (Long)	<i>Element</i>	param: ElementID [Long - in] Get a pointer to an Element using an absolute reference number (local ID). This is usually found using the ElementID property of an element - and stored for later use when you wish to open an Element without using the collection GetAt() function.
GetPackageByID (Long)	<i>Package</i>	param: PackageID [Long - in] Get a pointer to a Package using an absolute reference number (local ID). This is usually found using the PackageID property of a package- and stored for later use when you wish to open a Package without using the collection GetAt() function.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
GetDiagramByID (Long)	<i>Diagram</i>	param: DiagramID [Long - in] Get a pointer to a Diagram using an absolute reference number (local ID). This is usually found using the DiagramID property of an element - and stored for later use when you wish to open a Diagram without using the collection GetAt() function.
GetReferenceList (String)	<i>Reference</i>	param: Type [String - in] The list type to get Get a pointer to Reference List object. The parameter specifies the list type to get. Valid lists are: Diagram, Element, Constraint, Requirement, Connector, Status, Cardinality, Effort, Metric, Scenario, Status, Test.
GetProjectInterface ()	<i>Project</i>	Return a pointer to the EA.Project interface - the XML based automation server for EA. Use this interface to work with EA using XML, and also to access utility functions for loading diagrams, running reports etc.

Exit		Shut down EA immediately. Used by DotNET programmers where the garbage collector does not immediately release all referenced COM objects.
OpenFile2 (string FilePath, string Username, string Password)	<i>Boolean</i>	As for OpenFile() except allows the specification of a password.
ShowWindow(long Show)		Show or hides EA.
GetCurrentDiagram()	<i>Diagram</i>	Returns selected diagram.
GetConnectorByID (long ConnectorID)	<i>Connector</i>	Searches the repository for a connector with matching ID.
GetTreeSelectedItem (Object SelectedItem)	<i>ObjectType</i>	Used to get an object variable and type corresponding to the currently selected item in the treeview. To use this function, create a generic object variable and pass this as the parameter. Depending on the return type, cast it to a more specific type. This object passed back through the parameter can be a package, element, diagram, attribute or operation object.
GetTreeSelectedPackage ()	<i>Package</i>	Returns the package in which the currently selected treeview object is contained.
CloseAddins()		Called by automation controllers to ensure that Add-ins created in .NET will not linger after all controller references to EA have been cleared.
AdviseObjectChange()	<i>Boolean</i>	param: ObjectID [long - in] Provides an Add-In or automation client with the ability to advise the EA user interface that a particular element has changed and if it is visible in any open diagram, to reload and refresh that element for the user.
AdviseConnectorChange ()	<i>Boolean</i>	param: ConnectorID [long - in] Provides an Add-In or automation client with the ability to advise the EA user interface that a particular connector has changed and if it is visible in any open diagram, to reload and refresh that connector for the user.
OpenDiagram()	<i>Boolean</i>	param: DiagramID [long - in] Provides a method for an automation client or Add-In to open a diagram by its ID. The diagram will be added to the tabbed list of open diagrams in the main EA view.
CloseDiagram()	<i>Boolean</i>	param: DiagramID [long - in] Provides a method to close a diagram (if open) by its ID in the current list of diagrams EA has open.
ActivateDiagram()	<i>Boolean</i>	param: DiagramID [long - in] Provides the means to activate an already open diagram (ie. make it be the active tab) in the main EA user interface.

SaveDiagram()	<i>Boolean</i>	param: DiagramID [long - in] Call this to save an open diagram by its ID number. Assumes the diagram will be open in the main user interface Tab list.
ReloadDiagram()	<i>Boolean</i>	param: DiagramID [long - in] Reloads the diagram specified by ID value. This would commonly be used to refresh a visible diagram after code import/export or other batch process where the diagram requires complete refreshing.
CloseFile()		Closes any open file
GetTreeSelectedItemType()	<i><u>ObjectType</u></i>	param: none Returns the type of object currently selected in the tree. One of: <ul style="list-style-type: none"> • otDiagram • otElement • otPackage • otAttribute • otMethod
GetTechnologyVersion()	<i>String</i>	param: ID [string] Returns the version of the MDG Technology resource with the specified technology ID.
IsTechnologyLoaded()	<i>Boolean</i>	param: ID [string] Returns True, if the MDG Technology resource with the specified technology ID is loaded into the repository. Returns False, otherwise.
ImportTechnology()	<i>Boolean</i>	param: Technology [string] Installs the given MDG Technology resource into the repository. The Technology parameter represents the contents of the Technology resource file. Returns True, if the technology is successfully loaded into the model. Returns False, otherwise.
GetCounts()		Returns a set of counts from a number of tables within the base EA repository. These can be used to determine whether records have been added or deleted from the tables for which information is retrieved.

GetElementSet()	<i>Collection</i>	Returns a set of elements as a collection based on an input of element ID numbers in comma separated form. eg. GetElementSet("34,56,21,5")
DeleteTechnology()	<i>Boolean</i>	param: ID [string] Removes the MDG Technology resource with the specified technology ID from the repository. Returns True, if the technology is successfully removed from the model. Returns False otherwise.
AddTab()	<i>activeX custom control</i>	param: TabName [String - in] param: ControlID [String - in] Allows an ActiveX custom control to be added as a tabbed window. TabName is used as the tab caption. ControlID is the ProgID of the control. eg. Project1.UserControl1 EA creates a control and if successful, returns its IUnknown pointer which may be used by the caller to manipulate the control.
CreateOutputTab		param: Name [String in] Creates a tab in the Output View with the specified name.
RemoveOutputTab		param: Name [String in] Removes the tab in the Output View with the specified name.
WriteOutput		param: Name [String in] param: String [String in] param: ID [long in] Writes the text in String to the tab in the Output View with the specified name. It also associates the text in the Output View with the specified ID.
ClearOutput		param: Name [String in] Removes all the text in the tab in the Output View with the specified name.
EnsureOutputVisible		param: Name [String in] Ensures that the tab in the Output View with the specified name is visible to the user. The Output View will be made visible if it is hidden.

12.1.2.4.1.1 Author

Author

public Class

An Author object represents a named model author. Accessed using the Repository Authors collection.

Associated table in .EAP file: t_authors

Author Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. Author name
Roles	<i>String</i>	Read/Write. Roles the author may play in this project
Notes	<i>String</i>	Read/Write. Notes about the author
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface

Author Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Author object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.2 Client

Client

public Class

A Client represents one or more people or organizations related to the project. Accessed using the Repository Clients collection.

Associated table in .EAP file: t_clients

Client Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. Client name
Organization	<i>String</i>	Read/Write. Associated organization
Phone1	<i>String</i>	Read/Write. Main phone number
Phone2	<i>String</i>	Read/Write. Second phone number
Mobile	<i>String</i>	Read/Write. Mobile phone if available
Fax	<i>String</i>	Read/Write. Fax number
Email	<i>String</i>	Read/Write. Email address
Roles	<i>String</i>	Read/Write. Roles this client may play in project
Notes	<i>String</i>	Read/Write. Notes about client
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Client Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Client object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.3 Collection

Collection

public Class

This is the main collection class used by all elements within the automation interface. It contains methods to iterate through the collection, refresh the collection and delete an item from the collection. It is important to realize that when AddNew is called, the item is not automatically added to the current collection. The typical steps are:

1. Call AddNew to add a new item
2. Modify the item as required
3. Call Update on the item to save it to the database
4. Call Refresh on the collection to include it in the current set

Delete is much the same - until Refresh is called, the collection will still contain a reference to the deleted item - which should not be called.

For each - may be used to iterate through the collection for languages that support this type of construct.

Collection Attributes

Attribute	Type	Notes
Count	<i>short</i>	Read only. The number of objects referenced by this list.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Collection Methods

Method	Type	Notes
GetAt (short)	<i>Object</i>	param: index [short - in] Retrieves the array object using a numerical index. If the index is out of bounds, an error will occur.
Delete (short)	<i>void</i>	param: index [short - in] Delete the item at the selected reference
DeleteAt (Boolean, short)	<i>void</i>	param: refresh [Boolean - in] param: index [short - in] Delete the item at the selected index. Also provides an option to refresh the collection after the deletion. You should not refresh a collection while looping through it, as the count and item positions may alter. You should not access the item at the selected index once it is deleted or an exception will be thrown.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
GetByName (String)	<i>Object</i>	param: Name [String - in] Get an item in the current collection by Name. Only applies to the main objects - Element, Package etc.
Refresh ()	<i>void</i>	Refresh the collection by re-querying the model and reloading the collection. Should be called after adding a new item or after deleting an item.
AddNew (String, String)	<i>Object</i>	param: Type [String - in] param: Name [String - in] Add a new item to the current collection. Note that the interface is the same for all collections - you must provide a Name and Type argument. What these are used for depend on the actual collection member. Also note that you must call Update() on the returned object to complete the AddNew. If Update() is not called the object is left in an indeterminate state.

12.1.2.4.1.4 Datatype

Datatype

public Class

A Datatype is a named type that may be associated with attribute or method types. It typically is related to either code engineering or database modeling. Datatypes also indicate which language or database system they relate to. Accessed using the Repository Datatypes collection.

Associated table in .EAP file: t_datatypes

Datatype Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The datatype name (eg. "integer"). This will appear in the related drop down datatype lists where appropriate
Type	<i>String</i>	Read/Write. The type: may be "DDL" for database datatype or "Code" for language datatypes
Product	<i>String</i>	Read/Write. The datatype product - eg. Java, C++, Oracle.
Size	<i>Long</i>	Read/Write. The datatype size
MaxLen	<i>Long</i>	Read/Write. Maximum length (DDL only)
MaxPrec	<i>Long</i>	Read/Write. Maximum precision(DDL only)
MaxScale	<i>Long</i>	Read/Write. Maximum scale(DDL only)
DefaultLen	<i>Long</i>	Read/Write. Default length (DDL only)
DefaultPrec	<i>Long</i>	Read/Write. Default precision(DDL only)
DefaultScale	<i>Long</i>	Read/Write. Default scale(DDL only)
UserDefined	<i>Long</i>	Read/Write. Indicates if datatype is a user defined type or system generated. Datatypes distributed with EA are all system. Datatypes created in the Datatype dialog are marked 1 (true)
HasLength	<i>String</i>	Read/Write. Indicates datatype has a length component
GenericType	<i>String</i>	Read/Write. The associated generic type for this data type
DatatypeID	<i>Long</i>	Read/Write. Instance ID for this datatype within the current model. System maintained
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Datatype Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Datatype object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs

12.1.2.4.1.5 Package

Package

public Class

A Package object corresponds to a package element in the EA Project Browser. It is accessed either through the Repository Models collection (a Model is a special form of Package) or through the Package Packages collection. Note that a Package has an Element object as an attribute - this corresponds to an EA Package element in the t_object table and is used to associate additional information (such as

scenarios and constraints) with the logical package. To set additional information for a package, reference the Element object directly. Also note that if you add a Package to a Diagram, you should add an instance of the Element - not the Package itself - to the DiagramObjects collection for a Diagram.

Associated table in .EAP file: t_package

Package Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The Name of the package.
Packages	<i>Collection</i>	Read only. A collection of contained packages which may be walked through.
Elements	<i>Collection</i>	Read only. A collection of Elements which belong to this package.
Diagrams	<i>Collection</i>	Read only. A collection of Diagrams contained in this package.
Notes	<i>String</i>	Read/Write. Notes about this package.
IsNamespace	<i>Boolean</i>	Read/Write. True is package is a Namespace root. Use 0 and 1 to set False and True.
PackageID	<i>Long</i>	Read only. The local Package ID number. Valid only in this model file.
PackageGUID	<i>Variant</i>	Read only. The global Package ID. Valid across models.
ParentID	<i>Long</i>	Read/Write. The ID of the Package that is the Parent of this one. 0 indicates this package is a Model (ie. it has no parent).
Created	<i>Date</i>	Read/Write. Date package created.
Modified	<i>Date</i>	Read/Write. Date package last modified.
IsControlled	<i>Boolean</i>	Read/Write. Indicates if the package has been marked as 'Controlled'.
IsProtected	<i>Boolean</i>	Read/Write. Indicates if the package has been marked as Protected.
UseDTD	<i>Boolean</i>	Read/Write. Indicates if a DTD is to be used when exporting XML.
LogXML	<i>Boolean</i>	Read/Write. Indicates if XML export information is to be logged.
XMLPath	<i>String</i>	Read/Write. The path to the where XML will be saved when using controlled packages.
Version	<i>String</i>	Read/Write. The Version of the package.
LastLoadDate	<i>Date</i>	Read/Write. Date XML was last loaded for package.
LastSaveDate	<i>Date</i>	Read/Write. Date XML was last saved from package.

Flags	<i>String</i>	Read/Write. Extended information about package.
Owner	<i>String</i>	Read/Write. The package owner when using controlled packages.
CodePath	<i>String</i>	Read/Write. Path to where associated source code is found.
UMLVersion	<i>String</i>	Read/Write. UML Version for XMI export purposes.
TreePos	<i>Long</i>	Read/Write. The relative position in the tree compared to other packages (use to sort packages).
IsModel	<i>Boolean</i>	Read only. Indicates if package is a Model or Package.
Element	<i>Object</i>	Read only. The associated Element object. Use to set Element type information for a package ... including Stereotype, Complexity, Alias, Author, Constraints, Scenarios ... etc.
BatchSave	<i>long</i>	Read/Write. Boolean value to indicate whether package will be included in the batch XMI export list or not.
BatchLoad	<i>long</i>	Read/Write. Flag to indicate package will be batch loaded during batch import from controlled packages. Not yet implemented.
Connectors	<i>Collection</i>	Read only. Collection of connectors.
Alias	<i>string</i>	Read only. Alias.
IsVersionControlled	<i>boolean</i>	Read/Write. Whether or not this package is under version control.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Package Methods

GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the current Package object after modification or appending a new item. If false is returned, check the GetLastError function for more information. Note that a Package object also has an 'Element' component which must be taken into account. The Package object contains information about the hierarchy, contents etc. The Element attribute contains information about Stereotype, Constraints, Files & etc. - all the attributes of a typical Element.

12.1.2.4.1.6 ProjectIssues

ProjectIssues

public Class

A system level Issue. Indicates a problem or risk associated with the system as a whole. Accessed using the Repository Issues collection.

Associated table in .EAP file: t_issues

ProjectIssues Attributes

Attribute	Type	Notes
Category	String	Read/Write. The category this issue belongs to.
Name	String	Read/Write. Issue name (ie. the Issue itself).
Date	Date	Read/Write. Date created.
Owner	String	Read/Write. Owner of issue.
Status	String	Read/Write. Current issue status.
Notes	String	Read/Write. Associated description of issue.
Resolver	String	Read/Write. Person resolving issue.
DateResolved	Date	Read/Write. Date issue resolved.
Resolution	String	Read/Write. Description of resolution.
Priority	String	Read/Write. Issue priority ... generally should use Low, Medium or High.
Severity	String	Read/Write. Issue severity. Should be marked as Low, Medium or High.
IssueID	Long	Read only. The ID of this issue.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

ProjectIssues Methods

Method	Type	Notes
Update ()	Boolean	Update the current Issue object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	String	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.7 ProjectResource

ProjectResource

public Class

A Project Resource is a named person who is available to work on the current project in any capacity. Accessed using the Repository Resources collection.

Associated table in .EAP file: t_resources

ProjectResource Attributes

Attribute	Type	Notes
Name	<i>String</i>	Name of resource.
Organization	package: <i>String</i>	Organization resource associated with.
Phone1	<i>Variant</i>	Main phone.
Phone2	<i>Variant</i>	Alternate phone.
Mobile	<i>Variant</i>	Mobile number if available.
Fax	<i>String</i>	Fax number.
Email	<i>String</i>	Email address.
Roles	<i>String</i>	The roles this resource can play in the current project.
Notes	<i>String</i>	A description if appropriate.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

ProjectResource Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Resource object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.8 Reference

Reference

public Class

This Interface provides access to the various lookup tables within EA. Use the Repository GetReferenceList() method to get a handle to a list. Valid lists are:

- Diagram
- Element
- Constraint
- Requirement
- Connector
- Status
- Cardinality
- Effort
- Metric
- Scenario
- Status
- Test

Reference Attributes

Attribute	Type	Notes
Count	<i>Short</i>	Count of items in the list.
Type	<i>String</i>	The list type (eg. Diagram Types).
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Reference Methods

Method	Type	Notes
GetAt (Short)	<i>String</i>	param: index [Short - in] The index of the item to retrieve from the list. Get the item at the specified index.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Refresh ()	<i>short</i>	Refresh the current list and return the count of items.

12.1.2.4.1.9 Stereotype

Stereotype

public Class

The Stereotype element corresponds to a UML Stereotype, which is an extension mechanism for varying the behavior and type of a model element. Use the repository Stereotypes collection to add new elements and delete existing ones.

Associated table in .EAP file: t_stereotypes

Stereotype Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The Stereotype. Appears in the Stereotype drop list for elements that match the AppliesTo attribute.
AppliesTo	<i>String</i>	Read/Write. A reference to the Stereotype Base Class ... ie. which element it applies to.
Notes	<i>String</i>	Read/Write. Notes about the Stereotype.
MetafileLoadPath	<i>String</i>	Read/Write. Path to an associated metafile. The automation interface does not yet support loading metafiles - to do this you must use the Stereotype dialog in EA.
Style	<i>String</i>	Read/Write. Additional style specifier for stereotype.
StereotypeGUID	<i>String</i>	Read/Write. Unique identifier for stereotype, generally set and maintained by EA.
VisualType	<i>String</i>	Read/Write. Indicates an inbuilt visual style associated with a stereotype. Not currently implemented.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Stereotype Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Stereotype object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.10 Task

Task

public Class

A Task is an entry in the System ToDo list. Accessed using the Repository Tasks collection.

Associated table in .EAP file: t_tasks

Task Attributes

Attribute	Type	Notes
TaskID	Long	Read only. Local ID of task.
Name	Variant	Read/Write. Task name.
Notes	Variant	Read/Write. Description of the task.
Priority	String	Read/Write. Priority associated with this task.
Status	Variant	Read/Write. Current task status.
Owner	String	Read/Write. The task owner.
StartDate	Date	Read/Write. Date task is to start.
EndDate	Date	Read/Write. Date task scheduled to finish.
Phase	String	Read/Write. The phase of the project the task relates to.
History	String	Read/Write. Memo field to hold task history, notes etc.
Percent	Long	Read/Write. Percent the task is complete.
TotalTime	Long	Read/Write. The total expected time the task will run - may be in hours or days or some other unit.
ActualTime	Long	Read/Write. Time already expended on task. May be hours or days or other units.
AssignedTo	String	Read/Write. Person this task is assigned to - ie. the responsible resource.
Type	String	Read/Write. Sets or returns string representing the type.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface

Task Methods

Method	Type	Notes
Update ()	Boolean	Update the current Task object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	String	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.11 Term

Term

public Class

A Term object represents one entry in the system glossary. Accessed using the Terms collection of the repository.

Associated table in .EAP file: t_glossary

Term Attributes

Attribute	Type	Notes
Term	<i>String</i>	Read/Write. The glossary item name.
Type	<i>String</i>	Read/Write. The type this term applies to (eg. business or technical).
Meaning	<i>String</i>	Read/Write. The description of the term - its meaning.
TermID	<i>Long</i>	Read only. A local ID number to identify the term in the model.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Term Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Term object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.4.1.12 EventProperty

EventProperty

EventProperty objects are always part of an [EventProperties](#) collection, and are passed to add-in methods responding to [broadcast events](#).

EventProperty Attributes

Attribute	Type	Notes
Name	<i>String</i>	A string distinguishing this property from others in the list.
Value	<i>Variant</i>	A string, number or object reference representing the property value.
Description	<i>String</i>	Explanation of what this property represents.
ObjectType	ObjectType	Distinguishes objects referenced through Dispatch interface.

See Also:

[Event Properties](#)

12.1.2.4.1.13 EventProperties

EventProperties

An EventProperties object is passed to BroadcastFunctions to facilitate parameter passing.

EventProperties Attributes

Attribute	Type	Notes
Count	Long	Read only. Number of parameters being passed to this broadcast event.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

EventProperties Methods

Method	Type	Notes
Get	EventProperty	<p>Read only. Params: Index (Variant) Returns an EventProperty in the list, raising an error if Index is out of range.</p> <p>Index can either be a number representing a zero-based index into the array, or a string representing the name of the EventProperty.</p> <p>eg. Props.Get(3) or Props.Get("ObjectID")</p>

12.1.2.4.1.14 Property TypePropertyTypepublic Class

A PropertyType object represents a defined property that may be applied to UML elements as a tagged value. Accessed using the Repository PropertyTypes collection. Each PropertyType corresponds to one of the predefined tagged values defined for the model.

Associated table in .EAP file: t_propertytypes

Author Attributes

Attribute	Type	Notes
Tag	String	Read/Write. Name of the property (Tag Name)
Description	String	Read/Write. Short description for the property
Detail	String	Read/Write. Configuration information for the property
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface

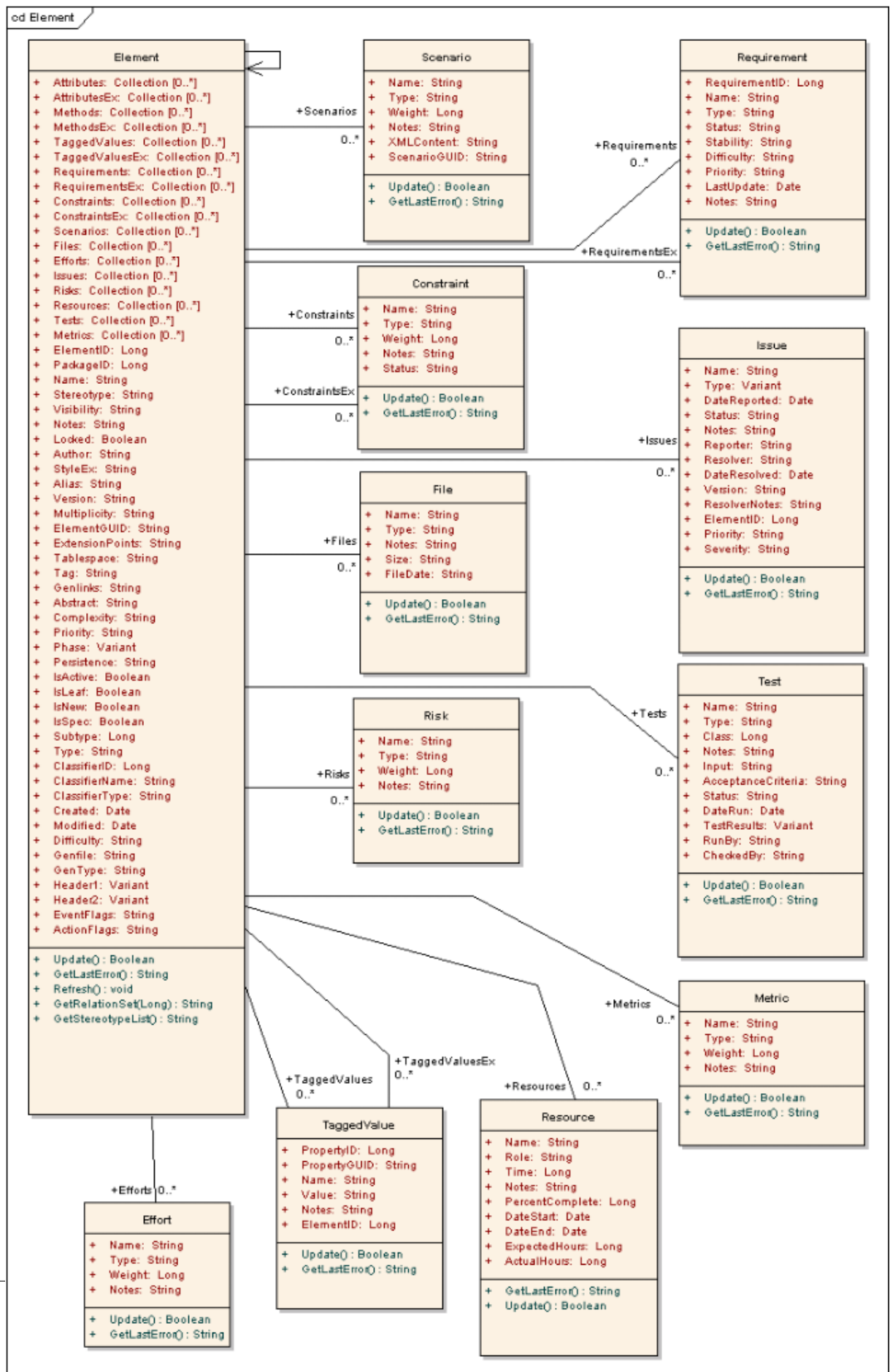
Author Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current PropertyType object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5 Element

This diagram illustrates the relationships between an Element and its associated extended information. The related information is accessed through the collections owned by the Element (eg. Scenarios and Tests). It also includes a full description of the Element object - the basic model structural unit.

Figure 4 : Element



Element**public Package**

The Element package contains information about an Element and its associated extended properties such as testing and project management information. An Element is the basic item in an EA model. Classes, Use Cases, Components & etc. are all different types of UML Elements.

12.1.2.5.1 Constraint**Constraint****public Class**

A Constraint is a condition imposed on an Element. Constraints are accessed through the Element Constraints collection

Associated table in .EAP file: t_objectconstraints

Constraint Attributes

Attribute	Type	Notes
Name	String	Read/Write. The name of the constraint (i.e. the constraint).
Type	String	Read/Write. Constraint type.
Weight	Long	Read/Write. A weighting factor.
Notes	String	Read/Write. Notes about the constraint.
Status	String	Read/Write. Current status.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.
ParentID	Long	Read only. The ElementID of the element to which this constraint applies.

Constraint Methods

Method	Type	Notes
Update ()	Boolean	Update the current Constraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	String	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.2 Effort**Effort****public Class**

Effort Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The name of the effort (i.e. the effort).
Type	<i>String</i>	Read/Write. Effort type.
Weight	<i>Long</i>	Read/Write. A weighting factor.
Notes	<i>String</i>	Read/Write. Notes about the constraint.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Effort Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Saves the Effort to the Model
GetLastError ()	<i>String</i>	This function is rarely used since an exception will be thrown when an error occurs. Returns a string value describing the most recent error that occurred in relation to this object.

12.1.2.5.3 Element**Element**public Class

An Element is the main modeling unit. It corresponds to Class, Use Case, Node, Component & etc. You create new Elements by adding to the Package Elements collection. Once you have created an element, you can add it to the DiagramObjects collection of a diagram to include it in a diagram.

Elements also have a collection of Connectors. Each entry in this collection indicates a relationship to another Element.

There are also some extended collections for managing additional information about the Element - including TaggedValues, Issues, Constraints, Requirement & etc.

Associated table in .EAP file: t_object

Element Attributes

Attribute	Type	Notes
Attributes	<i>Collection</i>	Read only. Collection of Attribute objects for current element. Use the AddNew and Delete function to manage attributes.
AttributesEx	<i>Collection</i>	Read only. Collection of Attribute objects belonging to the current element and its parent elements.
Methods	<i>Collection</i>	Read only. Collection of Method objects for current element.
MethodsEx	<i>Collection</i>	Read only. Collection of Method objects belonging to the current element and its parent elements.
TaggedValues	<i>Collection</i>	Read only. Collection of TaggedValue objects.
TaggedValuesEx	<i>Collection</i>	Read only. Collection of TaggedValue objects belonging to the current element and its parent elements.
Requirements	<i>Collection</i>	Read only. Collection of Requirement objects.
RequirementsEx	<i>Collection</i>	Read only. Collection of Requirement objects belonging to the current element and its parent elements.
Constraints	<i>Collection</i>	Read only. Collection of Constraint objects.
ConstraintsEx	<i>Collection</i>	Read only. Collection of Constraint objects belonging to the current element and its parent elements.
Scenarios	<i>Collection</i>	Read only. Collection of Scenario objects for current element
Files	<i>Collection</i>	Read only. Collection of File objects.
Efforts	<i>Collection</i>	Read only. Collection of Effort objects.
Issues	<i>Collection</i>	Read only. Collection of Issue objects.
Risks	<i>Collection</i>	Read only. Collection of Risk objects.
Resources	<i>Collection</i>	Read only. Collection of Resource objects for current element.
Tests	<i>Collection</i>	Read only. Collection of Test objects for current element.
Metrics	<i>Collection</i>	Read only. Collection of Metric elements for current element.
ElementID	<i>long</i>	Read only. The local ID of the Element. Valid for this file only.
PackageID	<i>long</i>	Read/Write. A local ID for the package containing this element.
Name	<i>String</i>	Read/Write. The element name - should be unique within the current package.
Stereotype	<i>String</i>	Read/Write. The element name - should be unique within the current package.
Visibility	<i>String</i>	Read/Write. The Scope of this element within the current package. Valid values are: Public, Private, Protected or Package.
Notes	<i>String</i>	Read/Write. Further descriptive text about Element.
Locked	<i>Boolean</i>	Read/Write. Indicates if Element has been locked against further change.
Author	<i>String</i>	Read/Write. The Element Author (see the Repository:Authors list for more details).
StyleEx	<i>String</i>	Read/Write: Advanced style settings. Not currently used.

Alias	<i>String</i>	Read/Write. An optional Alias for this element.
Version	<i>String</i>	Read/Write. The Version of the Element.
Multiplicity	<i>String</i>	Read/Write. Multiplicity value for this Element.
ElementGUID	<i>String</i>	Read only. A globally unique ID for this Element - unique across all model files. If you need to set this value manually, you should only do so when the element is first created - and make sure you format the GUID exactly as EA expects.
ExtensionPoints	<i>String</i>	Read/Write. Optional extension points for a Use Case as a comma-separated list.
Tablespace	<i>String</i>	Read/Write. Associated tablespace for a Table element.
Tag	<i>String</i> .	Read/Write. Optional Tag value for additional user defined information and searching.
Genlinks	<i>String</i>	Read/Write. Links to other classes discovered at code reversing time - Parents and Implements links only
Abstract	<i>String</i>	Read/Write. Indicates if Element is Abstract (1) or Concrete (0)
Complexity	<i>String</i>	Read/Write. A Complexity value indicating how difficult the Element is. May be used for metric reporting and estimation. Valid values are: 1 for Easy, 2 for Medium, 3 for Difficult
Priority	<i>String</i>	Read/Write. The priority of this element as compared to other project elements. Only applies to Requirement, Change and Issue types, otherwise ignored. Valid values are: "Low", "Medium" and "High"
Phase	<i>String</i>	Read/Write. Phase this element scheduled to be constructed in. Any string value
Persistence	<i>String</i>	Read/Write. The persistence associated with this element... may be "Persistent" or "Transient"
IsActive	<i>Boolean</i>	Read/Write. Boolean value indicating element is active or not. 1 = true, 0 = false
IsLeaf	<i>Boolean</i>	Read/Write. Boolean value indicating element is leaf node or not. 1 = true, 0 = false
IsNew	<i>Boolean</i>	Read/Write. Boolean value indicating element is new or not. 1 = true, 0 = false
IsSpec	<i>Boolean</i>	Read/Write. Boolean value indicating element is specification or not. 1 = true, 0 = false
Type	<i>String</i>	Read/Write. The Element type (eg. Class, Component etc) - Note that type is case sensitive inside EA and should be provided with an initial capital (proper case). Valid types are: Activity, Actor, Association, Boundary, Change, Class , Collaboration, Component, Decision, Entity, Event , GUIElement, Interface, Issue, Label, Node, Note , Object, Package, Requirement, Screen, Sequence, State, StateNode , Synchronization, Text, UseCase

Subtype	<i>Long</i>	Read/Write. A numeric subtype which varies the type of the main element: <ul style="list-style-type: none"> • For Event type 0 = Receiver, 1 = Sender • For Class type, 1 = Template class, 2 = Parameterised Template Class • For Note type, 1= Note linked to connector, 2 = Constraint linked to connector • For StateNode, 100 = ActivityInitial, 101 = ActivityFinal • For Activity type 0=Activity, 8 = SubActivity
ClassifierID	<i>Long</i>	Read/Write. Local ID of a Classifier associated with this Element - that is the base type. Only valid for instance type elements (eg. Object)
ClassifierName	<i>String</i>	Read/Write. Name of associated Classifier (if any)
ClassifierType	<i>String</i>	Read only. Type of associated classifier.
Created	<i>Date</i>	Read/Write. Date element created
Modified	<i>Date</i>	Read/Write. Date element last Modified
Difficulty	<i>String</i>	Read/Write. A difficulty level associated with this element for estimation/metrics - only useable for Requirement, Change and Issue element types, otherwise ignored. Valid values are: "Low", "Medium", "High"
Genfile	<i>String</i>	Read/Write. The file associated with this element for code generation and synchronization purposes. May include macro expansion tags for local conversion to full path.
GenType	<i>String</i>	Read/Write. The code generation type - eg. Java, C++, C#, VBNet, Visual Basic, Delphi.
Header1	<i>Variant</i>	Read/Write. A user defined string for inclusion as header in source files generated.
Header2	<i>Variant</i>	Read/Write. Same as for Header1 - but used in CPP source file
EventFlags	<i>String</i>	Read/Write. A structure to hold a variety of flags to do with signals, events etc.
ActionFlags	<i>String</i>	Read/Write. A structure to hold flags concerned with Action semantics.
Elements	<i>Collection</i>	The child elements of this element.
Diagrams	<i>Collection</i>	The child diagrams of this element.
ParentID	<i>Long</i>	Read/Write. Can be used to set or retrieve the If this element is a child of another, the ElementID of the other element. If not, returns 0.
Connectors	<i>Collection</i>	Read only. Returns a collection containing the connectors to other elements.
ClassifierID	<i>Long</i>	Read/Write. Sets or gets the ElementID of the Classifier.
Status	<i>String</i>	Read/Write. Sets or gets the status eg. "Proposed", "Approved" etc.

TreePos	<i>Long</i>	Read/Write. Sets or gets the tree position.
Elements	<i>Collection</i>	Read only. Returns a collection of sub-elements attached to this element as seen in the tree view.
Diagrams	<i>Collection</i>	Read only. Returns a collection of sub-diagrams attached this element as seen in the tree view.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.
Partitions	<i>Collection</i>	Read Only. List of logical partitions into which an element may be divided. Only valid for elements which support partitions, such as Activities and States.
CustomProperties	<i>Collection</i>	Read only. List of "Advanced Properties" for an element. The collection of advanced properties will change depending on element type – for example an Action and an Activity have different Advanced Properties. Currently only editable from the user interface.
StateTransitions	<i>Collection</i>	Read only. List of State Transitions that an element may support. Applies in particular to Timing Elements.
EmbeddedElements	<i>Collection</i>	Read only. List of elements which are embedded into this element. Includes Ports, Parts, Pins, Parameter Sets & etc.
BaseClasses	<i>Collection</i>	Read only. List of Base Classes for this element presented as a collection for convenience.
Realizes	<i>Collection</i>	Read only. List of Interfaces realized by this element for convenience.
MiscData	<i>String</i>	Read only. This low-level property provides information about the the contents of the PDATAx fields. These database fields are not documented and developers will need to gain understanding of these fields through their own endeavors to use this property. MiscData is zero based so MiscData(0) corresponds to PDATA1, MiscData(1) to PDATA2 etc.
StereotypeEx	<i>String</i>	Read/Write. Returns all the applied stereotypes of the element in a comma-separated list
PropertyType	<i>Long</i>	Read/Write. The GUID of the type which defines either a Port or a Part.

Element Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Element object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Refresh ()	<i>void</i>	Refresh the Element features in the Project Browser tree. Usually called after adding or deleting attributes or methods - when the user interface is required to be updated as well.
GetRelationSet (EnumRelationSetType Type)	<i>String</i>	Returns a string containing a comma-separated list of ElementIDs of related elements based on the given type. See EnumRelationSetType .
GetStereotypeList ()	<i>String</i>	Returns a comma-separated list of stereotypes allied to this element.
SetAppearance (long Scope, long Item, long Value)	<i>Long</i>	Sets the visual appearance of Element. Scope: Scope of appearance set to modify 0 – Local (Diagram-local appearance) 1 – Base (Default appearance across entire model) Item: Appearance item to modify 0 – Background color 1 – Font Color 2 – Border Color 3 – Border Width Value: Value to set appearance to.

12.1.2.5.4 File

File

public Class

A File represents an associated File for an Element. It is accessed through the Element Files collection

Associated table in .EAP file: t_objectfiles

File Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The file name - may be a logical file or a reference to a web address (using http://)
Type	<i>String</i>	Read/Write. File type.
Notes	<i>String</i>	Read/Write. Notes about the file.
Size	<i>String</i>	Read/Write. The file size.
FileDate	<i>String</i>	Read/Write. The file date when entry is created.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

File Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current File object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.5 Issue (Maintenance)

Issue

public Class

An Issue is either a "Change" or a "Defect" and is associated with the containing Element - and accessed through the Issues collection of an Element.

Associated table in .EAP file: t_objectproblems

Issue Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The Issue name - ie. the Issue itself.
Type	<i>Variant</i>	Read/Write. Issue type - may be "Defect" or "Change", "Issue" and "ToDo".
DateReported	<i>Date</i>	Read/Write. Date issue reported.
Status	<i>String</i>	Read/Write. The current status of the issue.
Notes	<i>String</i>	Read/Write. Issue Description.
Reporter	<i>String</i>	Read/Write. Person reporting issue.
Resolver	<i>String</i>	Read/Write. Person resolving issue.
DateResolved	<i>Date</i>	Read/Write. Date issue resolved.
Version	<i>String</i>	Read/Write. Version associated with issue. Note that this method is only available through the dispatch interface. eg. <code>Object ob = Issue;</code> <code>Print ob.Version;</code>
ResolverNotes	<i>String</i>	Read/Write. Notes entered by resolver about resolution.
ElementID	<i>Long</i>	Read/Write. ID of element associated with this issue.
Priority	<i>String</i>	Read/Write. Issue priority ... generally should use Low, Medium and High.
Severity	<i>String</i>	Read/Write. Issue severity. Should be marked as Low, Medium or High.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Issue Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Issue object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.6 Metric

Metric

public Class

A Metric is a named item with a weighting which may be associated with an element for purposes of building metrics about the model. Accessed through the Element Metrics collection

Associated table in .EAP file: t_objectmetrics

Metric Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The name of the Metric.
Type	<i>String</i>	Read/Write. The Metric type.
Weight	<i>Long</i>	Read/Write. A user defined weighting for estimation or metric purposes.
Notes	<i>String</i>	Read/Write. Notes about this metric.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Metric Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Metric object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.7 Requirement

Requirement

public Class

An Element Requirement object holds information about the responsibilities of an element in the context

of the model. Accessed using the Element Requirements collection

Associated table in .EAP file: t_objectrequires

Requirement Attributes

Attribute	Type	Notes
RequirementID	Long	Read only. A local ID for this requirement.
Name	String	Read/Write. The requirement itself.
Type	String	Read/Write. Requirement type.
Status	String	Read/Write. Current status of the requirement.
Stability	String	Read/Write. Estimated stability of the requirement.
Difficulty	String	Read/Write. Estimated difficulty to implement.
Priority	String	Read/Write. Assigned priority of the requirement.
LastUpdate	Date	Read/Write. Date requirement last updated.
Notes	String	Read/Write. Further notes about requirement.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.
ParentID	Long	Read only. The ElementID of the element to which this requirement applies.

Requirement Methods

Method	Type	Notes
Update ()	Boolean	Update the current Requirement object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	String	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.8 Resource

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

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. Name of resource (eg. person's name).
Role	<i>String</i>	Read/Write. Role they will play in implementing Element.
Time	<i>Long</i>	Read/Write. Time expected - numeric indicating number of days.
Notes	<i>String</i>	Read/Write. Descriptive notes.
PercentComplete	<i>Long</i>	Read/Write. Current percent complete figure.
DateStart	<i>Date</i>	Read/Write. Date to start work.
DateEnd	<i>Date</i>	Read/Write. Expected end date.
ExpectedHours	<i>Long</i>	Read/Write. The total expected time the task will run - may be in hours or days or some other unit.
ActualHours	<i>Long</i>	Read/Write. Time already expended on task. May be hours or days or other units.
History	<i>String</i>	Read/Write. Gets or sets history text.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Resource Methods

Method	Type	Notes
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the current Resource object after modification or appending a new item. If false is returned, check the GetLastError function for more information.

12.1.2.5.9 Risk

Risk

public Class

A Risk object represents a named risk associated with an element and may be used for project management purposes. Accessed through the Element Risks collection.

Associated table in .EAP file: t_objectrisks

Risk Attributes

Attribute	Type	Notes
Name	String	Read/Write. The risk.
Type	String	Read/Write. The risk type associated with this element.
Weight	Long	Read/Write. A weighting for estimation or metric purposes.
Notes	String	Read/Write. Further notes describing the risk.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Risk Methods

Method	Type	Notes
Update ()	Boolean	Update the current Risk object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	String	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.10 Scenario

Scenario

public Class

A Scenario corresponds to a Collaboration or Use Case instance. Each scenario is a path of execution through the logic of a Use Case. Scenarios may be added to using the Element Scenarios collection.

Associated table in .EAP file: t_objectscenarios

Scenario Attributes

Attribute	Type	Notes
Name	String	Read/Write. The Scenario name.
Type	String	Read/Write. The Scenario type (eg. "Basic Path").
Weight	Long	Read/Write. Currently used to position scenarios in the scenario list (ie. List Position).
Notes	String	Read/Write. Description of the Scenario. Usually contains the steps to execute the scenario.
XMLContent	String	Read/Write. A structured field which may contain scenario details in XML format. To be implemented in 2003.
ScenarioGUID	String	Read/Write. A unique ID for the scenario. Used to identify the scenario unambiguously within a model.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Scenario Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Scenario object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.11 TaggedValueTaggedValuepublic Class

A TaggedValue is a named property and value associated with an element and is accessed through the TaggedValues collection

Associated table in .EAP file: t_objectproperties

TaggedValue Attributes

Attribute	Type	Notes
PropertyID	<i>Long</i>	Read only. A local ID for the property.
PropertyGUID	<i>String</i>	Read/Write. A global ID for the property.
Name	<i>String</i>	Read/Write. Name of the property (Tag).
Value	<i>String</i>	Read/Write. The value assigned in this instance.
Notes	<i>String</i>	Read/Write. Further descriptive notes.
ElementID	<i>Long</i>	Read/Write. The local ID of the associated element.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

TaggedValue Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current TaggedValue object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.5.12 Test**Test****public Class**

A Test is a single Test Case applied to an Element. Tests are added and accessed through the Element Tests collection.

Associated table in .EAP file: t_objecttests

Test Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The Test Name
Type	<i>String</i>	Read/Write. The Test Type - eg. Load, Regression etc.
Class	<i>Long</i>	Read/Write. The Test Class: 1 = Unit Test, 2 = Integration Test, 3 = System test, 4 = Acceptance Test, 5 = Scenario Test.
Notes	<i>String</i>	Read/Write. Detailed notes about test to be carried out.
Input	<i>String</i>	Read/Write. Input data.
AcceptanceCriteria	<i>String</i>	Read/Write. The acceptance criteria for successful execution.
Status	<i>String</i>	Read/Write. Current status of test.
DateRun	<i>Date</i>	Read/Write. Date last run.
TestResults	<i>Variant</i>	Read/Write. Results of test.
RunBy	<i>String</i>	Read/Write. Person conducting test.
CheckedBy	<i>String</i>	Read/Write. Results confirmed by.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Test Methods

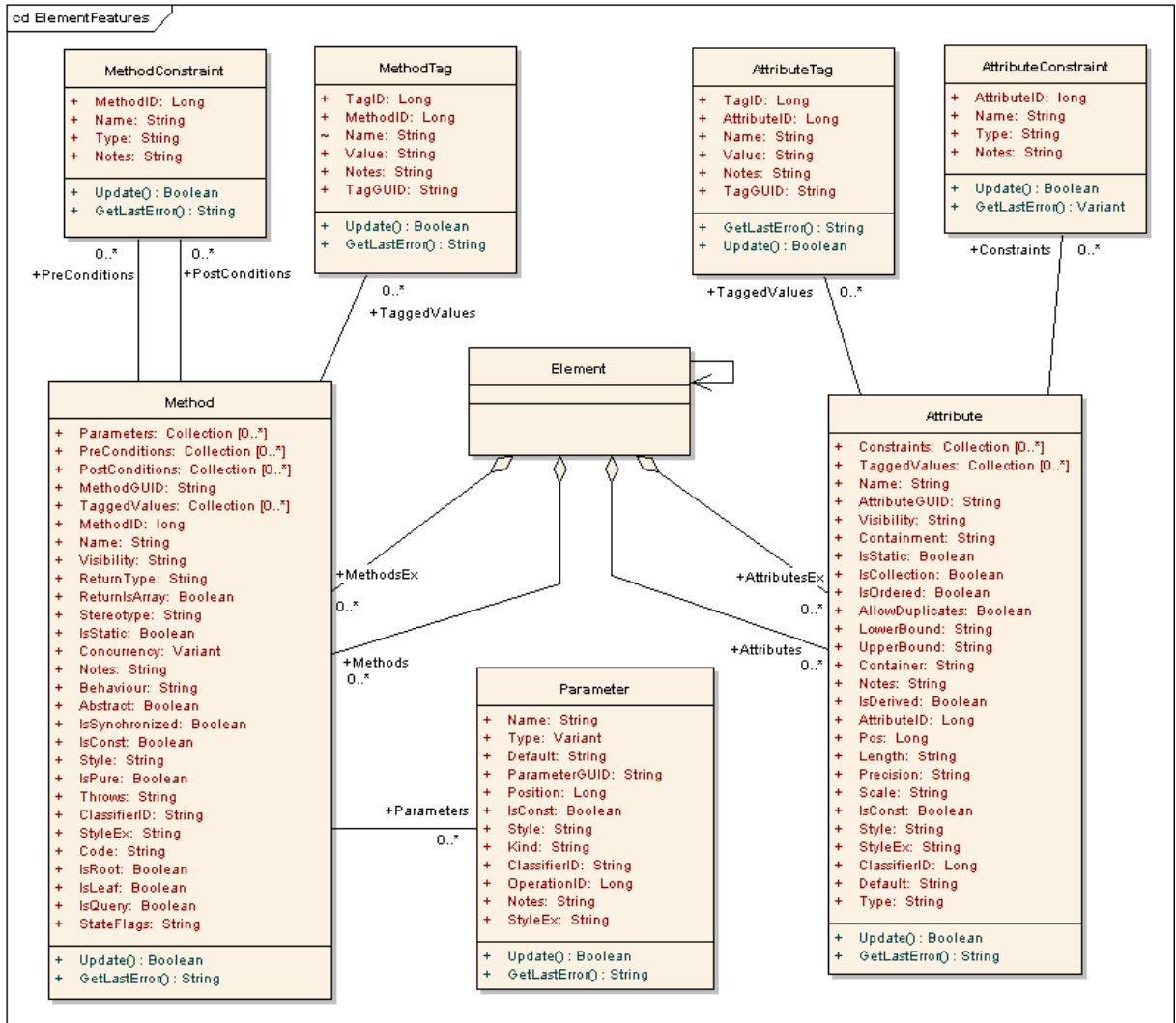
Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Test object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6 Element Features

This diagram illustrates the components associated with element features. These include Attributes and Methods - and the associated constraints and tagged values related to them.

It also includes the Parameter object which define the arguments associated with an operation (method).

Figure 5 : ElementFeatures



ElementFeatures

public Package

The Element Features package contains descriptions of the model interfaces that allow access to Operations and Attributes - and their associated tagged values and constraints.

12.1.2.6.1 Attribute

Attribute

public Class

An Attribute corresponds to a UML Attribute. It contains further collections for constraints and tagged values. Attributes are accessed from the Element Attributes collection.

Associated table in .EAP file: t_attribute

Attribute Attributes

Attribute	Type	Notes
Constraints	<i>Collection</i>	Read only. A collection of AttributeConstraints. Used to access and manage constraints associated with this Attribute.
TaggedValues	<i>Collection</i>	Read only. A collection of AttributeTags. Use to access and manage tagged values associated with this Attribute.
Name	<i>String</i>	Read/Write. The attribute name.
AttributeGUID	<i>String</i>	Read/Write. A globally unique ID for the current Attribute. System generated.
Visibility	<i>String</i>	Read/Write. The Scope of the Attribute. May be Private, Protected, Public or Package.
Containment	<i>String</i>	Read/Write. Type of Containment. May be "Not Specified", "By Reference" or "By Value".
IsStatic	<i>Boolean</i>	Read/Write. Indicates if the current attribute is a static feature or not.
IsCollection	<i>Boolean</i>	Read/Write. Indicates if the current feature is a collection or not.
IsOrdered	<i>Boolean</i>	Read/Write. Indicates if a collection is ordered or not.
AllowDuplicates	<i>Boolean</i>	Read/Write. Indicates if duplicates are allowed in the collection.
LowerBound	<i>String</i>	Read/Write. A value for the collection lower bound.
UpperBound	<i>String</i>	Read/Write. A value for the collection upper bound.
Container	<i>String</i>	Read/Write. The container type.
Notes	<i>String</i>	Read/Write. Further notes about this attribute.
IsDerived	<i>Boolean</i>	Read/Write. Indicates if attribute is derived (eg. a calculated value).
AttributeID	<i>Long</i>	Read only. Local ID number of Attribute.
Pos	<i>Long</i>	Read/Write. Position of Attribute in class attribute list.
Length	<i>String</i>	Read/Write. Attribute length where applicable.
Precision	<i>String</i>	Read/Write. Precision value.
Scale	<i>String</i>	Read/Write. Scale value.
IsConst	<i>Boolean</i>	Read/Write. Flag indicating if Attribute is Const or not.
Style	<i>String</i>	Read/Write. Further style information.
StyleEx	<i>String</i>	Read/Write. Advanced style settings. Use with care.
ClassifierID	<i>Long</i>	Read/Write. Classifier ID if appropriate - indicates base type associated with attribute if not a primitive type.
Default	<i>String</i>	Read/Write. Default value associated with attribute.
Type	<i>String</i>	Read/Write. The Attribute type (by name - also see ClassifierID).

Stereotype	<i>String</i>	Read/Write. Sets or gets the Stereotype for this attribute.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.
ParentID	<i>Long</i>	Read only. Returns the ElementID of the Element that this attribute is a part of.
TaggedValuesEx	<i>Collection</i>	Read only. Collection of TaggedValue objects belonging to the current Attribute and its parent attributes.

Attribute Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Attribute object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6.2 Attribute Constraint

AttributeConstraint

public Class

An AttributeConstraint is a constraint associated with the current Attribute.

Associated table in .EAP file: t_attributeconstraints

AttributeConstraint Attributes

Attribute	Type	Notes
AttributeID	<i>Long</i>	Read/Write. ID of Attribute this constraint applies to.
Name	<i>String</i>	Read/Write. The Constraint.
Type	<i>String</i>	Read/Write. Type of Constraint.
Notes	<i>String</i>	Read/Write. Descriptive notes about constraint.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

AttributeConstraint Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current AttributeConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6.3 Attribute Tag

AttributeTag

public Class

An AttributeTag represents a Tagged Value associated with an Attribute

Associated table in .EAP file: t_attributetag

AttributeTag Attributes

Attribute	Type	Notes
TagID	<i>Long</i>	Read only. Local ID to identify tagged value.
AttributeID	<i>Long</i>	Read/Write. Local ID of Attribute associated with this tagged value.
Name	<i>String</i>	Read/Write. Name or Tag.
Value	<i>String</i>	Read/Write. Value associated with this tag.
Notes	<i>String</i>	Read/Write. Descriptive notes.
TagGUID	<i>String</i>	Read/Write. A globally unique ID for this tagged value.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

AttributeTag Methods

Method	Type	Notes
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the current AttributeTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information.

12.1.2.6.4 Method

Method

public Class

A Method represents a UML operation. It is accessed from the Element Methods collection and includes collections for parameters, constraints and tagged values

Associated table in .EAP file: t_operation

Method Attributes

Attribute	Type	Notes
Parameters	<i>Collection</i>	Read only. The Parameters collection for the current Method. Use to add and access Parameter objects for the current Method.
PreConditions	<i>Collection</i>	Read only. PreConditions (constraints) as they apply to this method. Returns a MethodConstraint object of type "pre".
PostConditions	<i>Collection</i>	Read only. PostConditions (constraints) as they apply to this method. Returns a MethodConstraint object of type "post".
MethodGUID	<i>String</i>	Read/Write. A globally unique ID for the current Method. System generated.
TaggedValues	<i>Collection</i>	Read only. TaggedValues collection for the current method. Access a list of MethodTag objects.
MethodID	<i>Long</i>	Read only. A local ID for the current method - only valid within this .EAP file.
Name	<i>String</i>	Read/Write. The Method name.
Visibility	<i>String</i>	Read/Write. The method scope - Public, Protected, Private or Package.
ReturnType	<i>String</i>	Read/Write. Return type for the method - may be a primitive data type or a class or interface type.
ReturnsToArray	<i>Boolean</i>	Read/Write. Flag to indicate return value is an array.
Stereotype	<i>String</i>	Read/Write. The method stereotype (optional).
IsStatic	<i>Boolean</i>	Read/Write. Flag to indicate a static method.
Concurrency	<i>Variant</i>	Read/Write. Concurrency type of method.
Notes	<i>String</i>	Read/Write. Descriptive notes about the Method.
Behavior	<i>String</i>	Read/Write. Some further explanatory behavior notes (eg. pseudocode).
Abstract	<i>Boolean</i>	Read/Write. Flag indicating if Method is abstract (1) or not (0).
IsSynchronized	<i>Boolean</i>	Read/Write. Flag indicating a Synchronized method call.
IsConst	<i>Boolean</i>	Read/Write. Flag indicating method is Const.
Style	<i>String</i>	Read/Write. Extended style information about method.
IsPure	<i>Boolean</i>	Read/Write. Flag indicating method is defined as Pure in C++.
Throws	<i>String</i>	Read/Write. Exception information.
ClassifierID	<i>String</i>	Read/Write. ClassifierID that applies to the ReturnType.
StyleEx	<i>String</i>	Read/Write. Advanced style settings. Not currently used.
Code	<i>String</i>	Read/Write. Optional field to hold Method Code (currently not used in EA).
IsRoot	<i>Boolean</i>	Read/Write. Flag to indicate if Method is Root.
IsLeaf	<i>Boolean</i>	Read/Write. Flag to indicate Method is Leaf (cannot be overridden).
IsQuery	<i>Boolean</i>	Read/Write. Flag to indicate method is a query (ie. does not alter class variables).
StateFlags	<i>String</i>	Read/Write. Some flags as applied to methods in State elements.

Pos	<i>Long</i>	Read/Write. Specifies the position of the method within the set of operations defined for a class.
ParentID	<i>Long</i>	Read only. An optional ID of an element that 'owns' this diagram - eg. a Sequence diagram owned by a Use Case.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Method Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Method object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6.5 Method Constraint

MethodConstraint

public Class

A Method constraint is a condition imposed on a Method. It is accessed through either the Method PreConditions or Method PostConditions collection.

Associated table in .EAP file: t_operationpres and t_operationposts

MethodConstraint Attributes

Attribute	Type	Notes
MethodID	<i>Long</i>	Read/Write. The local ID of the associated method.
Name	<i>String</i>	Read/Write. The Name of the constraint.
Type	<i>String</i>	Read/Write. The constraint type.
Notes	<i>String</i>	Read/Write. Descriptive notes about this constraint.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

MethodConstraint Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current MethodConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6.6 Method Tag

MethodTag

public Class

A MethodTag is a tagged value associated with a method

Associated table in .EAP file: t_operationtag

MethodTag Attributes

Attribute	Type	Notes
TagID	<i>Long</i>	Read only. A unique ID for this tagged value.
MethodID	<i>Long</i>	Read/Write. The ID of the associated Method.
Name	<i>String</i>	Read/Write. The Tag or name of the property.
Value	<i>String</i>	Read/Write. A value to apply to this tag.
Notes	<i>String</i>	Read/Write. Descriptive notes about this item.
TagGUID	<i>String</i>	Read/Write. A uniqueID for this tagged value.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

MethodTag Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current MethodTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6.7 Parameter

Parameter

public Class

A Parameter object represents a method argument and is accessed through the Method Parameters collection.

Associated table in .EAP file: t_operationparams

Parameter Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The Parameter name - must be unique for a single Method.
Type	<i>Variant</i>	Read/Write. The Parameter type - may be a primitive type or defined classifier.
Default	<i>String</i>	Read/Write. A default value for this Parameter.
ParameterGUID	<i>String</i>	Read/Write. A globally unique ID for the current Parameter. System generated.
Position	<i>Long</i>	Read/Write. The position in the argument list.
IsConst	<i>Boolean</i>	Read/Write. Flag indicating the parameter is Const (cannot be altered).
Style	<i>String</i>	Read/Write. Some style information.
Kind	<i>String</i>	Read/Write. The parameter Kind - in, inout, out, return.
ClassifierID	<i>String</i>	Read/Write. A ClassifierID for the Parameter if known.
OperationID	<i>Long</i>	Read only. ID of the Method associated with this parameter.
Notes	<i>String</i>	Read/Write. Descriptive Notes.
StyleEx	<i>String</i>	Read/Write. Advanced style settings. Not currently used.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Parameter Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Parameter object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.6.8 Partitions

Partitions

public Collection

A collection of internal element partitions (regions). This is commonly seen in Activities, States, Boundary, Diagram Frame and similar elements. Not all elements support partitions.

This collection contains a set of Partition elements. The set is read/write: information is not saved until the host element is saved, so ensure that you call the Element.Save method after making changes to a Partition.

Partition Attributes

Attribute	Type	Notes
Name	<i>String</i>	Read/Write. The Partition name - may represent a condition or constraint in some cases.
Note	<i>String</i>	Read/Write. A free text note associated with this Partition.
Operator	<i>String</i>	Read/Write. An optional operator value that specifies the Partition type.
Size	<i>String</i>	Read/Write. Vertical or horizontal width of partition in pixels.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

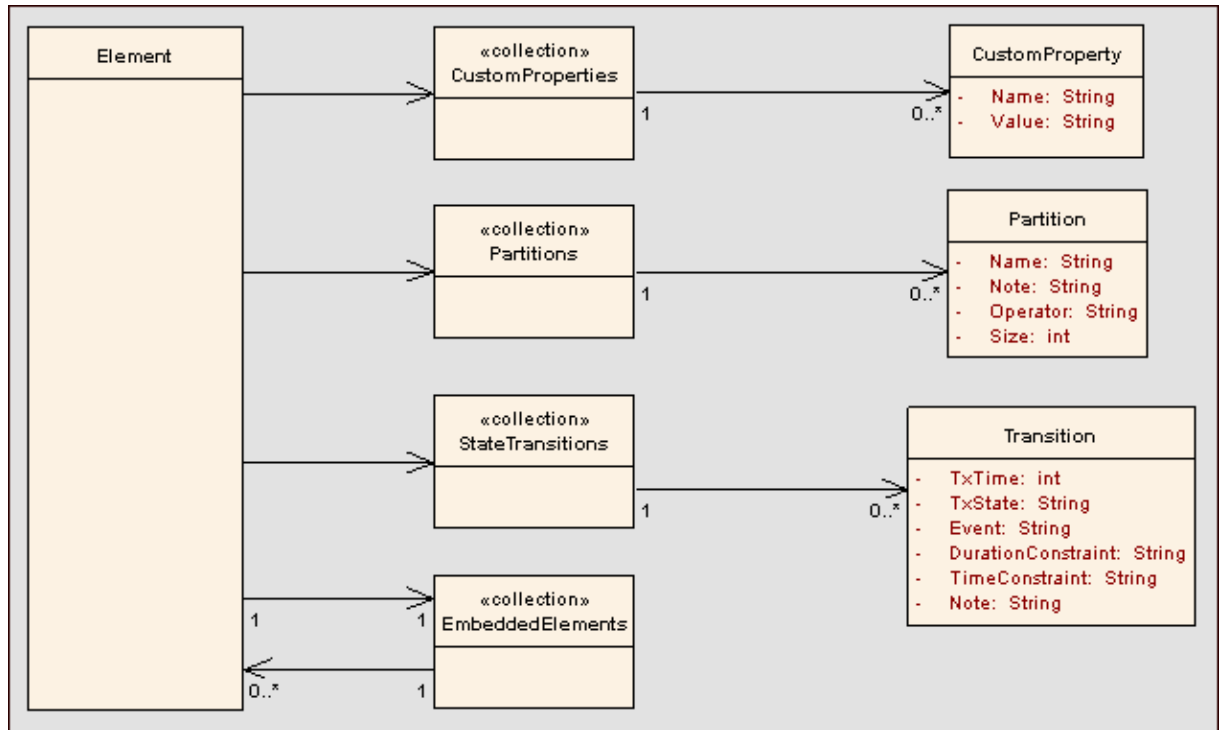
12.1.2.6.9 Embedded Elements

Embedded Elements

public Collection

In UML 2.0 an element may have one or more embedded elements such as Ports, Pins, Parameters, ObjectNodes etc. These are attached to the boundary of the host element and may not be moved off the element. They are owned by their host element. This collection gives easy access to the set of elements embedded on the surface of an element. Note that some embedded elements may have their own embedded element collection (eg. Ports may have interfaces embedded on them).

The Embedded Elements collection contains Element objects.



12.1.2.6.10 Transitions

Transitions

public Collection

Applies only to Timeline elements. A Timeline element displays 0 or more state transitions at set times on its extent. This collection lets you access the transition set. You can also access additional information by referring to the Connectors associated with the Timeline - and referencing messages passed between timelines. Note that any changes made to elements in this collection are only saved when the main Element is saved.

Partition Attributes

Attribute	Type	Notes
TxTime	String	Read/Write. The time that the transition occurs. Value depends on range set in diagram.
TxState	String	Read/Write. The state to transition to. Defined in the Timeline Properties dialog.
Event	String	Read/Write. Event (optional) that initiated transition.
DurationConstraint	String	Read/Write. A constraint on the time duration that the transition will take.
TimeConstraint	String	Read/Write. A constraint on when the transition has to be complete by.
Note	String	Read/Write. A free text note.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

12.1.2.6.11 Custom Properties

Custom Properties

public Collection

The CustomProperties collection contains 0 or more Cust Properties associated with the current element. These properties provide advanced UML configuration options, and may not be added to or deleted. The value of each property may be set.

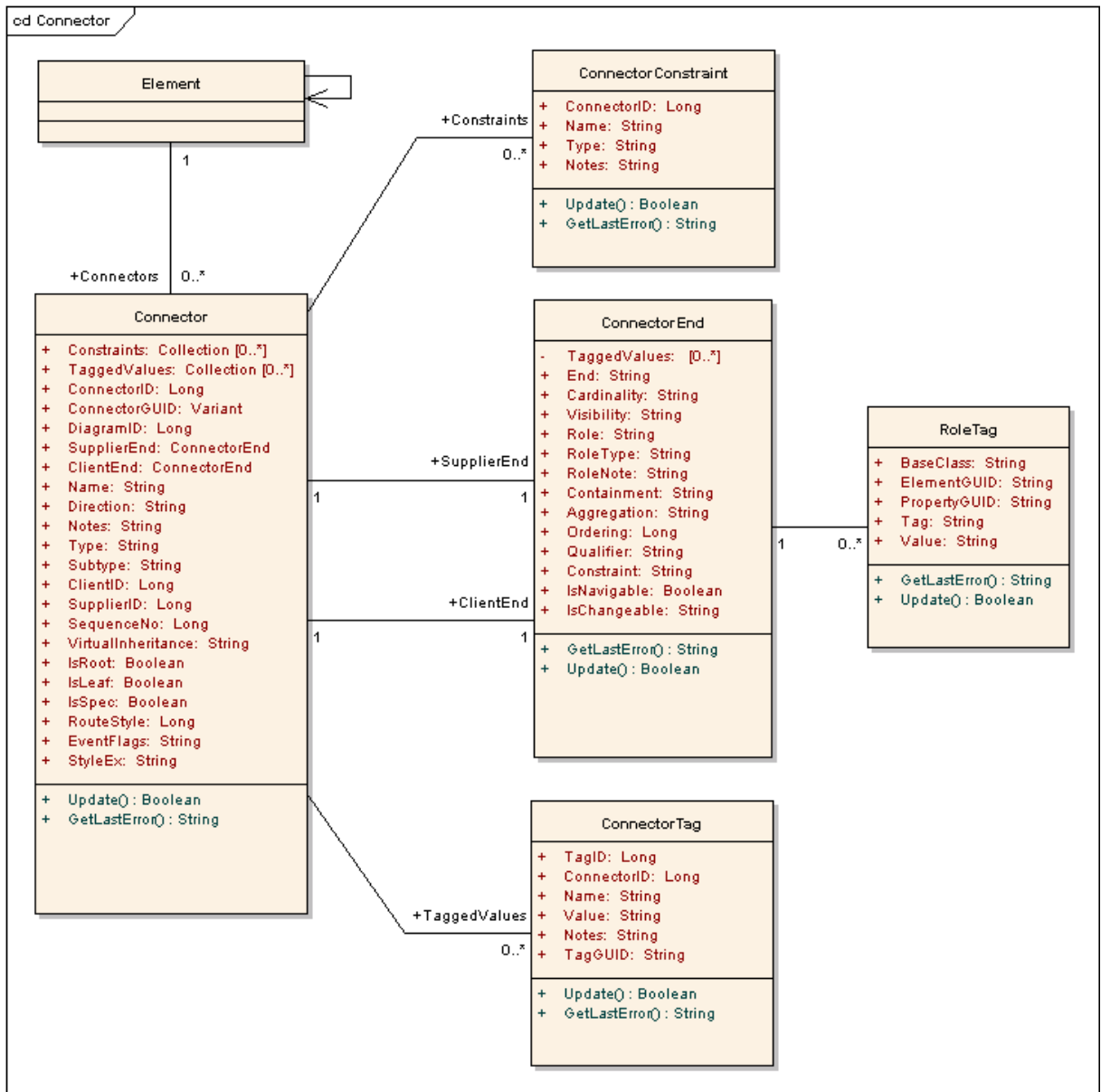
Note that the number and type of property will vary depending on the actual element.

CustomProperty

Attribute	Type	Notes
Name	<i>String</i>	Read-only. The Custom Property name.
Value	<i>String</i>	Read/Write. The value associated with this custom property. Mayb be a string, the boolean values 'true' or 'false' or an enumeration value from a defined list. The UML 2.0 specification in general provides information on enumeration kinds relevant here.
ObjectType	<u>ObjectType</u>	Read only. Distinguishes objects referenced through Dispatch interface.

12.1.2.7 Connector

Figure 6 : Connector



Connector

public Package

The Connector package details how connections between elements are accessed and managed.

12.1.2.7.1 Connector

Connector

public Class

A Connector represents the various kinds of Connections between UML Elements. It is accessed from either the Client or Supplier Element - using the Connectors collection of that element. When creating a

new connector you must assign it a valid type. These are:

- Aggregation
- Association
- Collaboration
- Dependency
- Generalization
- Instantiation
- Nesting
- NoteLink
- Realization
- Sequence
- StateFlow
- UseCase

Associated table in .EAP file: t_connector

Connector Attributes

Attribute	Type	Notes
Constraints	<i>Collection</i>	Read only. Collection of Constraint objects.
TaggedValues	<i>Collection</i>	Read only. Collection of TaggedValue objects.
ConnectorID	<i>Long</i>	Read only. Local identifier for the current connector. System generated.
ConnectorGUID	<i>Variant</i>	Read only. A globally unique ID for the current connector. System generated.
Color	<i>Long</i>	Read/Write. Sets the color of the connector.
DiagramID	<i>Long</i>	Read/Write. The DiagramID of the connector.
SupplierEnd	<i>ConnectorEnd</i>	Read only. A pointer to the a ConnectorEnd object that represents the supplier end of the relationship.
ClientEnd	<i>ConnectorEnd</i>	Read only. A pointer to the a ConnectorEnd object that represents the client end of the relationship.
Name	<i>String</i>	Read/Write. The connector name.
Direction	<i>String</i>	Read/Write. Connector Direction. May be set to one of the following: 1. Unspecified, 2. Bi-Directional, 3. Source -> Destination, 4. Destination -> Source.
Notes	<i>String</i>	Read/Write. Descriptive notes about connection.
Type	<i>String</i>	Read/Write. Connector type. Valid types are held in the t_connectortypes table in the .EAP file.
Subtype	<i>String</i>	Read/Write. A possible subtype to refine the meaning of the connection.
ClientID	<i>Long</i>	Read/Write. The ElementID of the Client (start) Element.
SupplierID	<i>Long</i>	Read/Write. An ElementID (long) of the supplier (end) Element.
SequenceNo	<i>Long</i>	Read/Write. The SequenceNo of the connection.
VirtuallInheritance	<i>String</i>	Read/Write. For Generalization indicates if inheritance is virtual.
IsRoot	<i>Boolean</i>	Read/Write. Flag indicating connector is a root.
IsLeaf	<i>Boolean</i>	Read/Write. Flag indicating connector is a leaf.
IsSpec	<i>Boolean</i>	Read/Write. Flag indicating connector is a Specification.
RouteStyle	<i>Long</i>	Read/Write. The route style.
EventFlags	<i>String</i>	Read/Write. Structure to hold a variety of flags concerned with event signaling on messages.
StyleEx	<i>String</i>	Read/Write. Advanced style settings. Not currently used.
Stereotype	<i>String</i>	Read/Write. Sets or gets the Stereotype for this connector end.
TransitionEvent	<i>String</i>	Read/Write. See the Transition topic in this document for appropriate values.
TransitionGuard	<i>String</i>	Read/Write. See the Transition topic in this document for appropriate values.
TransitionAction	<i>String</i>	Read/Write. See the Transition topic in this document for appropriate values.
Width	<i>Long</i>	Read/Write. Specifies the width of the connector.
CustomProperties	<i>Collection</i>	Read only. Returns a collection of Advanced properties associated with an element in the form of CustomProperty objects.
ObjectType	ObjectType	Read only. Distinguishes objects referenced

Connector Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Connector object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.7.2 Connector ConstraintsConnectorConstraintpublic Class

A ConnectorConstraint holds information about special conditions that apply to a Connector. It is accessed through the Connector Constraints collection

Associated table in .EAP file: t_connectorconstraints

ConnectorConstraint Attributes

Attribute	Type	Notes
ConnectorID	<i>Long</i>	Read/Write. A local ID value (long) - system generated.
Name	<i>String</i>	Read/Write. The constraint name.
Type	<i>String</i>	Read/Write. The constraint type.
Notes	<i>String</i>	Read/Write. Notes about this constraint.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

ConnectorConstraint Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current ConnectorConstraint object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs

12.1.2.7.3 Connector EndConnectorEnd

public Class

A ConnectorEnd contains information about a single end of a Connector. It may be a Supplier or a Client.
A ConnectorEnd is accessed from the Connector as either the ClientEnd or SupplierEnd

Associated table in .EAP file: derived from t_connector

ConnectorEnd Attributes

Attribute	Type	Notes
TaggedValues	<i>private:</i>	Read only collection.
End	<i>String</i>	Read only. The End this connectorEnd object applies to - Client or Supplier.
Cardinality	<i>String</i>	Read/Write. Cardinality associated with this end.
Visibility	<i>String</i>	Read/Write. Scope associated with this connection end. Valid types are: Public, Private, Protected and Package.
Role	<i>String</i>	Read/Write. The connection end role.
RoleType	<i>String</i>	Read/Write. The Role type applied to this end of the connection.
RoleNote	<i>String</i>	Read/Write. Notes associated with the role of this connection end.
Containment	<i>String</i>	Read/Write. Containment type applied to this connection end.
Aggregation	<i>Long</i>	Read/Write. Aggregation as it applies to this end. Valid values are: 0 = None, 1 = Shared, 2 = Composite.
Ordering	<i>Long</i>	Read/Write. Ordering for this connection end.
Qualifier	<i>String</i>	Read/Write. A qualifier that may apply to connection end.
Constraint	<i>String</i>	Read/Write. A constraint that may be applied to this connection end.
IsNavigable	<i>Boolean</i>	Read/Write. Flag indicating this end is navigable from the other.
IsChangeable	<i>String</i>	Read/Write. Flag indicating this end is changeable or not.
Stereotype	<i>String</i>	Read/Write. Sets or gets the Stereotype for this connector end.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

ConnectorEnd Methods

Method	Type	Notes
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the current ConnectorEnd object after modification or appending a new item. If false is returned, check the GetLastError function for more information.

12.1.2.7.4 Connector Tag**ConnectorTag****public Class**

A ConnectorTag is a tagged value for a connector and is accessed through the Connector TaggedValues collection

Associated table in .EAP file: t_connectortag

ConnectorTag Attributes

Attribute	Type	Notes
TagID	<i>Long</i>	Read only. A local ID to identify the tagged value.
ConnectorID	<i>Long</i>	Read/Write. The local ID of the associated connector.
Name	<i>String</i>	Read/Write. The tag or name.
Value	<i>String</i>	Read/Write. A value associated with the tag.
Notes	<i>String</i>	Read/Write. Descriptive notes associated with this tagged value.
TagGUID	<i>String</i>	Read/Write. A globally unique ID for this tagged value.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

ConnectorTag Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current ConnectorTag object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.

12.1.2.7.5 Role Tag**RoleTag****public Class**

This interface provides access to the association role tagged values. Each connector end has a RoleTag collection that can be accessed to add, delete and access the RoleTags.

In code you will do something like (where con is a connector object):

Code fragment for accessing a RoleTag in VB.NET:

```
client = con.ClientEnd
client.Role = "m_client"
client.Update()
tag = client.TaggedValues.AddNew("tag", "value")
```

```

tag.Update()
tag = client.TaggedValues.AddNew("tag2", "value2")
tag.Update()
client.TaggedValues.Refresh()
For idx = 0 To client.TaggedValues.Count - 1
    tag = client.TaggedValues.GetAt(idx)
    Console.WriteLine(tag.Tag)
    client.TaggedValues.DeleteAt(idx, False)
Next
tag = Nothing

```

RoleTag Attributes

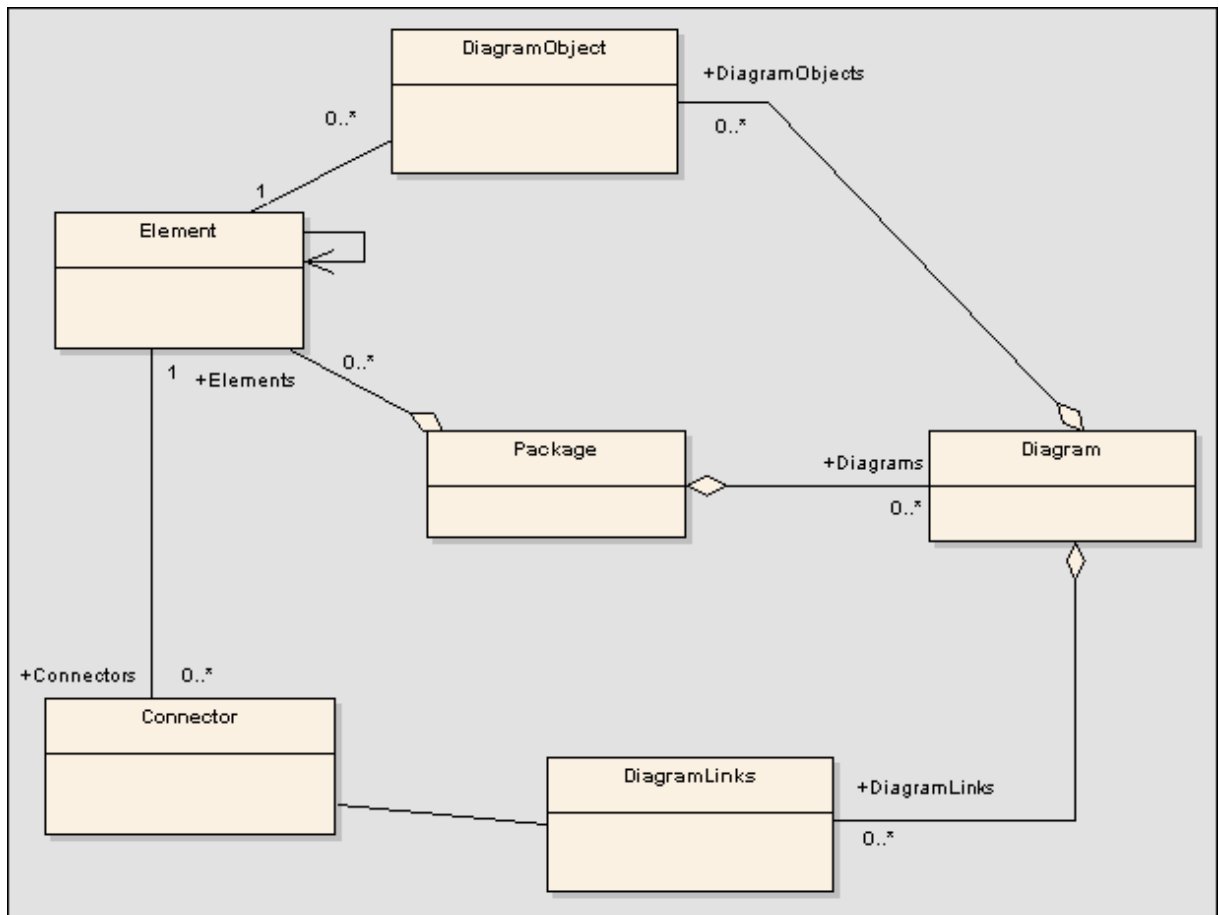
Attribute	Type	Notes
BaseClass	<i>String</i>	Read/Write. Indicates the role end - set to ASSOCIATION_SOURCE or ASSOCIATION_TARGET.
ElementGUID	<i>String</i>	Read/Write. GUID of the Connector with which this role tag is associated.
PropertyGUID	<i>String</i>	Read/Write. A system generated GUID to identify the tagged value.
Tag	<i>String</i>	Read/Write. The actual tag name.
Value	<i>String</i>	Read/Write. The value associated with this tag.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

RoleTag Methods

Method	Type	Notes
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the RoleTag after changes or on initial creation.

12.1.2.8 Diagram

Figure 7 : Diagram



Diagram

public Package

The Diagram package has information on a Diagram and DiagramObjects and DiagramLinks - which are the instances of Elements within a diagram.

12.1.2.8.1 Diagram

Diagram

public Class

A Diagram corresponds to a single EA diagram. It is accessed through the Package Diagrams collection and in turn contains a collection of diagram objects and diagram links. Adding to the DiagramObjects collection will add an element to the Diagram (the Element must already exist). When adding a new diagram, you must set the diagram type to a valid type - these are:

- Activity
- Analysis
- Component
- Custom
- Deployment
- Logical
- Sequence
- Statechart

- Use Case

Note: Use the Analysis type for a Collaboration Diagram.

Associated table in .EAP file: t_diagram

Diagram Attributes

Attribute	Type	Notes
DiagramObjects	<i>Collection</i>	Read only. A collection of references to DiagramObjects. A DiagramObject is an instance of an Element in a Diagram - and includes size and display characteristics.
DiagramLinks	<i>Collection</i>	Read only. A list of DiagramLink objects - each containing information about the display characteristics of a connector in a diagram. Note: A Diagram link is only created once a user modifies a connector in a diagram in some way. Until this condition has been met default values are used and the diagram link is not in use.
DiagramID	<i>Long</i>	Read only. A local ID for the diagram.
PackageID	<i>Long</i>	Read/Write. An ID of the package that this diagram belongs to.
ParentID	<i>Long</i>	Read/Write. An optional ID of an element that 'owns' this diagram - eg. a Sequence diagram owned by a Use Case.
Type	<i>String</i>	Read only. The Diagram Type. See the t_diagramtypes table in the .EAP file for more information.
Name	<i>String</i>	Read/Write. The diagram name.
Version	<i>String</i>	Read/Write. The version of the diagram.
Author	<i>String</i>	Read/Write. The author.
ShowDetails	<i>String</i>	Read/Write. Flag to indicate Diagram Details text should be shown.
ShowPublic	<i>Boolean</i>	Read/Write. Flag to show or hide Public features.
ShowPrivate	<i>Boolean</i>	Read/Write. Flag to show or hide Private features.
ShowProtected	<i>Boolean</i>	Read/Write. Flag to show or hide Protected features.
Orientation	<i>String</i>	Read/Write. Page orientation - use "P" or "L" for Portrait or Landscape respectively.
cx	<i>Long</i>	Read/Write. The X dimension of the diagram (800 is default).
cy	<i>Long</i>	Read/Write. The Y dimension of diagram (1100 is default).
Scale	<i>Long</i>	Read/Write. The zoom scale - 100 is default.
CreatedDate	<i>Date</i>	Read/Write. The date diagram created.
ModifiedDate	<i>Variant</i>	Read/Write. The date the diagram was last modified.
HighlightImports	<i>Boolean</i>	Read/Write. Flag to indicate elements from other packages should be highlighted.
ShowPackageContents	<i>Boolean</i>	Read/Write. Flag to indicate package contents should be shown in the current diagram.
StyleEx	<i>Long</i>	Read/Write. Advanced style settings. Not currently used.
ExtendedStyle	<i>String</i>	Read/Write. An extended style attribute.
IsLocked	<i>Boolean</i>	Read/Write. Flag indicating this diagram is locked or not.
DiagramGUID	<i>Variant</i>	Read/Write. A globally unique ID for this diagram.
Swimlanes	<i>String</i>	Read/Write. Information on swimlanes contained in the diagram.

Notes	<i>String</i>	Read/Write. Set/retrieve notes for this diagram.
Stereotype	<i>String</i>	Read/Write. Sets or gets the stereotype for this diagram.
SelectedObjects	<i>Collection</i>	Read only. Gets a collection representing the currently selected elements on the diagram.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Diagram Methods

Method	Type	Notes
Update ()	<i>Boolean</i>	Update the current Diagram object after modification or appending a new item. If false is returned, check the GetLastError function for more information.
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs

12.1.2.8.2 Diagram Links

DiagramLinks

public Class

A DiagramLink is an object which holds display information about a connection between two elements in a specific diagram. It includes the custom points, display appearance & etc. Accessed from the Diagram DiagramLinks collection.

Associated table in .EAP file: t_diagramlinks

DiagramLinks Attributes

Attribute	Type	Notes
DiagramID	<i>Long</i>	Read/Write. The local ID for the associated diagram.
ConnectorID	<i>Long</i>	Read/Write. The ID of the associated connector.
Geometry	<i>String</i>	Read/Write. The geometry associated with the current connector in this diagram.
IsHidden	<i>Boolean</i>	Read/Write. Flag to indicate if this item is hidden or not.
Path	<i>String</i>	Read/Write. The path of the connector in this diagram.
Style	<i>String</i>	Read/Write. Additional style information - eg. color, thickness.
InstanceID	<i>Long</i>	Read only attribute. Holds the link identifier for the current model.
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

DiagramLinks Methods

Method	Type	Notes
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the current DiagramLink object after modification or appending a new item. If false is returned, check the GetLastError function for more information.

12.1.2.8.3 Diagram Objects

DiagramObjects

public Class

The DiagramObjects collection holds a list of element ID's and presentation information that indicates what will be displayed in a diagram and how it will be shown

Associated table in .EAP file: t_diagramobjects

DiagramObjects Attributes

Attribute	Type	Notes
DiagramID	<i>Long</i>	Read/Write. The ID of the associated diagram (long).
ElementID	<i>Long</i>	Read/Write. The ElementID of the object instance in this diagram.
left	<i>Long</i>	Read/Write. The left position of the element.
right	<i>Long</i>	Read/Write. The right position of the element.
top	<i>Long</i>	Read/Write. The top position of the element.
bottom	<i>Long</i>	Read/Write. The bottom position of the element.
InstanceID	<i>Long</i>	Read/Write. Read only attribute. Holds the link identifier for the current model.
Sequence	<i>Long</i>	Read/Write. The sequence position when loading into diagram (affects Z order).
Style	<i>Variant</i>	Read/Write. Additional style information for this object BCol = Background Color BFol = Font Color LCol = Line Color LWth = Line Width The color value is a decimal value. eg. DiagObj.Style = "BCol=35723;BFol=9342520;LCol=9342520;LWth=1;"
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

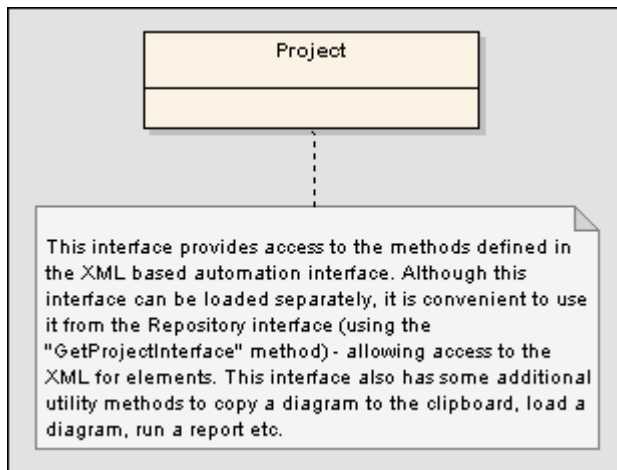
DiagramObjects Methods

Method	Type	Notes
GetLastError ()	<i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
Update ()	<i>Boolean</i>	Update the current DiagramObject object after modification or appending a new item. If false is returned, check the GetLastError function for more information.

12.1.2.9 Project Interface

The EA.Project interface. This is the XML based interface to EA elements - and also includes some utility functions. You can get a pointer to this interface using the Repository.GetProjectInterface method.

Figure 8 : Project Interface



Project Interface

public Package

This package contains details on the XML based automation interface and its use.

12.1.2.9.1 Project

Project

public Class

Implements: CCmdTarget. The Project interface can be accessed from the Repository using GetProjectInterface(). The returned interface provides access to the XML based automation interface present in EA. Use this interface to get XML for the various internal elements and to run some utility functions to load diagrams, run reports & etc.

Project Attributes

Attribute	Type	Notes
ObjectType	ObjectType	Read only. Distinguishes objects referenced through Dispatch interface.

Project Methods

Method	Type	Notes
LoadProject (<i>String</i>)	protected abstract: <i>Boolean</i>	param: FileName [<i>String</i> - in] Load an EA project file. Do not use this method if you have accessed the Project interface from the Repository, which will already have loaded a file.
ReloadProject ()	protected abstract: <i>Boolean</i>	Reload the current project. Convenience method to refresh the current loaded project (in case of outside changes to the .EAP file).
LoadDiagram (<i>String</i>)	protected abstract: <i>Boolean</i>	param: DiagramGUID [<i>String</i> - in] Load a diagram by its GUID. Note that EA expects this GUID in XML format ... if you retrieve the GUID using the Diagram interface, you will need to convert to XML format - use the GUIDtoXML and XMLtoGUID functions to do this.
SaveDiagramImageToFile (<i>String</i>)	protected abstract: <i>String</i>	param: FileName [<i>String</i> - in] The filename of the image to save. Save a diagram image of the current diagram to file.
GetElement (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] The GUID of the element to retrieve XML for. GUID must be in XML format (use GUIDtoXML to change an EA GUID to an XML GUID). Get XML for the specified element.
EnumViews ()	protected abstract: <i>String</i>	Enumerate the views for a project. Returned as an XML document.
EnumPackages (<i>String</i>)	protected abstract: <i>String</i>	param: PackageGUID [<i>String</i> - in] Get a list of packages inside another. Returned as XML. Supply the GUID of the parent package - in XML format.
EnumElements (<i>String</i>)	protected abstract: <i>String</i>	param: PackageGUID [<i>String</i> - in] GUID of package to get list of elements for. Must be in XML format. List elements inside a package. Returned in XML format.
EnumLinks (<i>String</i>)	protected abstract: <i>String</i>	param: PackageID [<i>String</i> - in] The package to get all associated links for.
EnumDiagrams (<i>String</i>)	protected abstract: <i>String</i>	param: PackageGUID [<i>String</i> - in] The GUID of the package to list diagrams for. Must be in XML format. Get an XML list of all diagrams in a specified package.
EnumDiagramElements (<i>String</i>)	protected abstract: <i>String</i>	param: DiagramGUID [<i>String</i> - in] The diagram GUID (in XML format) of the diagram to get elements for Get a list of all elements contained in a diagram - in XML format.

EnumDiagramLinks (<i>String</i>)	protected abstract: <i>String</i>	param: DiagramID [<i>String</i> - in] The Diagram ID to get links for. Get a list of links appearing in a diagram (in XML).
GetLink (<i>String</i>)	protected abstract: <i>String</i>	param: LinkGUID [<i>String</i> - in] The GUID (in XML format) to get details of. Get connector details in XML format.
GetDiagram (<i>String</i>)	protected abstract: <i>String</i>	param: DiagramGUID [<i>String</i> - in] The diagram ID (in XML format) Get diagram details in XML format.
GetElementConstraints (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get constraints (in XML) for an element. Supply the element ID in XML format.
GetElementEffort (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get effort (in XML format) for an element.
GetElementMetrics (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get metrics in XML format for an element.
GetElementFiles (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get files for an element in XML format.
GetElementRequirements (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get a list of requirements for an element in XML format.
GetElementProblems (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get a list of Issues (problems) associated with an element.
GetElementResources (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get a resource list (in XML) for an element.
GetElementRisks (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get a list of risks associated with an element - in XML format.
GetElementScenarios (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get a list of scenarios for an element - in XML format.
GetElementTests (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get a list of tests for an element (in XML format).
ShowWindow (<i>Long</i>)	protected abstract: <i>void</i>	param: Show [<i>Long</i> - in] Show or Hide the EA User Interface.

Exit ()	protected abstract: <i>void</i>	Exit the current instance of EA - this function is maintained for backward compatibility and should never be called. EA will automatically disappear when you are no longer using any of the provided objects.
PutDiagramImageOnClipboard (<i>String, Long</i>)	protected abstract: <i>Boolean</i>	param: DiagramGUID [<i>String</i> - in] param: Type [<i>Long</i> - in] Place an image of the current diagram on the clipboard.
PutDiagramImageToFile (<i>String, String, Long</i>)	protected abstract: <i>Boolean</i>	param: DiagramGUID [<i>String</i> - in] param: Filename [<i>String</i> - in] param: Type [<i>Long</i> - in] If type = 0 then it will be metafile If type = 1 then it will use the file type from the name extension (ie. .bmp, .jpg, .gif, .png, .tga) Place an image of the current diagram to file.
ExportPackageXML (<i>String, XMIMType, Long, Long, Long, Long, String</i>)	protected abstract: <i>String</i>	param: PackageGUID [<i>String</i> - in] The GUID of the package to export in XML format param: XMIMType [<i>EnumXMIMType</i> - in] Specifies the XML type and version information. See XMIMType Enum for allowable values. param: DiagramXML [<i>Long</i> - in] True if XML for diagrams is required param: DiagramImage [<i>Long</i> - in] True if images for diagrams should be created at the same time param: FormatXML [<i>Long</i> - in] True if XML output should be formatted prior to saving param: UseDTD [<i>Long</i> - in] True if a DTD should be used param: FileName [<i>String</i> - in] The filename to output to. Export XMI for a specified package.
EnumProjects ()	protected abstract: <i>String</i>	Get a list of projects in the current file - corresponds to Model in Repository.

EnumViewEx (<i>String</i>)	protected abstract: <i>String</i>	param: ProjectGUID [<i>String</i> - in] Get a list of Views in the current project.
RunReport (<i>String, String, String</i>)	protected abstract: <i>void</i>	param: PackageGUID [<i>String</i> - in] param: TemplateName [<i>String</i> - in] param: FileName [<i>String</i> - in] Run a named report - RTF.
GetLastError ()	protected abstract: <i>String</i>	Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used since an exception will be thrown when an error occurs.
GetElementProperties (<i>String</i>)	protected abstract: <i>String</i>	param: ElementGUID [<i>String</i> - in] Get tagged values for a specified element.
GUIDtoXML (<i>String</i>)	<i>String</i>	param: GUID [<i>String</i> - in] The EA style GUID to convert to XML format Change an internal GUID to the form used in XML
XMLtoGUID (<i>String</i>)	<i>String</i>	param: GUID [<i>String</i> - in] The XML style GUID to convert to EA internal format. Change a GUID in XML format to the form used inside EA.
RunHTMLReport (<i>String, String, String, String, String</i>)	<i>String</i>	param: PackageGUID [<i>String</i> - in] param: ExportPath [<i>String</i> - in] param: ImageFormat [<i>String</i> - in] param: Style [<i>String</i> - in] param: Extension [<i>String</i> - in] Run a HTML report (same as Documentation HTML Documentation on right click of Package in the Project Browser).
ImportPackageXMI (<i>String, String, Long, Long</i>)	<i>String</i>	param: PackageGUID [<i>String</i> - in] PackageGUID is the filename to import into (or overwrite). param: Filename or XMLText [<i>String</i> - in] The name of the XMI file. Note: If the <i>String</i> is of type <i>filename</i> it will be interpreted as a source file, otherwise the <i>String</i> will be imported as XML text. param: ImportDiagrams [<i>Long</i> - in] param: StripGUID [<i>Long</i> - in] Boolean value to indicate whether you want to replace the element UniqueID's on import. If stripped, then a package could be imported twice into EA - as two different versions. Provides the ability to import an XMI file at a point in the tree.

SaveControlledPackage (<i>String</i>)	<i>String</i>	<p>param: PackageGUID [String - in]</p> <p>Saves a package that has been configured as a controlled package - to XMI. The package GUID is only required, EA picks the rest up from the package control info.</p>
LoadControlledPackage (<i>String</i>)	<i>String</i>	<p>param: PackageGUID [String - in]</p> <p>Loads a package that has been marked and configured as controlled. The filename details & etc. are stored in the package control data.</p>
LayoutDiagram (<i>String, Long</i>) (Deprecated)	<i>Boolean</i>	<p>param: DiagramGUID [String - in] param: LayoutStyle [Long - in] this parameter is always ignored - it is recommended that LayoutDiagramEx is used instead</p> <p>Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for class and object diagrams.</p>
LayoutDiagramEx (<i>String, Long, Long, Long, Long, Boolean</i>)	<i>Boolean</i>	<p>param: DiagramGUID [String - in] param: LayoutStyle [Long - in] param: Iterations [Long - in] The number of layout iterations the Layout process should take to perform cross reduction (Default value = 4). param: LayerSpacing [Long - in] The per-element layer spacing the Layout process shall use (Default value = 20). param: ColumnSpacing [Long - in] The per-element column spacing the Layout process shall use (Default value = 20). param: SaveToDiagram [Boolean - in] Specifies whether or not EA should save the supplied layout options as default to the diagram in question.</p> <p>Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for class and object diagrams. LayoutStyle accepts the following options (refer to Layout a Diagram for further info):</p> <ul style="list-style-type: none"> •Default Options: IsDiagramDefault IsProgramDefault •Cycle Removal Options: IsCycleRemoveGreedy IsCycleRemoveDFS • Layering Options: IsLayeringLongestPathSink IsLayeringLongestPathSource IsLayeringOptimalLinkLength • Initialize Options: IsInitializeNaive IsInitializeDFSOut IsInitializeDFSIn

GenerateXSD(<i>String, String, String, String</i>)	<i>Boolean</i>	Create a XML schema for this GenerateXSD. Returns True on success. Parameters: PackageGUID: String, Identifies the package Filename: String, Target filepath Encoding: String, The XML encoding for the code page instruction. Options: String, Unused.
--	----------------	--

12.1.2.10 Code Samples

This section contains various code examples indicating how to use the Automation Interface, written in VB Dot Net:

- [Open the Repository](#)
- [Iterate Through an EAP File](#)
- [Add and Manage Packages](#)
- [Add and Manage Elements](#)
- [Add a Connector](#)
- [Add and Manage Diagrams](#)
- [Adding and Deleting Attributes and Methods](#)
- [Element Extras](#)
- [RepositoryExtras](#)
- [Stereotypes](#)
- [Working with Attributes](#)
- [Working with Methods](#)

12.1.2.10.1 Open the Repository

public Object

```
'an example of how to open an EA 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
```

12.1.2.10.2 Iterate Through an EAP File

public Object

```
'assume repository has already been opened

''start at the model level
Sub DumpModel()
  Dim idx as Integer
  For idx=0 to m_Repository.Models.Count-1
    DumpPackage(" ",m_Repository.Models.GetAt(idx))
  Next
End Sub

'output package name, then element contents, then process child packages
Sub DumpPackage(Indent as String, Package as Object)
  Dim idx as Integer
  Console.WriteLine(Indent + Package.Name)
  DumpElements(Indent + "  ", Package)

  For idx = 0 to Package.Packages.Count-1
    DumpPackage(Indent + "    ", Package.Packages.GetAt(idx))
  Next
End Sub

''dump element name
Sub DumpElements(Indent as String, Package as Object)
  Dim idx as Integer
  For idx = 0 to Package.Elements.Count-1
    Console.WriteLine(Indent + "::" + Package.Elements.GetAt(idx).Name)
  Next
End Sub
```

12.1.2.10.3 Add and Manage Packages

public Object

Example illustrating how to add a Model or a Package

```
Sub TestPackageLifecycle

  Dim idx as integer
  Dim idx2 as integer
  Dim package as object
  Dim model as object
  Dim o as object

  ''first add a new Model
  model = m_Repository.Models.AddNew("AdvancedModel","")
  If not model.Update() Then
    Console.WriteLine(model.GetLastError())
  End If

  ''refresh the models collection
  m_Repository.Models.Refresh

  ''now work through models collection and add a package

  For idx = 0 to m_Repository.Models.Count -1
    o = m_Repository.Models.GetAt(idx)
    Console.WriteLine(o.Name)
    If o.Name = "AdvancedModel" Then
      package = o.Packages.Addnew("Subpackage", "Nothing")
      If not package.Update() Then
        Console.WriteLine(package.GetLastError())
      End If

      package.Element.Stereotype = "system"
      package.Update
    End If
  Next
End Sub
```

```

        'for testing purposes just delete the
        'newly created Model and its contents
        m_Repository.Models.Delete(idx)

    End If
Next
End Sub

```

12.1.2.10.4 Add and Manage Elements

public Object

```

'Add and delete elements in a package

Sub ElementLifecycle

    Dim package as Object
    Dim element as Object

    package = m_Repository.GetPackageByID(2)
    element = package.elements.AddNew("Login to Website","UseCase")
    element.Stereotype = "testcase"
    element.Update
    package.elements.Refresh()

    Dim idx as integer

    'note the repeated calls to "package.elements.GetAt"
    'in general you should make this call once and assign to a local
    'variable - in the example below, EA will load the element required
    'everytime a call is made - rather than loading once and keeping
    'a local reference

    For idx = 0 to package.elements.count-1
        Console.WriteLine(package.elements.GetAt(idx).Name)
        If (package.elements.GetAt(idx).Name = "Login to Website" and _
            package.elements.GetAt(idx).Type = "UseCase") Then
            package.elements.deleteat(idx, false)
        End If
    Next
End Sub

```

12.1.2.10.5 Add a Connector

public Object

```

"Add a connector and set values

Sub ConnectorTest

    Dim source as object
    Dim target as object
    Dim con as object
    Dim o as object

    Dim client as object
    Dim supplier as object

    'use ElementID's to quickly load an element in this example
    '... you will need to find suitable ID's in your model

    source = m_Repository.GetElementByID(129)
    target = m_Repository.GetElementByID(169)

    con = source.Connectors.AddNew ("test link 2", "Association")

```

```
'again- replace ID with a suitable one from your model
con.SupplierID = 169

If not con.Update Then
    Console.WriteLine(con.GetLastError)
End If
source.Connectors.Refresh

Console.WriteLine("Connector Created")

o = con.Constraints.AddNew ("constraint2","type")
If not o.Update Then
    Console.WriteLine(o.GetLastError)
End If

o = con.TaggedValues.AddNew ("Tag","Value")
If not o.Update Then
    Console.WriteLine(o.GetLastError)
End If

''use the client and supplier ends to set
''additional information

client = con.ClientEnd
client.Visibility = "Private"
client.Role = "m_client"
client.Update
supplier = con.SupplierEnd
supplier.Visibility = "Protected"
supplier.Role = "m_supplier"
supplier.Update

Console.WriteLine("Client and Supplier set")

Console.WriteLine(client.Role)
Console.WriteLine(supplier.Role)

End Sub
```

12.1.2.10.6 Add and Manage Diagrams

public Object

```
''an example of how to create a diagram and add an element to it
''note the optional use of element rectangle setting using
''left,right,top and bottom dimensions in AddNew call

Sub DiagramLifeCycle

    Dim diagram as object
    Dim v as object
    Dim o as object
    Dim package as object

    Dim idx as Integer
    Dim idx2 as integer

    package = m_Repository.GetPackageByID(5)

    diagram = package.Diagrams.AddNew("Logical Diagram","Logical")
    If not diagram.Update Then
        Console.WriteLine(diagram.GetLastError)
    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

```

12.1.2.10.7 Adding and Deleting Attributes and Methods

public Object

```

Dim element as object
Dim idx as integer
Dim attribute as object
Dim method as object

'just load an element by ID - you will need to
'substitute a valid ID from your model
element = m_Repository.GetElementByID(246)

'create a new method
method = element.Methods.AddNew("newMethod", "int")
method.Update
element.Methods.Refresh

'now loop through methods for Element - and delete our addition
For idx = 0 to element.Methods.Count-1
    method =element.Methods.GetAt(idx)
    Console.WriteLine(method.Name)
    If(method.Name = "newMethod") Then
        element.Methods.Delete(idx)
    End if
Next

'create an attribute
attribute = element.attributes.AddNew("NewAttribute", "int")
attribute.Update
element.attributes.Refresh

'loop through and delete our new attribute
For idx = 0 to element.attributes.Count-1
    attribute =element.attributes.GetAt(idx)
    Console.WriteLine(attribute.Name)
    If(attribute.Name = "NewAttribute") Then
        element.attributes.Delete(idx)
    End If
Next

```

12.1.2.10.8 Element Extras

public Object

```

''examples of how to access and use element extras - such as
''scenarios, constraints and requirements

Sub ElementExtras

    Dim element as object
    Dim o as object
    Dim idx as Integer
    Dim bDel as boolean
    bDel = true

    try
        element = m_Repository.GetElementByID(129)
    
```



```
'manage constraints for an element
'demonstrate addnew and delete
o = element.Constraints.AddNew("Appended", "Type")
If not o.Update Then
    Console.WriteLine("Constraint error:" + o.GetLastError())
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
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
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
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
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
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
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

catch e as exception
    Console.WriteLine(element.Methods.GetLastError())
    Console.WriteLine(e)
End try

```

```
End Sub
```

12.1.2.10.9 RepositoryExtras

public Object

```
'' Examples of how to access repository
'' collections for system level information

Sub RepositoryExtras

    Dim o as object
    Dim idx as integer

    'issues
    o = m_Repository.Issues.AddNew("Problem","Type")
    If(o.Update=false) Then
        Console.WriteLine(o.GetLastError())
    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
End Sub
```

```

        Console.WriteLine("Delete Authors")
    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)
        Console.WriteLine("Delete Clients")
    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)
        Console.WriteLine("Delete Resources")
    End if
Next

End Sub

```

12.1.2.10.10 Stereotypes

public Object

```

Sub TestStereotypes

    Dim o as object
    Dim idx as integer

    ''add a new stereotype to the Stereotypes collection
    o = m_Repository.Stereotypes.AddNew("funky","class")
    If(o.Update=false) Then
        Console.WriteLine(o.GetLastError())
    End if
    o = nothing

    ''make sure we refresh
    m_Repository.Stereotypes.Refresh

    ''then iterate through - deleting our new entry in the process
    For idx = 0 to m_Repository.Stereotypes.Count-1
        Console.WriteLine(m_Repository.Stereotypes.GetAt(idx).Name)
        If(m_Repository.Stereotypes.GetAt(idx).Name = "funky") then
            m_Repository.Stereotypes.DeleteAt(idx,false)
            Console.WriteLine("Delete element")
        End if
    Next

End Sub

```

12.1.2.10.11 Working with Attributes

public Object

```
'an example of working with attributes

Sub AttributeLifecycle

    Dim element as object
    Dim o as object
    Dim t as object
    Dim idx as Integer
    Dim idx2 as integer
    try
        element = m_Repository.GetElementByID(129)

        For idx = 0 to element.Attributes.Count -1

            Console.WriteLine("attribute=" + element.Attributes.GetAt(idx).Name)

            o = element.Attributes.GetAt(idx)
            t = o.Constraints.AddNew("> 123", "Precision")
            t.Update()
            o.Constraints.Refresh
            For idx2 = 0 to o.Constraints.Count-1
                t = o.Constraints.GetAt(idx2)
                Console.WriteLine("Constraint: " + t.Name)
                If(t.Name="> 123") Then
                    o.Constraints.DeleteAt(idx2, false)
                End if
            Next

            For idx2 = 0 to o.TaggedValues.Count-1
                t = o.TaggedValues.GetAt(idx2)
                If(t.Name = "Type2") Then
                    'Console.WriteLine("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.Attributes.GetAt(idx).Name = "m_Tootle") Then
                Console.WriteLine("delete attribute")
                element.Attributes.DeleteAt(idx, false)
            End If

        Next

    catch e as exception
        Console.WriteLine(element.Attributes.GetLastError())
        Console.WriteLine(e)
    End try
End Sub
```

12.1.2.10.12 Working with Methods

public Object

```
'example of working with the Methods collection
'of an element - and with Method collections

Sub MethodLifeCycle
```

```
Dim element as object
Dim method as object
Dim t as object
Dim idx as Integer
Dim idx2 as integer

try
    element = m_Repository.GetElementByID(129)

    For idx = 0 to element.Methods.Count -1
        method = element.Methods.GetAt(idx)
        Console.WriteLine(method.Name)

        t = method.PreConditions.AddNew("TestConstraint","something")
        If t.Update = false Then
            Console.WriteLine("PreConditions: " + t.GetLastError)
        End if

        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

12.2 The Project Interface (XML)

Windows ActiveX Automation provides a way for applications to control and interface with Enterprise Architect.

Typically you use an automation capable client - such as Visual Basic or MS Word, and write a suitable program to open an Enterprise Architect model and iterate through the model contents. In this way you can write custom reports, code generators and other applications which draw on the information in an Enterprise Architect project.

This section will describe the interfaces - or methods - you will use to script Enterprise Architect.

See:

- [The Read Only Interface](#)
- [Automation Interfaces](#)
- [Opening a Project](#)
- [Enumerating Views](#)
- [Enumerating](#)
- [Enumerating Diagrams](#)
- [Get Diagrams](#)
- [Enumerating Elements](#)
- [Get Element](#)
- [Enumerating Links](#)
- [Get Link](#)
- [Get Diagram Image](#)

12.2.1 The Read Only Interface

This section describes the read only interface to EA. Using this you can retrieve any information required from the model using clients such as Visual Basic.

12.2.2 Automation Interfaces

An interface is a scripting method supported by Enterprise Architect that you can use to retrieve information about a model. To use these interfaces, you need an automation client, such as Visual Basic or MS Word.

The *Automation Interface* to read information from Enterprise Architect is defined as follows:

Project Interfaces

```
BOOL LoadProject(Const Variant FAR& FileName);
BOOL ReloadProject();
void ShowWindow(Long Show);
void Exit();
```

List Interfaces

```
BSTR EnumPackages(Const Variant FAR& PackageGUID);
BSTR EnumElements(Const Variant FAR& PackageGUID);
BSTR EnumLinks(Const Variant FAR& PackageID);
BSTR EnumDiagrams(Const Variant FAR& PackageGUID);
BSTR EnumDiagramElements(Const Variant FAR& DiagramGUID);
BSTR EnumDiagramLinks(Const Variant FAR& DiagramID);
BSTR EnumViews();
```

Diagram Interfaces

```

BOOL LoadDiagram(Const Variant FAR& DiagramGUID);
BSTR SaveDiagramImageToFile(Const Variant FAR& FileName);
BOOL PutDiagramImageOnClipboard(Const Variant FAR& DiagramGUID, Long Type);
BOOL PutDiagramImageToFile(Const Variant FAR& DiagramGUID, Const Variant FAR& Filename,
Long Type);

```

Retrieval Interfaces

```

BSTR GetElement(Const Variant FAR& ElementGUID);
BSTR GetLink(Const Variant FAR& LinkGUID);
BSTR GetDiagram(Const Variant FAR& DiagramGUID);
BSTR GetElementConstraints(Const Variant FAR& ElementGUID);
BSTR GetElementEffort(Const Variant FAR& ElementGUID);
BSTR GetElementMetrics(Const Variant FAR& ElementGUID);
BSTR GetElementFiles(Const Variant FAR& ElementGUID);
BSTR GetElementRequirements(Const Variant FAR& ElementGUID);
BSTR GetElementProblems(Const Variant FAR& ElementGUID);
BSTR GetElementResources(Const Variant FAR& ElementGUID);
BSTR GetElementRisks(Const Variant FAR& ElementGUID);
BSTR GetElementScenarios(Const Variant FAR& ElementGUID);
BSTR GetElementTests(Const Variant FAR& ElementGUID);

```

12.2.3 Opening a Project

The first task when using automation to open work with a model is to create the automation server.

When you run EA normally, it registers itself as an automation server and makes entries into the Windows registry for other programs to find it. The following code fragment from Visual Basic shows how to create an EA automation server: Once loaded you can iterate through the model using the EnumXXX functions described later.

Visual Basic Example Code to Open an EA Model

```

Option Explicit
'-----
'This model demonstrates how to invoke and read from
'an EA project using Automation scripting
'The scripting is based on XMI 1.1 compliant XML
'with the addition of enumeration calls (also XML based)

'the project object
Private EAPProject As Object

Private Sub CMDLoad_Click()

    'create the one and only EAPProject
    Set EAPProject = CreateObject("EA.Project")

    'load up with sepcified path
    EAPProject.LoadProject ("C:\Projects\AutomationExample.EAP")

    'optionally hide window
    'EAPProject.ShowWindow (0)

End Sub

```


12.2.4 Enumerating Views

The root packages of an Enterprise Architect project are its views. These include the Use Case View, the Logical View and the Physical View - as well as custom views created by analysts during modeling.

To work with a model, the usual starting point is to get a list of the defined views - depending on what is required, you can then iterate through all sub-packages for each or only specialised views.

EA provides the interface BSTR EnumViews() for retrieving a list of views. The returned XML includes a node for each node- detailing the view name and its universal identifier (GUID). You can use the GUID to retrieve the contents of any view.

Example XML from EnumViews

```
<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <View>
    <Name>Use Case View</Name>
    <GUID>EAID_DAF38564_68F1_4929_8760_8DDF15614F77</GUID>
  </View>
  <View>
    <Name>Dynamic View</Name>
    <GUID>EAID_DE77A45B_726E_468f_83AB_86C0391CC126</GUID>
  </View>
  <View>
    <Name>Logical View</Name>
    <GUID>EAID_A634A9A5_5589_44e9_B9D6_D1D4BC5EC544</GUID>
  </View>
  <View>
    <Name>Component View</Name>
    <GUID>EAID_3BA53453_8B77_4798_A0B5_DC02671804B7</GUID>
  </View>
  <View>
    <Name>Deployment View</Name>
    <GUID>EAID_A7E4B27B_DB98_4ef8_AE44_D58F389146F0</GUID>
  </View>
  <View>
    <Name>Custom</Name>
    <GUID>EAID_BE0B8A1C_2F09_4b8a_9FA3_015EA8E66056</GUID>
  </View>
</Document>
```

Sample Visual Basic Code to Enumerate through Views

```
Private Sub CMDGetViews_Click()

  ' Demonstrates how to iterate through the top level
  ' view of an Enterprise Architect project and
  ' access information on packages and elements contained therein
  ' All access is XML based, so create some documents and nodes

  Dim xmlDoc As New MSXML2.DOMDocument
  Dim xmlPkg As New MSXML2.DOMDocument
  Dim xmlView As MSXML2.IXMLDOMNode
  Dim xmlPkgNode As MSXML2.IXMLDOMNode

  List1.Clear
  List2.Clear

  'first get the View list
  xmlDoc.loadXML EAProject.EnumViews

  ' Debug.Print xmlDoc.xml

  ' While there are views, iterate through and retrieve sub information
  Set xmlView = xmlDoc.selectSingleNode("Document/View")
  Do While (Not xmlView Is Nothing)
    AddToTreeList xmlView.selectSingleNode("Name").Text
  Loop
End Sub
```

```

'load the list of packages for this view
xmlPkg.loadXML EAProject.EnumPackages(xmlView.selectSingleNode("GUID").Text)
Debug.Print xmlPkg.xml

Set xmlPkgNode = xmlPkg.selectSingleNode("Document/Package")

'while there are packages to process
Do While (Not xmlPkgNode Is Nothing)
  AddToTreeList " " & xmlPkgNode.selectSingleNode("Name").Text
  GetPackage xmlPkgNode.selectSingleNode("GUID").Text, Indent
  'if this is class model go a bit deeper
  'If (xmlPkgNode.selectSingleNode('Name').Text = 'Class Model') Then
  '  GetElements xmlPkgNode.selectSingleNode('GUID').Text
  'End If
  Set xmlPkgNode = xmlPkgNode.nextSibling
Loop

Set xmlView = xmlView.nextSibling
Loop

'finished
AddToDebugList "Complete"

End Sub

```

12.2.5 Enumerating

Once you have obtained a list of views, you can recursively iterate through the packages within each view using the *EnumPackages* interface to obtain a list of child packages for each parent package/view.

EA provides the *EnumPackages*(const VARIANT FAR& PackageGUID) interface to get an XML representation of all child packages of another package/view. The XML contains a node for each package - and each node contains a Name and a GUID element. The GUID element is used to get more information on a package (including listing packages within a package).

The following is an example of the XML returned by this interface:

```

<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Package>
    <Name>UC01: Use Case Model</Name>
    <GUID>EAID_508A37AC_5B27_4b98_8BBE_81D6AA03E8B4</GUID>
  </Package>
</Document>

```

The following Visual Basic code illustrates how to access and use the EnumPackages interface:

```

Private Sub GetPackage(PackageGUID As String, Offset As String)

  Dim xmlDoc As New MSXML2.DOMDocument
  Dim xmlPkg As New MSXML2.DOMDocument
  Dim xmlView As MSXML2.IXMLDOMNode
  Dim xmlPkgNode As MSXML2.IXMLDOMNode

  'list all diagrams in this package
  GetDiagrams PackageGUID, Offset

  'display all elements in this package
  GetElements PackageGUID, Offset

  'then load the list of packages for this package
  xmlPkg.loadXML EAProject.EnumPackages(PackageGUID)
  Set xmlPkgNode = xmlPkg.selectSingleNode("Document/Package")

```

```

'while there are packages to process
Do While (Not xmlPkgNode Is Nothing)
  AddToTreeList " " & xmlPkgNode.selectSingleNode("Name").Text
  GetPackage xmlPkgNode.selectSingleNode("GUID").Text, Indent + Offset
  Set xmlPkgNode = xmlPkgNode.nextSibling
Loop
End Sub

```

12.2.6 Enumerating Diagrams

You can retrieve a list of all diagrams within a package using the EnumDiagrams interface. This interface will return an XML list of all diagrams - including the name, type and universal identifier (GUID). You can use the GUID to retrieve more information about the diagram - and to either copy the diagram image to file or clipboard.

The EnumDiagrams interface is defined as *EnumDiagrams(const VARIANT FAR& PackageGUID)* - it returns an XML list of diagrams within a package - including name, type and GUID.

Example of XML Returned by Interface

```

<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Diagram>
    <Name>Overview</Name>
    <Type>Use Case</Type>
    <GUID>EAID_DA388E78_CF6B_4331_932A_565F1AADE18A</GUID>
  </Diagram>
</Document>

```

Example Visual Basic code which uses the EnumDiagrams interface

```

Private Sub GetDiagrams(PackageGUID As String, Offset As String)

  Dim xmlNode As MSXML2.IXMLDOMNode
  Dim xmlDoc As New MSXML2.DOMDocument
  Dim str As String

  'get list of elements in package
  xmlDoc.loadXML EAProject.EnumDiagrams(PackageGUID)

  ' Debug.Print xmlDoc.xml

  Set xmlNode = xmlDoc.selectSingleNode("Document/Diagram")
  Do While (Not xmlNode Is Nothing)
    'add to list
    str = xmlNode.selectSingleNode("Name").Text + ":" +
xmlNode.selectSingleNode("Type").Text
    AddToTreeList Offset + Indent + "Diagram - " + str
    GetDiagram xmlNode.selectSingleNode("GUID").Text

    'go to next diagram
    Set xmlNode = xmlNode.nextSibling
  Loop
End Sub

```

12.2.7 Get Diagrams

Use this interface to retrieve information about a diagram - including, name, type, characteristics and a list of elements within the diagram. This interface requires you to pass in a diagram identifier (GUID) - typically retrieved from the EnumDiagrams interface, and returns an XML document detailing diagram attributes.

The interface is defined as GetDiagram(const VARIANT FAR& DiagramGUID)

An example of the XML returned by this interface is shown below:

```
<Document xmlns:UML="omg.org/UML1.3">
  <Diagram>
    <UML:Diagram name="Overview" xmi.id="EAID_DA388E78_CF6B_4331_932A_565F1AADE18A"
    diagramType="UseCaseDiagram" owner="owner" toolName="Enterprise Architect 2.5">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="documentation" value="This diagram provides an
        overview of the Customer's available activities in the proposed system - from logging
        on to browsing the book store, selecting items and making purchases." />
        <UML:TaggedValue tag="version" value="1.0" />
        <UML:TaggedValue tag="author" value="Geoffrey Sparks" />
        <UML:TaggedValue tag="created_date" value="6/30/2001" />
        <UML:TaggedValue tag="modified_date" value="6/30/2001" />
        <UML:TaggedValue tag="package"
        value="EAPK_508A37AC_5B27_4b98_8BBE_81D6AA03E8B4" />
        <UML:TaggedValue tag="documentation" value="This diagram provides an
        overview of the Customer's available activities in the proposed system - from logging
        on to browsing the book store, selecting items and making purchases." />
        <UML:TaggedValue tag="type" value="Use Case" />
        <UML:TaggedValue tag="EAStyle"
        value="ShowPrivate=1;ShowProtected=1;ShowPublic=1;HideRelationships=0;Locked=0;Border=
        1;HighlightForeign=1;PackageContents=1;SequenceNotes=0;ScalePrintImage=0;DocSize.cx=82
        6;DocSize.cy=1169;ShowDetails=0;Orientation=P;Zoom=100;ShowTags=1;OpParams=1;" />
      </UML:ModelElement.taggedValue>

      <UML:Diagram.element>
        <UML:DiagramElement geometry="Left=32;Top=49;Right=222;Bottom=125;"
        subject="EAID_942CA067_D717_433d_85DB_87A4A4B28660" seqno="1" />
        <UML:DiagramElement geometry="Left=124;Top=411;Right=367;Bottom=499;"
        subject="EAID_D4597C98_506F_421a_AC73_75E9B3619602" seqno="2" />
        <UML:DiagramElement geometry="Left=291;Top=207;Right=336;Bottom=297;"
        subject="EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2" seqno="3" />
        <UML:DiagramElement geometry="SX=0;SY=0;EX=0;EY=0;"
        subject="EAID_4E742B84_D86E_4e37_B91C_8D3E25299646" style="Mode=1;;Hidden=0;" />
        <UML:DiagramElement geometry="SX=0;SY=0;EX=0;EY=0;"
        subject="EAID_0D03CF69_61BF_4041_9CD8_540901F171F9" style="Mode=1;;Hidden=0;" />
      </UML:Diagram.element>
    </UML:Diagram>
  </Diagram>
</Document>
```

The following Visual Basic code illustrates the use of this interface

```
Private Sub GetDiagram(DiagramGUID As String)
On Error GoTo errDiagram

'get a diagram from the model - includes
'all attributes and tagged values

Dim xmlNode As MSXML2.IXMLDOMNode
Dim xmlDiagram As New MSXML2.DOMDocument
Dim xmlDoc As New MSXML2.DOMDocument
Dim xmlTagged As MSXML2.IXMLDOMNode
Dim xmlElement As MSXML2.IXMLDOMNode

Dim n As Integer

EAPProject.PutDiagramImageOnClipboard DiagramGUID, 0

xmlDiagram.loadXML EAPProject.GetDiagram(DiagramGUID)
```

```

' Debug.Print xmlDiagram.xml

Set xmlNode = xmlDiagram.selectSingleNode("Document/Diagram")

'go to first element - will be the actual diagram (eg. UML:StateChart)
Set xmlNode = xmlNode.firstChild()

AddToDebugList "-----"
AddToDebugList xmlNode.nodeName

'get attributes first
For n = 0 To xmlNode.Attributes.length - 1
    AddToDebugList xmlNode.Attributes.item(n).nodeName + " =: " +
xmlNode.Attributes.item(n).Text
Next n

'then get tagged values
Set xmlTagged =
xmlNode.selectSingleNode("UML:ModelElement.taggedValue/UML:TaggedValue")
Do While (Not xmlTagged Is Nothing)
    AddToDebugList xmlTagged.Attributes(0).Text + " =: " +
xmlTagged.Attributes(1).Text
    Set xmlTagged = xmlTagged.nextSibling
Loop

'now get diagram elements
Set xmlElement = xmlNode.selectSingleNode("UML:Diagram.element/UML:DiagramElement")
Do While (Not xmlElement Is Nothing)
    'get attributes first
    AddToDebugList "... Diagram Element ..."
    For n = 0 To xmlElement.Attributes.length - 1
        AddToDebugList xmlElement.Attributes.item(n).nodeName + " =: " +
xmlElement.Attributes.item(n).Text
    Next n
    Set xmlElement = xmlElement.nextSibling
Loop

Exit Sub

errDiagram:
AddToDebugList "ERROR: " + Error$

End Sub

```

12.2.8 Enumerating Elements

Once you have a package GUID, you can pass it into the EnumElements interface to get a list of elements within a package. An element is any kind of UML object - such as a Class or Object or Activity. The EnumElements interface returns XML that lists the name, type and GUID for each contained element. From this you can use the element GUID to obtain more information about each element.

The interface is defined as *EnumElements(const VARIANT FAR& PackageGUID)* - and returns an XML string.

Example XML returned by EnumElements

```

<Document xmlns:UML="omg.org/UML1.3">
  <Element>
    <Name>Customer</Name>
    <Type>Actor</Type>
    <GUID>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</GUID>
  </Element>
  <Element>
    <Name>Note</Name>
    <Type>Note</Type>
    <GUID>EAID_D4597C98_506F_421a_AC73_75E9B3619602</GUID>
  </Element>

```

```

    <Element>
      <Name>UC01-1: User Management</Name>
      <Type>Package</Type>
      <GUID>EAID_942CA067_D717_433d_85DB_87A4A4B28660</GUID>
    </Element>
  </Document>

```

Example Visual Basic code used to enumerate elements

```

Private Sub GetElements(PackageGUID As String, Offset As String)

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xmlDoc As New MSXML2.DOMDocument
    Dim str As String

    'get list of elements in package
    xmlDoc.loadXML EAProject.EnumElements(PackageGUID)

    ' Debug.Print xmlDoc.xml

    Set xmlNode = xmlDoc.selectSingleNode("Document/Element")
    Do While (Not xmlNode Is Nothing)
        'add to list
        str = xmlNode.selectSingleNode("Name").Text + ":" +
xmlNode.selectSingleNode("Type").Text
        AddToTreeList Offset + Indent + str

        GetElement xmlNode.selectSingleNode("GUID").Text
        'get details
        GetLinks xmlNode.selectSingleNode("GUID").Text, Offset + Indent

        'go to next element
        Set xmlNode = xmlNode.nextSibling
    Loop

End Sub

```

12.2.9 Get Element

To retrieve information about an element in EA, use the `GetElement` interface. This takes an Element GUID previously obtained using one of the enumeration interfaces, and returns an XML document containing all element details. For class elements, this will also include Operation and Attribute details.

The interface is defined as `GetElement(const VARIANT FAR& ElementGUID)` - it takes an element GUID and returns an XML document.

An example XML element document is given below

```

<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
  <Element>
    <UML:Class name="Customer" xmi.id="EAID_D5663F5E_D116_4767_A13B_F9EE25BCFF9E"
visibility="public" namespace="EAPK_C14247A1_6C15_4b5a_9AFC_98A4A5611138" isRoot="false"
isLeaf="false" isAbstract="true">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="ea_stype" value="Class" />
        <UML:TaggedValue tag="ea_ntype" value="0" />
        <UML:TaggedValue tag="version" value="1.0" />
        <UML:TaggedValue tag="package"
value="EAPK_C14247A1_6C15_4b5a_9AFC_98A4A5611138" />
        <UML:TaggedValue tag="date_created" value="9/30/2000" />
        <UML:TaggedValue tag="date_modified" value="9/2/2001" />
        <UML:TaggedValue tag="genfile" value="C:\Documents and
Settings\Administrator\Desktop\Customer.cls" />
        <UML:TaggedValue tag="gentype" value="Visual Basic" />
        <UML:TaggedValue tag="tagged" value="0" />
      </UML:ModelElement.taggedValue>
    </UML:Class>
  </Element>
</Document>

```

```

    <UML:TaggedValue tag="package_name" value="LM01-1: Customer Domain" />
    <UML:TaggedValue tag="phase" value="1.0" />
    <UML:TaggedValue tag="author" value="Geoffrey Sparks" />
    <UML:TaggedValue tag="complexity" value="1" />
    <UML:TaggedValue tag="documentation" value="A customer class. Contains
attributes and behavior corresponding to a customer of the on-line bookstore. A customer
has a current account with the book store and preferred shipping methods." />
    <UML:TaggedValue tag="status" value="Approved" />
    <UML:TaggedValue tag="style" value="BackColor=-1;BorderColor=-1;BorderWidth=-
1;FontColor=-1;VSwimLanes=0;HSwimLanes=0;BorderStyle=0;" />
  </UML:ModelElement.taggedValue>

  <UML:Classifier.feature>
    <UML:Attribute name="Account" visibility="private" ownerScope="instance"
changeable="none" targetScope="instance">
      <UML:Attribute.initialValue>
        <UML:Expression />
      </UML:Attribute.initialValue>
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="CustomerAccount" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="containment" value="Not Specified" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="description" value="The customer account object"
/>

        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="genoption" value="PROPERTY=Account;" />
        <UML:TaggedValue tag="scope" value="Private" />
      </UML:ModelElement.taggedValue>

      <UML:ModelElement.constraint>
        <UML:Constraint name="Not null">
          <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="Pre-Condition" />
            <UML:TaggedValue tag="documentation" value="Customer account
cannot be null" />
          </UML:ModelElement.taggedValue>
        </UML:Constraint>
      </UML:ModelElement.constraint>
    </UML:Attribute>

    <UML:Attribute name="Address" visibility="private" ownerScope="instance"
changeable="none" targetScope="instance">
      <UML:Attribute.initialValue>
        <UML:Expression />
      </UML:Attribute.initialValue>
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="containment" value="Not Specified" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="1" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="genoption" value="PROPERTY=DED;" />
        <UML:TaggedValue tag="scope" value="Private" />
      </UML:ModelElement.taggedValue>
    </UML:Attribute>

    <UML:Attribute name="City" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
      <UML:Attribute.initialValue>
        <UML:Expression />
      </UML:Attribute.initialValue>

      <UML:ModelElement.taggedValue>

```

```

        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="containment" value="Not Specified" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="2" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>
</UML:Attribute>

    <UML:Attribute name="Country" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
    <UML:Attribute.initialValue>
        <UML:Expression />
    </UML:Attribute.initialValue>

    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="Country" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="3" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>
</UML:Attribute>

    <UML:Attribute name="CustomerID" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
    <UML:Attribute.initialValue>
        <UML:Expression />
    </UML:Attribute.initialValue>

    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="4" />
        <UML:TaggedValue tag="description" value="Unique identifier for a
customer. Used internally only." />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>
</UML:Attribute>

    <UML:Attribute name="FirstName" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
    <UML:Attribute.initialValue>
        <UML:Expression />
    </UML:Attribute.initialValue>

    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="5" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>

```



```

</UML:Attribute>

<UML:Attribute name="LastLogin" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="string" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="6" />
    <UML:TaggedValue tag="description" value="The last time this user
logged in. Display when user logs in" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

<UML:Attribute name="Login" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="string" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="7" />
    <UML:TaggedValue tag="description" value="The user login in" />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>
</UML:Attribute>

<UML:Attribute name="Password" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
  <UML:Attribute.initialValue>
    <UML:Expression />
  </UML:Attribute.initialValue>

  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="type" value="DateTime" />
    <UML:TaggedValue tag="derived" value="0" />
    <UML:TaggedValue tag="length" value="0" />
    <UML:TaggedValue tag="ordered" value="0" />
    <UML:TaggedValue tag="precision" value="0" />
    <UML:TaggedValue tag="scale" value="0" />
    <UML:TaggedValue tag="collection" value="false" />
    <UML:TaggedValue tag="position" value="8" />
    <UML:TaggedValue tag="description" value="The user's password. " />
    <UML:TaggedValue tag="duplicates" value="0" />
    <UML:TaggedValue tag="scope" value="Public" />
  </UML:ModelElement.taggedValue>

  <UML:ModelElement.constraint>
    <UML:Constraint name="&gt;=8 Characters">
      <UML:ModelElement.taggedValue />
    </UML:Constraint>
  </UML:ModelElement.constraint>
</UML:Attribute>

<UML:Attribute name="Preferences" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
  <UML:Attribute.initialValue>

```

```

        <UML:Expression />
    </UML:Attribute.initialValue>

    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="CustomerPreferences" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="9" />
        <UML:TaggedValue tag="description" value="A Customer preferences
object" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>
</UML:Attribute>

    <UML:Attribute name="Surname" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
    <UML:Attribute.initialValue>
        <UML:Expression />
    </UML:Attribute.initialValue>

    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="10" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>
</UML:Attribute>

    <UML:Attribute name="Zip" visibility="public" ownerScope="instance"
changeable="none" targetScope="instance">
    <UML:Attribute.initialValue>
        <UML:Expression />
    </UML:Attribute.initialValue>

    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="Zip" />
        <UML:TaggedValue tag="derived" value="0" />
        <UML:TaggedValue tag="length" value="0" />
        <UML:TaggedValue tag="ordered" value="0" />
        <UML:TaggedValue tag="precision" value="0" />
        <UML:TaggedValue tag="scale" value="0" />
        <UML:TaggedValue tag="collection" value="false" />
        <UML:TaggedValue tag="position" value="11" />
        <UML:TaggedValue tag="duplicates" value="0" />
        <UML:TaggedValue tag="scope" value="Public" />
    </UML:ModelElement.taggedValue>
</UML:Attribute>

    <UML:Operation name="Account" visibility="public" ownerScope="instance"
isQuery="false" concurrency="sequential">
    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="CustomerAccount" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="stereotype" value="property get" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>
</UML:Operation>

    <UML:Operation name="Account" visibility="public" ownerScope="instance"
isQuery="false" concurrency="sequential">
    <UML:ModelElement.taggedValue>

```

```

        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="stereotype" value="property let" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>

    <UML:BehavioralFeature.parameter>
        <UML:Parameter name="NewVal" visibility="public">
            <UML:ModelElement.taggedValue>
                <UML:TaggedValue tag="pos" value="0" />
                <UML:TaggedValue tag="type" value="CustomerAccount" />
                <UML:TaggedValue tag="const" value="0" />
            </UML:ModelElement.taggedValue>

            <UML:Parameter.defaultValue>
                <UML:Expression />
            </UML:Parameter.defaultValue>
        </UML:Parameter>
    </UML:BehavioralFeature.parameter>
</UML:Operation>

    <UML:Operation name="AddCustomer" visibility="public" ownerScope="instance"
isQuery="false" concurrency="sequential">
    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="bool" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>

    <UML:BehavioralFeature.parameter>
        <UML:Parameter name="Name" visibility="public">
            <UML:ModelElement.taggedValue>
                <UML:TaggedValue tag="pos" value="1" />
                <UML:TaggedValue tag="type" value="String" />
                <UML:TaggedValue tag="const" value="0" />
            </UML:ModelElement.taggedValue>

            <UML:Parameter.defaultValue>
                <UML:Expression />
            </UML:Parameter.defaultValue>
        </UML:Parameter>

        <UML:Parameter name="AccountID" kind="in" visibility="public">
            <UML:ModelElement.taggedValue>
                <UML:TaggedValue tag="pos" value="0" />
                <UML:TaggedValue tag="note" value="The default customer account
ID" />

                <UML:TaggedValue tag="type" value="string" />
                <UML:TaggedValue tag="const" value="0" />
            </UML:ModelElement.taggedValue>

            <UML:Parameter.defaultValue>
                <UML:Expression />
            </UML:Parameter.defaultValue>
        </UML:Parameter>
    </UML:BehavioralFeature.parameter>
</UML:Operation>

    <UML:Operation name="DeleteCustomer" visibility="public"
ownerScope="instance" isQuery="false" concurrency="sequential">
    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="bool" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />

```

```

        <UML:TaggedValue tag="pure" value="0" />
      </UML:ModelElement.taggedValue>
    </UML:Operation>

    <UML:Operation name="GetAccount" visibility="public" ownerScope="instance"
isQuery="false" concurrency="sequential">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="CustomerAccount" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
      </UML:ModelElement.taggedValue>

      <UML:BehavioralFeature.parameter>
        <UML:Parameter name="dsd" visibility="public">
          <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="pos" value="0" />
            <UML:TaggedValue tag="type" value="Functional" />
            <UML:TaggedValue tag="const" value="0" />
          </UML:ModelElement.taggedValue>

          <UML:Parameter.defaultValue>
            <UML:Expression body="sd" />
          </UML:Parameter.defaultValue>
        </UML:Parameter>
      </UML:BehavioralFeature.parameter>
    </UML:Operation>

    <UML:Operation name="GetCustomerAsXML" visibility="public"
ownerScope="instance" isQuery="false" concurrency="sequential">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="string" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
      </UML:ModelElement.taggedValue>
    </UML:Operation>

    <UML:Operation name="GetPreferences" visibility="public"
ownerScope="instance" isQuery="false" concurrency="sequential">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="CustomerPreferences" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
      </UML:ModelElement.taggedValue>
    </UML:Operation>

    <UML:Operation name="UpdateCustomer" visibility="public"
ownerScope="instance" isQuery="false" concurrency="sequential">
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="bool" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
      </UML:ModelElement.taggedValue>

      <UML:BehavioralFeature.parameter>
        <UML:Parameter name="Fred" visibility="public">

```

```

        <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="pos" value="0" />
            <UML:TaggedValue tag="type" value="xml" />
            <UML:TaggedValue tag="const" value="0" />
        </UML:ModelElement.taggedValue>
        <UML:Parameter.defaultValue>
            <UML:Expression />
        </UML:Parameter.defaultValue>
    </UML:Parameter>
</UML:BehavioralFeature.parameter>
</UML:Operation>

    <UML:Operation name="ValidateCustomer" visibility="public"
ownerScope="instance" isQuery="false" concurrency="sequential">
    <UML:ModelElement.taggedValue>
        <UML:TaggedValue tag="type" value="void" />
        <UML:TaggedValue tag="const" value="false" />
        <UML:TaggedValue tag="static" value="1" />
        <UML:TaggedValue tag="synchronised" value="0" />
        <UML:TaggedValue tag="concurrency" value="Sequential" />
        <UML:TaggedValue tag="position" value="0" />
        <UML:TaggedValue tag="returnarray" value="0" />
        <UML:TaggedValue tag="pure" value="0" />
    </UML:ModelElement.taggedValue>
    </UML:Operation>
</UML:Classifier.feature>
</UML:Class>
</Element>
</Document>

```

An example of Visual Basic code using this interface is given below - note that it includes calls to get element extensions - such as tests and resources

```

Private Sub GetElement(ObjectGUID As String)
On Error GoTo errElement

    'get an element from the model - includes
    'all attributes and tagged value
    'followed by element extensions - such as constraints, tests etc.

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xmlElement As New MSXML2.DOMDocument
    Dim xmlDoc As New MSXML2.DOMDocument
    Dim xmlTagged As MSXML2.IXMLDOMNode
    Dim n As Integer

    xmlElement.loadXML EAPProject.GetElement(ObjectGUID)

    Set xmlNode = xmlElement.selectSingleNode("Document/Element")

    'go to first element - will be the actual element (eg. UML:Actor)
    Set xmlNode = xmlNode.firstChild()

    'If (xmlNode.nodeName = 'UML:Class') Then Debug.Print vbCrLf + xmlElement.xml +
    vbCrLf

    AddToDebugList "-----"
    AddToDebugList xmlNode.nodeName

    'get attributes first
    For n = 0 To xmlNode.Attributes.length - 1
        AddToDebugList xmlNode.Attributes.item(n).nodeName + " =: " +
xmlNode.Attributes.item(n).Text
    Next n

    'get operations and attributes
    GetFeatures xmlNode

    'then get tagged values
    Set xmlTagged =
xmlNode.selectSingleNode("UML:ModelElement.taggedValue/UML:TaggedValue")
    Do While (Not xmlTagged Is Nothing)
        AddToDebugList xmlTagged.Attributes(0).Text + " =: " +

```

```

xmlTagged.Attributes(1).Text
    Set xmlTagged = xmlTagged.nextSibling
    Loop

'now process extensions

xmlDoc.loadXML (EAPProject.GetElementScenarios(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.scenario", "EAScenario")

xmlDoc.loadXML (EAPProject.GetElementRequirements(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/UML:ModelElement.requirement",
"UML:Dependency")

    xmlDoc.loadXML (EAPProject.GetElementConstraints(ObjectGUID))
    Call GetExtension(xmlDoc, "Document/Element/UML:ModelElement.Constraint",
"UML:Constraint")

xmlDoc.loadXML (EAPProject.GetElementEffort(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.effort", "EAEffort")

xmlDoc.loadXML (EAPProject.GetElementMetrics(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.metric", "EAMetric")

xmlDoc.loadXML (EAPProject.GetElementFiles(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.file", "EAFile")

xmlDoc.loadXML (EAPProject.GetElementProblems(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.defect", "EADefect")

xmlDoc.loadXML (EAPProject.GetElementResources(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.resource", "EAResource")

xmlDoc.loadXML (EAPProject.GetElementRisks(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.risk", "EARisk")

xmlDoc.loadXML (EAPProject.GetElementTests(ObjectGUID))
Call GetExtension(xmlDoc, "Document/Element/EAModel.test", "EATest")

AddToDebugList "End " + xmlNode.nodeName
AddToDebugList "-----"
AddToDebugList " "

Exit Sub

errElement:
    AddToDebugList Error$

End Sub

```

12.2.10 Enumerating Links

Once you have an element, you can enumerate all the links for it. To do this, first locate your element using the EnumElements interface, then pass the element GUID to the EnumLinks interface.

The interface is defined as *EnumLinks(const VARIANT FAR& ElementID)* - and returns an XML list of all links associated with an element. You can pass the Link GUID to the GetLink interface to retrieve specific information about a link.

Example XML from the EnumLinks interface

```

<Document xmlns:UML="omg.org/UML1.3">
  <Link>
    <Type>NoteLink</Type>
    <LinkID>EAID_0D03CF69_61BF_4041_9CD8_540901F171F9</LinkID>
    <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
  </Link>
</Document>

```

```

        <OtherType>Actor</OtherType>
        <OtherIsTarget>0</OtherIsTarget>
    </Link>
    <Link>
        <Type>UseCase</Type>
        <LinkID>EAID_23D317E7_B50A_45a5_AEE8_081A26CEA18D</LinkID>
        <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
        <OtherType>Actor</OtherType>
        <OtherIsTarget>-1</OtherIsTarget>
    </Link>
    <Link>
        <Type>UseCase</Type>
        <LinkID>EAID_4E742B84_D86E_4e37_B91C_8D3E25299646</LinkID>
        <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
        <OtherType>Actor</OtherType>
        <OtherIsTarget>-1</OtherIsTarget>
    </Link>
    <Link>
        <Type>UseCase</Type>
        <LinkID>EAID_77FA191A_C775_40b0_96FC_A741D1083D88</LinkID>
        <Other>EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2</Other>
        <OtherType>Actor</OtherType>
        <OtherIsTarget>-1</OtherIsTarget>
    </Link>
</Document>

```

Example Visual Basic code used to enumerate links

```

Private Sub GetLinks(ObjectGUID As String, Offset As String)

    'enumerate through the list of links for an object
    'for each link, call the GetLink function to retrieve actual details

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xml As New MSXML2.DOMDocument
    Dim str As String

    xml.loadXML EAPProject.EnumLinks(ObjectGUID)

    ' Debug.Print xml.xml

    Set xmlNode = xml.selectSingleNode("Document/Link")

    Do While (Not xmlNode Is Nothing)
        AddToTreeList Offset + Indent + "link: " + xmlNode.selectSingleNode("Type").Text
        GetLink xmlNode.selectSingleNode("LinkID").Text
        Set xmlNode = xmlNode.nextSibling
    Loop

End Sub

```

12.2.11 Get Link

You may retrieve information about a link in Enterprise Architect using the GetLink interface. This takes a Link GUID obtained from the EnumLinks method and returns an XML document containing full details about a UML link.

The interface is defined as *GetLink(const VARIANT FAR& LinkGUID)* - and returns an XML document of a link.

XML example from a call to GetLink

```

<?xml version="1.0"?>
<Document xmlns:UML="omg.org/UML1.3">
    <Link>
        <UML:Association xmi.id="EAID_23D317E7_B50A_45a5_AEE8_081A26CEA18D"
visibility="public" isRoot="false" isLeaf="false" isAbstract="false">

```

```

    <UML:ModelElement.taggedValue>
      <UML:TaggedValue tag="style" value="1" />
      <UML:TaggedValue tag="ea_type" value="UseCase" />
      <UML:TaggedValue tag="linemode" value="1" />
      <UML:TaggedValue tag="seqno" value="0" />
      <UML:TaggedValue tag="documentation" value="A customer uses the login use
case to access the book store" />
    </UML:ModelElement.taggedValue>
    <UML:Association.connection>
      <UML:AssociationEnd visibility="public" aggregation="none"
isOrdered="false" targetScope="instance" isNavigable="true"
Type="EAID_2F73E1ED_B7E2_4ce4_8758_02223B6C17B2">
        <UML:ModelElement.taggedValue>
          <UML:TaggedValue tag="containment" value="Unspecified" />
        </UML:ModelElement.taggedValue>
      </UML:AssociationEnd>
      <UML:AssociationEnd visibility="public" aggregation="none"
isOrdered="false" targetScope="instance" isNavigable="true"
Type="EAID_B00C2EA6_3DD4_4e7b_886E_A8B7D2B17AEA">
        <UML:ModelElement.taggedValue>
          <UML:TaggedValue tag="containment" value="Unspecified" />
        </UML:ModelElement.taggedValue>
      </UML:AssociationEnd>
    </UML:Association.connection>
  </UML:Association>
</Link>
</Document>

```

A Visual Basic Example of retrieving Link information using a Link GUID

```

Private Sub GetLink(LinkGUID As String)
On Error GoTo errLink

    'get a single link details

    Dim xmlNode As MSXML2.IXMLDOMNode
    Dim xmlElement As New MSXML2.DOMDocument
    Dim xmlDoc As New MSXML2.DOMDocument
    Dim xmlTagged As MSXML2.IXMLDOMNode
    Dim n As Integer

    Dim str As String
    str = EAProject.GetLink(LinkGUID)
    Debug.Print str

    xmlElement.loadXML EAProject.GetLink(LinkGUID)

    Set xmlNode = xmlElement.selectSingleNode("Document/Link")

    If (xmlNode Is Nothing) Then
        Exit Sub
    End If

    Debug.Print xmlElement.xml

    'go to first element - will be the actual element (eg. UML:Actor)
    Set xmlNode = xmlNode.firstChild()

    AddToDebugList " "
    AddToDebugList xmlNode.nodeName

    'get attributes first
    For n = 0 To xmlNode.Attributes.length - 1
        AddToDebugList xmlNode.Attributes.item(n).nodeName + " =: " +
xmlNode.Attributes.item(n).Text
    Next n

    'then get tagged values
    Set xmlTagged =
xmlNode.selectSingleNode("UML:ModelElement.taggedValue/UML:TaggedValue")
    Do While (Not xmlTagged Is Nothing)
        AddToDebugList xmlTagged.Attributes(0).Text + " =: " +
xmlTagged.Attributes(1).Text
    Loop

```



```
        Set xmlTagged = xmlTagged.nextSibling
    Loop

    AddToDebugList " "

    Exit Sub

errLink:
    AddToDebugList Error$
End Sub
```

12.2.12 Get Diagram Image

You can retrieve a diagram image to either a file or onto the Windows clipboard for pasting into other applications.

In the first case you use the interface PutDiagramImageToFile(const VARIANT FAR& DiagramGUID, const VARIANT FAR& Filename, long Type) supplying the Diagram GUID, the output filename and the image type(metafile or bitmap = 0 or 1). If type = 0 then then file type will be metafile, If type = 1 then it will use the file type from the name extension (ie. .bmp, .jpg, .gif, .png, .tga).

In the second case you call PutDiagramImageOnClipboard(const VARIANT FAR& DiagramGUID, long Type) passing the GUID and the image type.

A Visual Basic example of calling this code is given below:

```
' Code to select a diagram GUID
'
'Call the get image interface to put image on clipboard

EAProject.PutDiagramImageOnClipboard DiagramGUID, 0
'
'
```

Once complete, this Call will result In the requested diagram image being placed onto the Windows clipboard.

12.3 Add-Ins

Introduction

Add-ins let you add functionality to Enterprise Architect. The Enterprise Architect Add-in model builds on the features provided by the [Automation Interface](#) to allow you to extend the Enterprise Architect user interface.

Add-ins are ActiveX COM objects that expose public Dispatch methods. They have several advantages over stand-alone automation clients:

- Add-ins may define EA menus and sub-menus.
- Add-ins receive notifications about various EA user-interface events including menu clicks and file changes.
- Add-ins can (and should) be written as in-process (DLL) components. This provides lower call overhead and better integration into the EA environment.
- Because a current version of EA is already running there is no need to start a second copy of EA via the automation interface.
- Because the add-in receives object handles associated with the currently running copy of EA, more information is available about the current user's activity eg. which diagram objects are selected.
- The user is not required to do anything other than to install the add-in to make it usable - ie. Users of add-

ins do not have to configure add-ins to run on their systems.

Creating and Using Add-Ins

This section of the help file covers the following information on Add-Ins:

- [Add-In Tasks](#)
- [Add-In Events](#)
- [Broadcast Events](#)
- [Custom Views](#)
- [MDG Add-Ins](#)

See also: [The Automation Interface](#)

12.3.1 Overview

The Enterprise Architect Add-in model builds on the features provided by the automation interface to allow Windows programmers to extend the Enterprise Architect user interface.

Add-Ins are ActiveX COM objects that expose public Dispatch methods that repond to EA.

Add-ins have several advantages over stand-alone automation clients:

- Add-ins may define EA menus and sub-menus.
- Add-ins receive notifications about various EA user-interface events including menu clicks and file changes.
- Add-ins can (and should) be written as in-process (DLL) components. This provides lower call overhead and better integration into the EA environment.
- Because a current version of EA is already running there is no need to start a second copy of EA via the automation interface.
- Because the add-in receives object handles associated with the currently running copy of EA, more information is available about the current user's activity eg. Which diagram objects are selected.
- The user is not required to do anything other than to install the add-in to make it usable - ie. Users of add-ins do not have to configure add-ins to run on their systems.

Because Enterprise Architect is constantly evolving in response to customer requests - the previous "hard-wired" add-in interface has been replaced by a more flexible one:

- The Addin interface no longer has its own version - rather it is identified by the version of EA it first appeared in. For example, the current version of the EA Add-in interface is version 2.1.
- There is no need to refer to a type-library when creating your add-in.
- Add-ins no longer need to implement methods that they never use.

Add-ins created using the previous mechanism will still run - though you will need to change your style of add-in to use new features.

Additional changes include:

- Add-ins now prompt users via context menus in the treeview and the diagram.
- Menu check and disable states can be controlled by the add-in.

12.3.2 Add-In tasks

This section provides instructions to the developer on how to perform the tasks associated with creating, testing and deploying add-ins.

1. [Create an Add-In](#)
 - [Define Menu Items](#)

- [Responding to Menu Events](#)
 - [Handling Add-In Events](#)
2. [Deploy your Add-In](#)
 3. [Potential Pitfalls](#)

12.3.2.1 Creating Add-Ins

Before you start you will require an application development tool that is capable of creating ActiveX COM objects supporting the IDispatch interface. These include:

- Borland Delphi
- Microsoft Visual Basic
- Microsoft Visual Studio .Net

Create an Add-In

An EA Add-In can be created in three steps:

1. Use a development tool to create a ActiveX COM DLL project. Visual Basic users, for example, choose File>Create New Project>ActiveX DLL.
2. Connect to the interface using the syntax appropriate to the language as detailed in the [Connecting to the Interface](#) topic.
3. Create a COM class and implement each of the [general Add-In Events](#) applicable to your Add-In. You only need to define methods for events you wish to respond to.
4. Add a registry key identifying your Add-In to EA as described in [Deploying Add-Ins](#).

See Also:

[Defining Menu Items](#)

[Examples of Automation Interfaces](#), this web page provides examples of code used to create Add-Ins for EA.

12.3.2.1.1 Defining Menu Items

Menu items are defined by responding to the GetMenuItems event.

The first time this event is called, MenuName will be an empty string, representing the top-level menu. For a simple Add-in with just a single menu option you can return a string, eg:

```
Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant
    Addin_GetMenuItems = "&Joe's Add-in"
End Function
```

If you wish to define sub-menus, prefix a parent menu with a dash. Parent and sub-items are defined as follows:

```
Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As String) As Variant
    Select Case MenuName
        Case ""
            'Parent Menu Item
            EA_GetMenuItems = "-&Joe's Add-in"
        Case "-&Joe's Add-in"
            'Define Sub-Menu Items using the Array notation.
            'In this example, "Diagram" and "Treeview" will compose the "Joe's Add-in" sub-
            menu.
```

```

        EA_GetMenuItems = Array("&Diagram", "&Treeview")
    Case Else
        MsgBox "Invalid Menu", vbCritical
    End Select
End Function

```

Similarly, further sub items may be defined:

```

Function EA_GetMenuItems(Repository as EA.Repository, MenuLocation As String, MenuName As
String) As Variant
    Select Case MenuName
        Case ""
            EA_GetMenuItems = "-Joe's Add-in"
        Case "-Joe's Add-in"
            EA_GetMenuItems = Array("-&Diagram", "&TreeView")
        Case "-&Diagram"
            EA_GetMenuItems = "&Properties"
        Case Else
            MsgBox "Invalid Menu", vbCritical
        End Select
    End Function

```

If you wish to enable or disable menu items by default, this method may be used to show particular items to the user:

```

Sub EA_GetMenuState(Repository As EA.Repository, Location As String, MenuName As String,
ItemName As String, IsEnabled As Boolean, IsChecked As Boolean)
    Select Case Location
        Case "TreeView"
            'Always enable
        Case "Diagram"
            'Always enable
        Case "MainMenu"
            Select Case ItemName
                Case "&Translate", "Save &Project"
                    If GetIsProjectSelected() Then
                        IsEnabled = False
                    End If
            End Select
        End Select
    End Select
    IsChecked = GetIsCurrentSelection()
End Sub

```

12.3.2.1.2 Deploying Add-Ins

Deploying Add-ins to users' sites requires three steps

1. Add the Add-in DLL file to an appropriate directory on the user's computer i.e. C:\Program Files\ [new dir].
2. Register the DLL by using the "RegSvr32" command from the command prompt as shown in the example below.

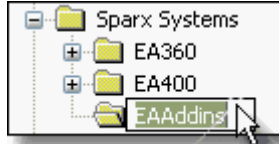


3. Place a new entry into the registry so that EA recognizes the presence of your Add-in by using the

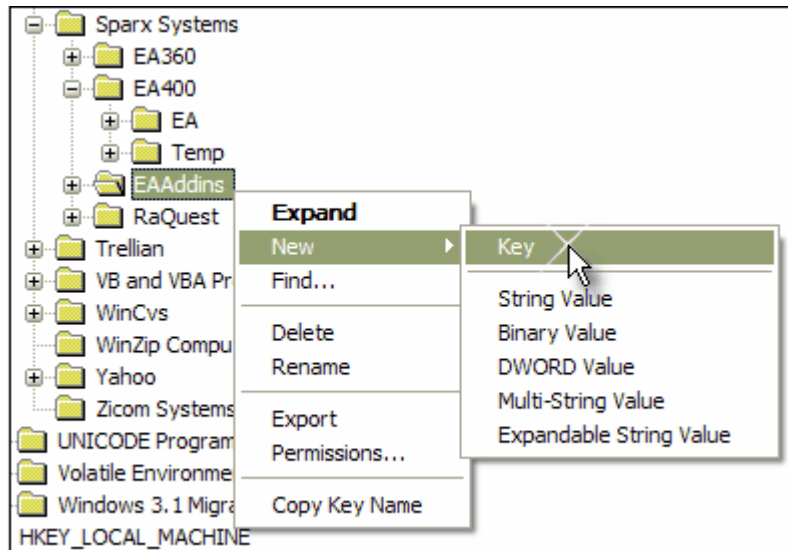
registry editor (run regedit).

4. Add a new key value EAAddIns under the following location:

HKEY_CURRENT_USER\Software\Sparx Systems

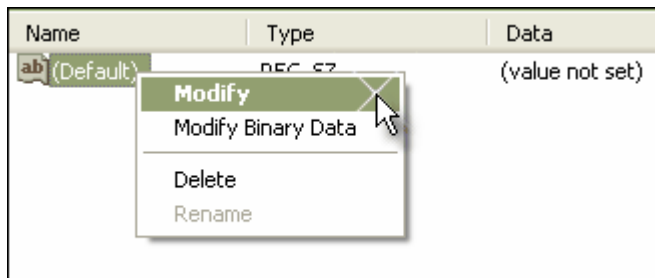


5. Then add a new key under this key with the project name.

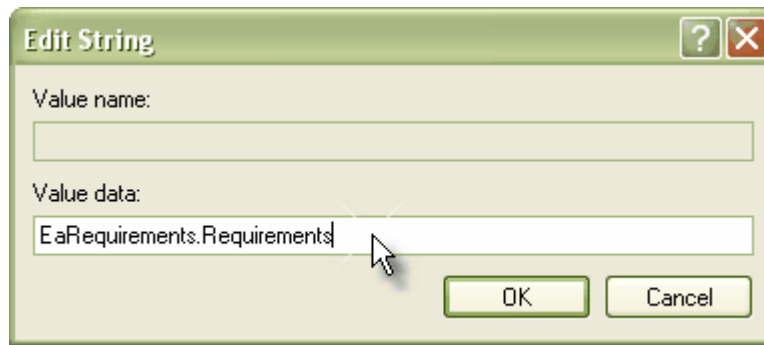


Note: [ProjectName] is not necessarily the name of your DLL, but the name of the Project. In VB, this is the value for the property "Name" corresponding to the project file.

6. Then specify the default value by modifying the default value of the key.



7. Enter the value of the key by entering the [project name].[class name]. i.e. EaRequirements.Requirements where EaRequirements is the class name as shown in the example below.



12.3.2.1.3 Tricks and Traps

Visual Basic 5/6 Users Note

Visual Basic 5/6 users should note that the version number of the EA interface is stored in the VBP project file in a form similar to the following:

```
Reference=*G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\..\Program Files\Sparx Systems\EA\EA.TLB#Enterprise Architect Object Model 2.02
```

If you experience problems moving from one version of EA to another, open the VBP file in a text editor and remove this line. Then open the project in Visual Basic and use Project-References to create a new reference to the EA Object model.

Holding State Information

It is possible for an Add-in to hold state information, meaning that data can be stored in member variables in response to one event and retrieved in another. There are some dangers in doing this:

- EA Automation Objects do not update themselves in response to user activity, to activity on other workstations, or even to the actions of other objects in the same automation client. Retaining handles to such objects between calls may result in the second event querying objects that have no relationship with the current state of EA.
- When the user asks EA to close, all Add-ins are asked to shut down. If there are any external automation clients EA will need to stay active, in which case all the add-ins will be reloaded - losing all the data.
- EA acting as an automation client will not close if an Add-in still holds a reference to it (releasing all references in the Disconnect() event avoids this problem).

It is recommended that unless there is a specific reason for doing so, the add-in should use the repository parameter and its method and properties to provide the necessary data.

EA Not Closing

.Net Specific Issues

Automation checks the use of objects and won't allow any of them to be destroyed until they are no longer being used.

As noted in the automation interface section, if your automation controller was written using the .NET framework, EA will not close even after you release all your references to it. To force the release of the COM pointers, call the memory management functions as shown below:

```
GC.Collect();
GC.WaitForPendingFinalizers();
```

Additionally, because automation clients hook into EA which create Add-ins which in turn hook back into EA,

it is possible to get into a deadlock situation where EA and the Add-ins won't let go of one another and keep each other active. An Add-in may retain hooks into EA because:

- It keeps a private reference to an EA object (see Holding state information above).
- It has been created by .NET and the GC mechanism hasn't got around to releasing it.

There are two actions required to avoid deadlock situations:

- Automation controllers must call `Repository.CloseAddins()` at some point (presumably at the end of processing).
- Add-ins must release all references to EA in the `Disconnect()` event. See the [Add-In Methods](#) page for details.

It is possible that your Automation client controls a running instance of EA where the Add-ins have not complied to the rule above. In this case you may wish to call `Repository.Exit()` to terminate EA.

Miscellaneous

In developing add-ins using the .Net framework you will be required to select COM Interoperability in the project's properties in order for it to be recognised as an add-in.

Some development environments do not automatically register com dlls on creation. You may need to do that manually before EA recognises it.

You can use your private Add-In key (as required for Add-in deployment) to store configuration information pertinent to your Add-in.

12.3.3 Add-In Events

All EA Add-Ins may choose to respond to general Add-In events.

See:

- [EA_Connect](#)
- [EA_Disconnect](#)
- [EA_GetMenuItems](#)
- [EA_MenuClick](#)
- [EA_GetMenuState](#)
- [EA_ShowHelp](#)

12.3.3.1 EA_Connect

EA_Connect

EA_Connect events allow Add-Ins to identify their type and to respond to EA start up.

Syntax

Function EA_Connect(*Repository As EA.Repository*) As String

The **EA_Connect** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.

Return Value

String identifying a specialized type of Add-In. Currently the only valid value is "MDG":

Type	Details
"MDG"	MDG Add-Ins receive MDG Events and extra menu options.

Details

This event occurs when EA first loads your Add-In. EA itself is loading at this time so that while a Repository object is supplied, there is limited information that you can extract from it.

The chief uses for EA_Connect are in initializing global Add-In data and for identifying the Add-In as an MDG Add-In.

See Also:

[EA_Disconnect](#), [MDG Add-Ins](#)

12.3.3.2 EA_Disconnect

EEA_Disconnect

The EA_Disconnect event allows the Add-In to respond to user requests to disconnect the model branch from an external project.

Syntax

Sub EA_Disconnect()

The **EA_Disconnect** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.

Return Value

None

Details

This function will be called when the EA closes. If you have stored references to EA objects (not particularly recommended anyway), you must release them here.

In addition, .NET users must call memory management functions as shown below:


```
GC.Collect();
GC.WaitForPendingFinalizers();
```

See Also:

[EA_Connect](#)

12.3.3.3 EA_GetMenuItems

EA_GetMenuItems

The EA_GetMenuItems allows the Add-In to provide the EA user interface with additional "add-In" menu items in various context and main menus. When a user selects an Add-In menu option, and event is raised and passed back to the Add-In that originally defined that menu item.

Syntax

Function EA_GetMenuItems(*Repository As EA.Repository, MenuLocation As String, MenuName As String*) As Variant

The EA_GetMenuItems function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>MenuLocation</i>	String		String representing the part of the user interface that brought up the menu. May be "TreeView", "MainMenu" or "Diagram".
<i>MenuName</i>	String		The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string.

Return Value

Either a variant containing a string or an array of strings. In the case of the top-level menu it should be a single string or an array containing only one item.

Details

This event is raised just before EA needs to show particular menu items to the user, and its use is described in the [Defining Menu Items](#) topic.

See Also:

[EA_MenuClick](#), [EA_GetMenuState](#)

12.3.3.4 EA_MenuClick

EA_MenuClick

EA_MenuClick events are received by an add-in in response to user selection of a menu item.

Syntax

Sub EA_MenuClick(Repository As EA.Repository, MenuLocation As String, MenuName As String, ItemName As String)

The **EA_GetMenuClick** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>MenuName</i>	String		The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string.
<i>ItemName</i>	String		The name of the option actually clicked, eg. "Create &New Invoice".

Return Value

None.

Details

The event is raised when the user clicks on a particular menu item. When a user clicks on one of your non-parent menu items, your Add-in will receive a MenuClick event, defined as follows:

Sub EA_MenuClick(Repository As EA.Repository, ByVal MenuName As String, ByVal ItemName As String)

The code below illustrates an example of use:

```
If MenuName = "-&Diagram" And ItemName = "&Properties" then
    MsgBox Repository.GetCurrentDiagram.Name, vbInformation
Else
    MsgBox "Not Implemented", vbCritical
End If
```

Notice that your code can directly access EA data and UI elements using [Repository](#) methods.

See Also:

[EA_GetMenuItems](#)

12.3.3.5 EA_GetMenuState**EA_GetMenuState**

The EA_GetMenuState allows the Add-In to set a particular menu option to either enabled or disabled. This is useful when dealing with locked packages and other situations where it is convenient to show a menu item - but not enable it for use. .

Syntax

Sub EA_GetMenuState(Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String, IsEnabled as Boolean, IsChecked as Boolean)

The **EA_GetMenuItems** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>MenuLocation</i>	String		String representing the part of the user interface that brought up the menu. May be "Treeview", "MainMenu" or "Diagram".
<i>MenuName</i>	String		The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string.
<i>ItemName</i>	String		The name of the option actually clicked, eg. "Create &New Invoice".
<i>IsEnabled</i>	Boolean		Boolean. Set to False to disable this particular menu item.
<i>IsChecked</i>	Boolean		Boolean. Set to True to check this particular menu item.

Return Value

None.

Details

This event is raised just before EA needs to show particular menu items to the user, and its use is described in the [Defining Menu Items](#) topic.

See Also:

[EA_GetMenuItems](#)

12.3.3.6 EA_ShowHelp

EA_ShowHelp

The EA_ShowHelp allows the Add-In to show a help topic for a particular menu option. When the user has an Add-In menu option selected, pressing F1 can be delegated to the required Help topic by the Add-In and a suitable help message shown. .

Syntax

Sub EA_ShowHelp(Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String)

The **EA_GetMenuItems** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>MenuLocation</i>	String		String representing the part of the user interface that brought up the menu. May be "Treeview", "MainMenu" or "Diagram".
<i>MenuName</i>	String		The name of the parent menu that needs its sub-items defined. In the case of the top-level menu it will be an empty string.
<i>ItemName</i>	String		The name of the option actually clicked, eg. "Create &New Invoice".

Return Value

None.

Details

This event is raised when the user presses F1 on a menu option that is not a parent menu.

See Also:

[EA_GetMenuItems](#)

12.3.4 Broadcast Events

General Broadcast are sent to all loaded Add-Ins. For an Add-In to receive the event, they must first implement the required automation event interface. If EA detects that the Add-In has the required interface, the event is dispatched on to the Add-In. MDG Events add quite a number of additional events - but the Add-In must first have registered as an MDG style Add-In, rather than as a generic Add-In.

See:

- [EA_FileOpen](#)
- [EA_OnPreNewElement](#)
- [EA_OnPreNewConnector](#)
- [EA_OnPostNewElement](#)
- [EA_OnPostNewConnector](#)
- [EA_OnDeleteTechnology](#)
- [EA_OnImportTechnology](#)

12.3.4.1 EA_FileOpen

EA_FileOpen

The EA_FileOpen allows the Add-In to respond to a File Open event. When EA opens a new Model file, this event is raised and passed to all Add-Ins implementing this method.

Syntax

Sub EA_FileOpen(*Repository As EA.Repository*)

The EA_FileOpen function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.

Return Value

None

Details

This event occurs when the model being viewed by the EA user changes, for whatever reason (through user interaction or Add-In activity).

12.3.4.2 EA_OnPreNewElement**EA_OnPreNewElement**

EA_OnPreNewElement notifies Add-ins that a new element is about to be created on a diagram. It allows add-ins to permit or deny the creation of the new element.

Syntax

Function EA_OnPreNewElement(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreNewElement** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>Info</i>	EA.EventProperties	IN	Contains the following EventProperty Objects for the element to be created : <ul style="list-style-type: none"> • Type : A string value corresponding to Element.Type • Stereotype : A string value corresponding to Element.Stereotype • ParentID : A long value corresponding to Element.ParentID • DiagramID : A long value corresponding to the ID of the diagram to which the element is being added

Return Value

Return True to allow the addition of the new element to the model. Return False to disallow the addition of the new element.

Details

This event occurs when a user drags a new element from the UML toolbox or Resource Tree, onto a

diagram. The notification is provided immediately before the element is created, so that the add-in may disallow the addition of the element.

See Also:

[EA.EventProperties](#)

12.3.4.3 EA_OnPreNewConnector

EA_OnPreNewConnector

EA_OnPreNewConnector notifies Add-ins that a new connector is about to be created on a diagram. It allows add-ins to permit or deny the creation of new connector.

Syntax

Function EA_OnPreNewConnector(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPreNewConnector** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>Info</i>	EA.EventProperties	IN	Contains the following EventProperty Objects for the connector to be created : <ul style="list-style-type: none"> •Type : A string value corresponding to Connector.Type •Subtype : A string value corresponding to Connector.Subtype •Stereotype : A string value corresponding to Connector.Stereotype •ClientID : A long value corresponding to Connector.ClientID •SupplierID : A long value corresponding to Connector.SupplierID DiagramID : A long value corresponding to Connector.DiagramID

Return Value

Return True to allow the addition of the new connector to the model. Return False to disallow the addition of the new connector.

Details

This event occurs when a user drags a new connector from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately before the connector is created, so that the add-in may disallow the addition of the connector.

See Also:

[EA.EventProperties](#)

12.3.4.4 EA_OnPostNewElement

EA_OnPostNewElement

EA_OnPostNewElement notifies Add-ins that a new element has been created on a diagram. It allows add-ins to modify the element upon creation.

Syntax

Function EA_OnPostNewElement(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPostNewElement** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>Info</i>	EA.EventProperties	IN	Contains the following EventProperty Objects for the element to be created : <ul style="list-style-type: none"> • ElementID : A long value corresponding to Element.ElementID

Return Value

Return True if the element has been updated during this notification. Return False otherwise.

Details

This event occurs after a user has dragged a new element from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately after the element is added to the model.

See Also:

[EA.EventProperties](#)

12.3.4.5 EA_OnPostNewConnector

EA_OnPostNewConnector

EA_OnPostNewConnector notifies Add-ins that a new connector has been created on a diagram. It allows add-ins to modify the connector upon creation.

Syntax

Function EA_OnPostNewConnector(*Repository As EA.Repository, Info As EA.EventProperties*) As Boolean

The **EA_OnPostNewConnector** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>Info</i>	EA.EventProperties	IN	Contains the following EventProperty Objects for the connector to be created: <ul style="list-style-type: none"> ConnectorID : A long value corresponding to Connector.ConnectorID

Return Value

Return True if the connector has been updated during this notification. Return False otherwise.

Details

This event occurs after a user has dragged a new connector from the UML toolbox or Resource Tree, onto a diagram. The notification is provided immediately after the connector is added to the model.

See Also:

[EA.EventProperties](#)

12.3.4.6 EA_OnDeleteTechnology

EA OnDeleteTechnology

EA_OnDeleteTechnology notifies Add-ins that an MDG Technology resource has been deleted from the model.

Syntax

Sub EA_OnDeleteTechnology(Repository As EA.Repository, Info As EA.EventProperties)

The **EA_OnDeleteTechnology** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>Info</i>	EA.EventProperties	IN	Contains the following EventProperty Objects : <ul style="list-style-type: none"> TechnologyID : A string value corresponding to the MDG Technology ID

Return Value

None

Details

This event occurs after a user has deleted an MDG Technology resource from the model. Add-ins that require an MDG Technology resource to be loaded, may catch this add-in to disable certain functionality

See Also:

[EA.EventProperties](#), [EA_OnImportTechnology](#)

12.3.4.7 EA_OnImportTechnology**EA_OnImportTechnology**

EA_OnImportTechnology notifies Add-ins that the user has imported an MDG Technology resource into the model.

Syntax

Sub EA_OnImportTechnology(Repository As EA.Repository, Info As EA.EventProperties)

The **EA_OnImportTechnology** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>Info</i>	EA.EventProperties	IN	Contains the following EventProperty Objects: <ul style="list-style-type: none"> TechnologyID : A string value corresponding to the MDG Technology ID

Return Value

None

Details

This event occurs after a user has imported an MDG Technology resource into the model. Add-ins that require an MDG Technology resource to be loaded, may catch this add-in to enable certain functionality

See Also:

[EA.EventProperties](#), [EA_OnDeleteTechnology](#)

12.3.5 Custom Views

Version 4.5 of EA allows custom windows to be inserted as a tab into the Diagram Tabs that appear at the center of the EA frame.

Providing a custom view allows users to easily and quickly tab between a custom interface and diagrams and other views normally provided by EA.

Uses for this facility include:

- Reports and graphs showing summary data of the model.
- Alternate views of a diagram.
- Alternate views of the model.
- Views of external data related to model data.
- Documentation tools.

See:

- [Creating a Custom View](#)

12.3.5.1 Creating a Custom View

A custom view needs to be designed as an ActiveX custom control and inserted through the automation interface.

ActiveX custom controls can be created using most well-known programming tools including Microsoft Visual Studio.NET. Refer to the documentation provided by the relevant vendor on how to create a custom control to produce an OCX file.

Once it has been created and registered on the target system, it can be added through the AddTab() method of the [Repository](#) object.

While it is possible to call AddTab() from any automation client, it is likely that you will want to call it from an add-in, and that add-in will be defined in the same OCX that provides the custom view.

C# example code is shown here:

```

public class Addin
{
    UserControl1 m_MyControl;

    public void EA_Connect(EA.Repository Rep)
    {
    }

    public object EA_GetMenuItems(EA.Repository Repository, string Location, string
MenuName)
    {
        if( MenuName == "" )
            return "-&C# Control Demo";
        else
        {
            String[] ret = {"&Create", "&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();
    }
}

```

12.3.6 MDG Add-Ins

MDG Add_Ins are specialized types of Add-Ins that have additional features and extra requirements for Add-In authors wishing to contribute to EA's goal of Model Driven Generation.

Unlike general Add-In events, MDG Add-In events are only sent to the add-in that has taken ownership of a EA model branch on a particular PC.

One of the additional responsibilities of an MDG Add-In is to take ownership of a branch of an EA model which is done through the [MDG_Connect](#) event.

MDG Add-Ins identify themselves as such during [EA_Connect](#) by returning the string "MDG".

Unlike ordinary Add-Ins, responding to MDG Add-In events is not optional, and methods need to be published for each of the [MDG Events](#).

Two examples of MDG Add-Ins are the commercially available MDG Link for Eclipse and MDG Link for Visual Studio, published by [Sparx Systems](#).

12.3.6.1 MDG Events

An MDG Add-In must respond to all MDG Events, these events usually identify processes such as Build, Run, Synchronize, PreMerge, PostMerge and many others.

An MDG Link Add-In is expected to implement some form of forward and reverse engineering capability within EA - and as such requires access to a specific set of events - all to do with generation, synchronization and general processes concerned with converting Models to Code and Code to Models.

See:

- [MDG_BuildProject](#)
- [MDG_Connect](#)
- [MDG_Disconnect](#)
- [MDG_GetConnectedPackages](#)
- [MDG_GetProperty](#)
- [MDG_Merge](#)
- [MDG_NewClass](#)
- [MDG_PostGenerate](#)
- [MDG_PostMerge](#)
- [MDG_PreGenerate](#)
- [MDG_PreMerge](#)
- [MDG_PreReverse](#)
- [MDG_RunExe](#)
- [MDG_View](#)

12.3.6.1.1 MDG Add-Ins MDGBuild Project

MDG_BuildProject

The MDG_BuildProject allows the Add-In to handle file changes caused by generation.

Syntax

Sub MDG_BuildProject(*Repository As EA.Repository, PackageGuid As String*)

The **MDG_BuildProject** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.

Return Value

None

Details

This function will be called in response to a user selecting "Build Project" from the Add-ins menu. Respond to this event by compiling the project source files into a running application.

See Also:

[MDG_RunExe](#)

12.3.6.1.2 MDG Add-Ins MDGConnect**MDG Connect**

The MDG_Connect allows the Add-In to handle user driven request to connect a model branch to an external application.

Syntax

Function MDG_Connect(*Repository As EA.Repository, ProjectGuid As String*) As Long

The **MDG_Connect** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageID</i>	Long	IN	The PackageID of the EA package the user has requested to have connected to an external project.
<i>ProjectGuid</i>	String	IN	The unique ID identifying the project provided by the add-in when a connection to a project branch of an EA model was first established

Return Value

Return a non-zero to indicate that a connection has been made, a zero to indicates that the user has not nominated a project and connection should not proceed

Details

This function will be called when the user has attempts to connect a particular EA package to a yet unspecified external project. This event allows the Add-In to interact with the user to specify such a project.

The Add-In is responsible for retaining the connection details, which need to be stored on a per-user or per-workstation basis. i.e. Users who share a common EA model over a network should be able to connect and disconnect to external projects in a way that is completely transparent to other users.

The Add-In must therefore not store connection details in an EA repository. A suitable place to store such details would be `SHGetFolderPath(..CSIDL_APPDATA..)AddinName`.

The PackageGuid parameter is the same identifier is required for most events relating to the MDG Add-In. Therefore it is recommended that the connection details be indexed using the PackageGuid value.

See Also:

[MDG_Disconnect](#)

12.3.6.1.3 MDG Add-Ins MDGDisconnect

MDG_Disconnect

The MDG_Disconnect allows the Add-In to respond to user requests to disconnect the model branch from an external project.

Syntax

Function MDG_Disconnect(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_Disconnect** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.

Return Value

Return a non-zero to indicate that a disconnection has occurred allowing EA to update the user interface. A zero to indicates that the user has not disconnected from an external project.

Details

This function will be called when the user has attempts to disconnect an associated external project. The Add-In is required to delete the details the connection.

See Also:

[MDG_Connect](#)

12.3.6.1.4 MDG Add-Ins MDGGetConnectedPackages

MDG GetConnectedPackages

The MDG_GetConnectedPackages allows the Add-In to return a list of current connection between EA and an external application.

Syntax

Function MDG_GetConnectedPackages(*Repository As EA.Repository*) As Variant

The MDG_GetConnectedPackages function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.

Return Value

Return an array of guid strings representing individual EA packages.

Details

This function will be called when the Add-In is first loaded, and is expected to return a list of the available connections to external projects for this Add-In.

See Also:

[MDG Connect](#)

12.3.6.1.5 MDG Add-Ins MDGGetProperty

MDG GetProperty

MDG_GetProperty provides miscellaneous Add-In details to EA.

Syntax

Function MDG_GetProperty(*Repository As EA.Repository, PackageGuid As String, PropertyName As String*) As Variant

The MDG_GetProperty function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.
<i>PropertyName</i>	String	IN	The name of the property that will be used by EA See details for the possible values

Return Value

See Details

Details

This function will be called by EA to poll the Add-In for information relating to the PropertyName. This event should occur in as short a duration as possible as EA does not cache the information provided by the function.

Values corresponding to the following PropertyNames need to be provided:

IconID

Return the name of a dll and a resource identifier in the format #ResID, where the resource ID indicates an Icon e.g. c:\program files\myapp\myapp.dll#101

Language

Return the default language that classes should be assigned when they are created in EA.

HiddenMenus

Return one or more values from the MDGMenu enumeration to hide menus that do not apply to your Add-In. eg.

```
if( PropertyName == "HiddenMenus" )
    return mgBuildProject + mgRun;
```

12.3.6.1.6 MDG Add-Ins MDGMerge**MDG Merge**

The MDG_Merge allows the Add-In to jointly handle changes to both the model branch and the code project that the model branch is connected to.

Syntax

Function MDG_Merge(*Repository As EA.Repository, PackageGuid As String, SynchObjects As Variant, SynchType As Variant, ExportObjects As Variant, ExportFiles As Variant, ImportFiles As Variant, IgnoreLocked As Variant, Language As String*) As Long

The **MDG_Merge** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.
<i>SynchObjects</i>	Variant	OUT	A string array containing a list of objects (Object ID format) to be jointly synchronized between the model branch and the project. Refer below for the format of the Object IDs
<i>SynchType</i>	Variant	OUT	The integer value determining the user-selected type of synchronization to take place. Refer below for a list of valid values
<i>ExportObjects</i>	Variant	OUT	The string array containing the list of new model objects (in Object ID format) to be exported by EA to the code project.
<i>ExportFiles</i>	Variant	OUT	A string array containing the list of files for each model object chosen for export by the add-in. Each entry in this array shall have a corresponding entry in the ExportObjects parameter at the same array index, so ExportFiles(2) shall contain the filename of the object by ExportObjects(2).
<i>ImportFiles</i>	Variant	OUT	A string array containing the list of code files made available to the code project to be newly imported to the model. EA will import each file listed in this array for import into the connected model branch.
<i>IgnoreLocked</i>	Variant	OUT	A boolean value containing the user-selected option to ignore any files locked by the code project.
<i>Language</i>	String	OUT	The string value containing the name of the code language supported by the code project connected to the model branch.

Return Value

Return a non-zero if the Merge operation completed successfully and a zero value when the operation has been unsuccessful.

Details

This event shall be called whenever the user has asked to merge their model branch with its connected code project, or whenever the user has established a new connection to a code project. The purpose of this event is to allow the add-in to interact with the user to perform a merge between the model branch and the connected project.

Merge

A merge comprises of three major operations:

- Export: Where newly created model objects are exported into code and made available to the code project.
- Import: Where newly created code objects, classes, etc... are imported into the model.
- Synchronize: Where objects available both to the model and in code are jointly updated to reflect changes made in either the model, code project or both.

Synchronize Type

The Synchronize operation can take place in one of four different ways, each of these ways correspond to a value returned by SynchType:

- None: (SynchType = 0) No synchronization is to be performed.
- Forward: (SynchType = 1) Forward synchronization, between the model branch and the code project is to occur.
- Reverse: (SynchType = 2) Reverse synchronization, between the code project and the model branch is to occur.
- Both: (SynchType = 3) Reverse, then Forward synchronization's are to occur.

Object ID Format

Each of the Object IDs listed in the string arrays described above shall be composed of the following format: (@namespace) (#class)*(\$attribute|%operation|:property)*

See Also:

[MDG_Connect](#), [MDG_PreMerge](#), [MDG_PostMerge](#)

12.3.6.1.7 MDG Add-Ins MDGNewClass

MDG NewClass

The MDG_NewClass allows the Add-In to run the target application.

Syntax

Function MDG_NewClass(*Repository As EA.Repository, PackageGuid As String, CodeID As String, Language As String*) As String

The MDG_NewClass function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.
<i>CodeID</i>	String	IN	A string used to identify the code element before it is created, for more information see MDG_View
<i>Language</i>	String	IN	A string used to identify the programming language for the new class. The language needs to be supported by EA.

Return Value

Return a string containing the file path that should be assigned to the class.

Details

This method is called when EA generates a new class, and needs information relating to assigning the language and file path. The file path should be passed back as a return value and the language should be passed back via the language parameter.

See Also:

[MDG_PreGenerate](#)

12.3.6.1.8 MDG Add-Ins MDGPostGenerate**MDG_PostGenerate**

The MDG_PostGenerate allows the Add-In to handle file changes caused by generation.

Syntax

Function MDG_PostGenerate(*Repository As EA.Repository, PackageGuid As String, FilePath As String, FileContents As String*) As Long

The **MDG_PostGenerate** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.
<i>FilePath</i>	String	IN	The path of the file EA intends to overwrite.
<i>FileContents</i>	String	IN	A string containing the proposed contents of the file.

Return Value

Return a value depends on the type of event that this function is responding to (see Details). This function is required to handle two separate and distinct cases.

Details

This event will be called after EA has prepared text to replace the existing contents of a file. Responding to this event will allow the add-in to write to the linked application's user interface rather than modify the file directly. When the contents of a file is changed, EA will pass FileContents as a non-empty string. New files created as a result of code generation are also sent through this mechanism, allowing Add-Ins to add new files to the linked project's file list. When new files are created EA will pass FileContents as an empty string. When a non-zero is returned by this function, the Add-In has successfully written the contents of the file. A zero value for the return indicates to EA that the file needs to be saved.

See Also:

[MDG_PreGenerate](#)

12.3.6.1.9 MDG Add-Ins MDGPostMerge

MDG PostMerge

The MDG_PostMerge is called after a merge process has been completed.

Syntax

Function MDG_PostMerge(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_PostMerge** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.

Return Value

Return a zero value if the post-merge process has failed, a non-zero return indicates that the post-merge has been successful. EA assumes a non-zero return if this method is not implemented

Details

This function will be called by EA after the merge process has been completed.

Note: File save checking should not be performed with this function, but should be handled by Pre/Post Generate and Reverse.

See Also:

[MDG_PreMerge](#), [MDG_PreGenerate](#), [MDG_PostGenerate](#), [MDG_PreReverse](#), [MDG_Merge](#)

12.3.6.1.10 MDG Add-Ins MDGPreGenerate

MDG PreGenerate

The MDG_PreGenerate allows the Add-In to deal with unsaved changes.

Syntax

Function MDG_PreGenerate(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_PreGenerate** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.

Return Value

Return a zero value to abort generation. Any other value will allow the generation to continue.

Details

This function will be called immediately before EA attempts to generate files from the model. A possible use of this function would be to prompt the user to save unsaved source files.

See Also:

[MDG_PostGenerate](#)

12.3.6.1.11 MDG Add-Ins MDGPreMerge**MDG PreMerge**

The MDG_PreMerge is called after merge process has been initiated by the user and before EA performs the merge process.

Syntax

Function MDG_PreMerge(*Repository As EA.Repository, PackageGuid As String*) As Long

The **MDG_PreMerge** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.

Return Value

A return value of zero indicates that the merge process will not occur, if the value is not zero the merge process will go ahead. If this method is not implemented then it is assumed that a merge process will be used.

Details

This event will be called after a user has performed their interactions with the merge screen and has confirmed the merge with the **OK** button, but before EA performs the merge process using the data provided by the MDG_Merge call, before any changes have been made to the model or the connected project.

This event is made available to provide the Add-In the opportunity to generally set internal Add-In flags to augment the MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse events.

Note: File save checking should not be performed with this function, but should be handled by Pre/Post Generate and Reverse.

See Also:

[MDG_PreGenerate](#), [MDG_PostGenerate](#), [MDG_PreReverse](#), [MDG_Merge](#), [MDG_PostMerge](#)

12.3.6.1.12 MDG Add-Ins MDGPreReverse

MDG PreReverse

The MDG_PreReverse allows the Add-In to save file changes before being imported into EA.

Syntax

Sub MDG_PreReverse(*Repository As EA.Repository, PackageGuid As String, FilePaths As Variant*)

The **MDG_PreReverse** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.
<i>FilePaths</i>	String array	IN	An array of filepaths pointed to the files that are to be reverse engineered

Return Value

None.

Details

This function will be with a list of files that about to be reverse-engineered into EA. If the user is working on unsaved versions of these files in an editor, you may wish to prompt the user or save automatically.

See Also:

[MDG_PostGenerate](#), [MDG_PreGenerate](#)

12.3.6.1.13 MDG Add-Ins MDGRunExe

MDG RunExe

The MDG_RunExe allows the Add-In to run the target application.

Syntax

Sub MDG_RunExe(*Repository As EA.Repository, PackageGuid As String*)

The **MDG_RunExe** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.

Return Value

None

Details

This function will be called in response to a user selecting "Run Exe" from the Add-ins menu. Respond to this event by launching the compiled application.

See Also:

[MDG_BuildProject](#)

12.3.6.1.14 MDG Add-Ins MDGView**MDG_View**

The MDG_View allows the Add-In to display user specified code elements.

Syntax

Function MDG_View(*Repository As EA.Repository, PackageGuid As String, CodeID as String*) As Long

The **MDG_View** function syntax contains the following elements

Parameter	Type	Direction	Description
<i>Repository</i>	EA.Repository	IN	An EA.Repository object representing the currently open EA model. Poll its members to retrieve model data and user interface status information.
<i>PackageGuid</i>	String	IN	The guid identifying the EA package sub-tree that is controlled by the Add-In.
<i>CodeID</i>	String	IN	<p>A CodeID identifies the code element in the following format:</p> <pre><type>ElementPart<type>ElementPart...</pre> <p>where each element is preceded with a token identifying its type:</p> <ul style="list-style-type: none">@ - namespace# - class\$ - attribute% - operation <p>For example if a user has selected the m_Name attribute of Class1 located in namespace Name1, the class ID would be passed through in the following format:</p> <pre>@Name1#Class1%m_Name</pre>

Return Value

Return a non-zero value to indicate that the Add-In has processed the request. Returning a zero value will result in EA employing the standard viewing process which is to launch the associated source file.

Details

This function will be called by EA when the user asks to view a particular code element. This allows the Add-In to present that element in its own way – usually in a code editor.

Part

XIII

13 Glossary of Terms

This section provides a detailed glossary for Enterprise Architect.

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#)

13.1 A (Glossary)

~A~



abstract class

A class that cannot be directly instantiated.

Contrast: concrete class

abstraction

The essential characteristics of an entity that distinguish it from all other kinds of entities. An abstraction defines a boundary relative to the perspective of the viewer.

action

The specification of an executable statement that forms an abstraction of a computational procedure. An action typically results in a change in the state of the system, and can be realized by sending a message to an object or modifying a link or a value of an attribute. .

action sequence

An expression that resolves to a sequence of actions.

action state

A state that represents the execution of an atomic action, typically the invocation of an operation.

activation

The execution of an action.

active class

A class whose instances are active objects. When instantiated, an active class will control its execution. Rather than being invoked or activated by other objects, it can operate standalone, and define its own thread of behavior.

See also: active object

activation

An object that owns a thread and can initiate control activity. An instance of active class.

See also: Active class, thread

activity

An activity defines the bounds for the structural organization that contains a set of basic or fundamental behaviors. It can be used to model procedural type application development for system design through to modeling business processes in organizational structures and workflow.

activity diagram

An activity diagram can be used to model procedural type application development for system design through to modeling business processes in organizational structures and workflow.

activity graph

A special case of a state machine that is used to model processes involving one or more classifiers.

Contrast: state chart diagram

actor [class]

A coherent set of roles that users of use cases play when interacting with these use cases. An actor has one role for each use case with which it communicates.

actual parameter

Synonym: argument

aggregate [class]

A class that represents the "whole" in an aggregation (whole-part) relationship.

See also: aggregation

aggregation

A special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part.

See also: composition

analysis

The part of the software development process whose primary purpose is to formulate a model of the problem domain. Analysis focuses on what to do, design focuses on how to do it.

Contrast: design

analysis diagram

An *Analysis diagram* is used to capture high level business processes and early models of system behavior and elements. It is less formal than some other diagrams, but provides a good means of capturing the essential business characteristics and needs.

analysis time

Refers to something that occurs during an analysis phase of the software development process.

See also: design time, modeling time

architecture

The organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

argument

A binding for a parameter that resolves to a run-time instance.

Synonym: actual parameter

Contrast: parameter

artifact

A physical piece of information that is used or produced by a software development process. Examples of Artifacts include models, source files, scripts, and binary executable files. An artifact may constitute the implementation of a deployable component.

Synonym: product

Contrast: component

assembly

An assembly connector bridges the required interface of a component with the provided interface of a second component.

association

The semantic relationship between two or more classifiers that specifies connections among their instances.

association class

A model element that has both association and class properties. An association class can be seen as an association that also has class properties, or as a class that also has association properties.

association end

The endpoint of an association, which connects the association to a classifier.

attribute

A feature within a classifier that describes a range of values that instances of the classifier may hold.

auxiliary class

A stereotyped class that supports another more central or fundamental class, typically by implementing secondary logic or control flow. Auxiliary classes are typically used together with focus classes, and are particularly useful for specifying the secondary business logic or control flow of components during design.

See also: focus

13.2 B (Glossary)

~B~

behavior

The observable effects of an operation or event, including its results.

behavioral diagram

Behavioral diagrams depict the behavioral features of a system or business process. Behavioral diagrams include Activity diagrams, State Machine diagrams, Communication diagrams, Interaction



Overview diagrams, Sequence diagrams, Timing diagrams and Use Case diagrams.

behavioral feature

A dynamic feature of a model element, such as an operation or method.

behavioral model aspect

A model aspect that emphasizes the behavior of the instances in a system, including their methods, collaborations, and state histories.

binary association

An association between two classes. A special case of an n-ary association.

binding

The creation of a model element from a template by supplying arguments for the parameters of the template.

bookmark

A marker in a rich text document that allows you to link inner sections of a document into a master document (using Word insert file function).

boolean

An enumeration whose values are true and false.

boolean expression

An expression that evaluates to a boolean value.

boundary

1. A *Boundary* is a stereotyped class that models some system boundary – typically a user interface screen. It is used in the conceptual phase to capture users interacting with the system at a screen level (or some other boundary interface type). It is often used in sequence and robustness (analysis) diagrams. It is the View in the Model-View-Controller pattern.
2. A *System Boundary* element is used to delineate a particular part of the system.

13.3 C (Glossary)

~C~**C++**

An object-oriented programming language based on the earlier 'C' language.

call

An action state that invokes an operation on a classifier.

cardinality

The number of elements in a set.

Contrast: multiplicity



CASE

Computer Aided Software Engineering. A tool designed for the purpose of modeling and building software systems.

child

In a generalization relationship, the specialization of another element, the parent.

See also: subclass, subtype.

Contrast: parent

choice

The choice pseudo-state is used to compose complex transitional paths, where the outgoing transition path is decided by dynamic, run-time conditions determined by the actions performed by the state machine on the path leading to the choice.

class

A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment.

See also: interface

class diagram

A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.

classification

The assignment of an object to a classifier. See dynamic classification, multiple classification and static classification.

classifier

A mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, datatypes, and components.

client

A classifier that requests a service from another classifier.

Contrast: supplier

collaboration

The specification of how an operation or classifier, such as a use case, is realized by a set of classifiers and associations playing specific roles used in a specific way. The collaboration defines an interaction.

See also: interaction

collaboration diagram

Used pre - UML 2.0.

collaboration occurrence

Use an Occurrence to apply a pattern defined by a collaboration to a specific situation.

combined fragment

A combined fragment reflects a piece or pieces of interaction (called interaction operands) controlled by an interaction operator, whose corresponding boolean conditions are known as interaction constraints. It appears graphically as a transparent window, divided by horizontal dashed lines for each operand.

comment

An annotation attached to an element or a collection of elements. A note has no semantics.

Contrast: constraint

communication diagram

A Communication diagram shows the interactions between elements at run-time in much the same manner as a sequence diagram. However, communication diagrams are used to visualize inter-object relationships, while sequence diagrams are more effective at visualizing processing over time.

compile time

Refers to something that occurs during the compilation of a software module.

See also: modeling time, run time

component

A modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component is typically specified by one or more classifiers (e.g., implementation classes) that reside on it, and may be implemented by one or more artifacts (e.g., binary, executable, or script files).

Contrast: artifact

component diagram

A diagram that shows the organizations and dependencies among components.

composite [class]

A class that is related to one or more classes by a composition relationship.

See also: composition

composite state

A state that consists of either concurrent (orthogonal) substates or sequential (disjoint) substates.

See also: substate

composite structure diagram

A composite structure diagram reflects the internal collaboration of classes, interfaces, or components to describe a functionality. Composite structure diagrams are similar to class diagrams, except that they model a specific usage of the structure.

composition

A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive.

Synonym: composite aggregation

concrete class

A class that can be directly instantiated.

Contrast: abstract class

concurrency

The occurrence of two or more activities during the same time interval. Concurrency can be achieved by interleaving or simultaneously executing two or more threads.

See also: thread

concurrent substate

A substate that can be held simultaneously with other substates contained in the same composite state.

See also: composite state

Contrast: disjoint substate

connection

A logical link between model elements. May be structural, dynamic or possessive.

constraint

A semantic condition or restriction. Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML.

See also: tagged value, stereotype

constraint

A rule or condition that applies to some element. It is often modeled as a pre- or post- condition.

container

1. An instance that exists to contain other instances, and that provides operations to access or iterate over its contents.(for example, arrays, lists, sets).
2. A component that exists to contain other components.

containment hierarchy

A namespace hierarchy consisting of model elements, and the containment relationships that exist between them. A containment hierarchy forms a graph.

context

A view of a set of related modeling elements for a particular purpose, such as specifying an operation.

continuation

A Continuation is used in seq and alt combined fragments, to indicate the branches of continuation an operand follows.

control

A Control is a stereotyped class that represents a controlling entity or manager. A control organizes and schedules other activities and elements. It is the controller of the Model-View-Controller pattern.

control flow

The control flow is a connector linking two nodes in an activity diagram. Control Flow connectors start a nodes activity when the preceding nodes action is finished.

13.4 D (Glossary)

~D~



database schema

A database schema is the description of a database structure. It defines tables and fields and the relationship between them.

datastore

A datastore element is used to define permanently stored data. A token of data that is stored in the Datastore is stored permanently. A token of data that comes out of the Datastore is a copy of the original data. The tokens imported are kept for the life of the Activity in which it exists.

datatype

A descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations.

decision

A Decision is an element of an Activity diagram that indicates a point of conditional progression: if a condition is true, then processing continues one way, if not, then another.

defining model [MOF]

The model on which a repository is based. Any number of repositories can have the same defining model.

delegate

A delegate connector defines the internal assembly of a component's external ports and interfaces. Using a delegate connector wires the internal workings of the system to the outside world, by a delegation of the external interfaces' connections.

delegation

The ability of an object to issue a message to another object in response to a message. Delegation can be used as an alternative to inheritance.

Contrast: inheritance

dependency

A relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).

deployment

A deployment is a type of dependency relationship that indicates the deployment of an artifact onto a node or executable target.

deployment diagram

A diagram that shows the configuration of run-time processing nodes and the components, processes, and objects that live on them. Components represent run-time manifestations of code units.

See also: component diagrams

deployment spec

A deployment specification specifies parameters guiding deployment of an artifact, as is common with most hardware and software technologies.

derived element

A model element that can be computed from another element, but that is shown for clarity or that is included for design purposes even though it adds no semantic information.

design

The part of the software development process whose primary purpose is to decide how the system will be implemented. During design strategic and tactical decisions are made to meet the required functional and quality requirements of a system.

design time

Refers to something that occurs during a design phase of the software development process.

See also: modeling time

Contrast: analysis time

development process

A set of partially ordered steps performed for a given purpose during software development, such as constructing models or implementing models.

diagram

A graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs (relationships) and vertices (other model elements). UML supports the following diagrams: class diagram, object diagram, use case diagram, sequence diagram, collaboration diagram, state diagram, activity diagram, component diagram, and deployment diagram.

diagram gate

A diagram gate is a simple graphical way to indicate the point at which messages can be transmitted into and out of interaction fragments.

diagram view

The workspace area where the UML diagrams are displayed.

disjoint substate

A substate that cannot be held simultaneously with other substates contained in the same composite state.

See also: composite state

Contrast: concurrent substate

distribution unit

A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.

domain

An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.

dynamic classification

A semantic variation of generalization in which an object may change its classifier.

Contrast: static classification

13.5 E (Glossary)

~E~



element

An atomic constituent of a model.

element

A model object of any type - class, component, node, object or etc.

endpoint

An endpoint is used in interaction diagrams to reflect a lost message in sequence.

entity

An Entity is a store or persistence mechanism that captures the information or knowledge in a system. It is the Model in the Model-View-Controller pattern.

entry action

An action executed upon entering a state in a state machine regardless of the transition taken to reach that state.

entry point

Entry points are used to define where external states can enter a submachine.

enumeration

A list of named values used as the range of a particular attribute type. For example, RGBColor = {red, green, blue}. Boolean is a predefined enumeration with values from the set {false, true}.

event

The specification of a significant occurrence that has a location in time and space. In the context of state diagrams, an event is an occurrence that can trigger a transition.

exception handler

The exception handler element defines the group of operations to carry out when an exception occurs.

exit action

An action executed upon exiting a state in a state machine regardless of the transition taken to exit that state.

exit point

Exit points are used in submachine states and state machines to denote the point where the machine will be exited and the transition sourcing this exit point, for submachines, will be triggered. Exit points are a type of pseudo-state used in the state machine diagram.

export

In the context of packages, to make an element visible outside its enclosing namespace.

See also: visibility

Contrast: export [OMA], import

expose interface

The expose interface toolbox element is a graphical way to depict the required and supplied interfaces of a component, class, or part.

expression

A string that evaluates to a value of a particular type. For example, the expression "(7 + 5 * 3)" evaluates to a value of type number. A relationship from an extension use case to a base use case, specifying how the behavior defined for the extension use case augments (subject to conditions specified in the extension) the behavior defined for the base use case. The behavior is inserted at the location defined by the extension point in the base use case. The base use case does not depend on performing the behavior of the extension use case. See extension point, include.

extend

An Extend connection is used to indicate an element extends the behavior of another. Extensions are used in use case models to indicate one use case (optionally) extends the behavior of another.

13.6 F (Glossary)

~F~**facade**

A stereotyped package containing only references to model elements owned by another package. It is used to provide a 'public view' of some of the contents of a package.

feature

A property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a datatype.

final

A final pseudo-state indicates an end.

final state

A special kind of state signifying that the enclosing composite state or the entire state machine is completed.

fire

To execute a state transition.

See also: transition

flow final

The flow final element depicts an exit from the system as opposed to the Activity Final which represents the completion of the activity.

focus class

A stereotyped class that defines the core logic or control flow for one or more auxiliary classes that support it. Focus classes are typically used together with one or more auxiliary classes, and are particularly useful for specifying the core business logic or control flow of components during design.

See also: auxiliary

focus of control

A symbol on a sequence diagram that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

forward engineering

The process of generating source code from the UML model.

fork

Forks are utilized in State Machine diagrams as pseudo-states. With respect to state machine diagrams, a fork pseudo-state signifies that its incoming transition will come from a single state, and it will have multiple outgoing transitions.

framework

A stereotyped package that contains model elements which specify a reusable architecture for all or part of a system. Frameworks typically include classes, patterns or templates. When frameworks are specialized for an application domain, they are sometimes referred to as application frameworks.

See also: pattern

13.7 G (Glossary)

~G~**generalizable element**

A model element that may participate in a generalization relationship.

See also: generalization

generalization

A taxonomic relationship between a more general element and a more specific element. The more specific element is fully consistent with the more general element and contains additional information. An instance of the more specific element may be used where the more general element is allowed.

See also: inheritance

guard condition

A condition that must be satisfied in order to enable an associated transition to fire.

13.8 H (Glossary)

~H~



history state

There are two types of history pseudo-states defined in UML, shallow and deep history. A shallow history sub-state is used to represent the most recently active sub-state of a composite state. A deep history sub-state, in contrast, reflects the most recent active configuration of the composite state.

13.9 I (Glossary)

~I~



implementation

A definition of how something is constructed or computed. For example, a class is an implementation of a type, a method is an implementation of an operation.

implementation class

A stereotyped class that specifies the implementation of a class in some programming language (e.g., C++, Smalltalk, Java) in which an instance may not have more than one class. An Implementation class is said to realize a type if it provides all of the operations defined for the type with the same behavior as specified for the type's operations.

See also: type

implementation inheritance

The inheritance of the implementation of a more general element. Includes inheritance of the interface.

Contrast: interface inheritance

import

In the context of packages, a dependency that shows the packages whose classes may be referenced within a given package (including packages recursively embedded within it).

Contrast: export

include

A relationship from a base use case to an inclusion use case, specifying how the behavior for the base use case contains the behavior of the inclusion use case. The behavior is included at the location which is defined in the base use case. The base use case depends on performing the behavior of the inclusion use case, but not on its structure (ie., attributes or operations).

See also: extend

inheritance

The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior.

See also: generalization

initial state

The Initial pseudo-state is used to denote the default state of a composite state; there can be one initial vertex in each region of the composite state.

interaction diagram

Interaction diagrams can be sequence diagrams, communication diagrams, interaction overview diagrams, and timing diagrams. Interaction diagrams include Timing Diagrams, Sequence Diagrams, Interaction Overview Diagrams and Communication Diagrams.

instance

An entity that has unique identity, a set of operations that can be applied to it, and state that stores the effects of the operations.

See also: object

interaction

A specification of how stimuli are sent between instances to perform a specific task. The interaction is defined in the context of a collaboration.

See also: collaboration

interaction diagram

A generic term that applies to several types of diagrams that emphasize object interactions. These include collaboration diagrams and sequence diagrams.

interaction occurrence

An interaction occurrence is a reference to an existing interaction element. Interaction occurrences are visually represented by a frame, with "ref" in the frame's title space. The diagram name is indicated in the frame contents.

interaction overview diagram

Interaction Overview diagrams visualize the cooperation between other interaction diagrams to illustrate a control flow serving an encompassing purpose. As interaction overview diagrams are a variant of activity diagrams, most of the diagram notation is similar, as is the process in constructing the diagram.

interface

A named set of operations that characterize the behavior of an element.

interface inheritance

The inheritance of the interface of a more general element. Does not include inheritance of the implementation.

Contrast: implementation inheritance

internal transition

A transition signifying a response to an event without changing the state of an object.

interrupt flow

A EA defined toolbox element used to define the exception handler and interruptible activity region concepts.

13.10 J (Glossary)

~J~



Java

A fully object-oriented, cross platform language based on elements from Smalltalk, C++ and other OO languages.

join

Joins are utilized in state machine diagrams and in activity diagrams to synchronize multiple flows.

junction

Junction pseudo-states are used to design complex transitional paths. A junction can be used to combine, or merge, multiple paths into a shared transition path or to split an incoming path into multiple paths.

13.11 L (Glossary)

~L~



layer

The organization of classifiers or packages at the same level of abstraction. A layer represents a horizontal slice through an architecture, whereas a partition represents a vertical slice.

Contrast: partition

lifeline

A lifeline is an individual participant in an interaction (i.e., lifelines cannot have multiplicity). A lifeline represents a distinct connectable element.

link

A semantic connection among a tuple of objects. An instance of an association.

See also: association

link end

An instance of an association end.

See also: association end

local path

A relative path on a local machine. Allows developers to store shared source code in machine specific directories, but still generate and synchronize code.

13.12 M (Glossary)

~M~



maintenance

The support of a software system after it is deployed.

manifest

A manifest relationship indicates that the artifact source embodies the target model element. Stereotypes can be added to Enterprise Architect to classify the type of manifestation of the model element.

message

Messages indicate a flow of information, or transition of control, between elements. Messages are used by Communication diagrams, Sequence diagrams, Interaction Overview diagrams and Timing diagrams.

message endpoint

The Message Endpoint element defines an endpoint of a Lifeline such a State or Value Lifeline in a Timing diagram.

message label

A message label is used for messages sent between lifelines to make the diagram appear less cluttered. Labels with the same name indicate that a message may be interrupted.

metaclass

A class whose instances are classes. Metaclasses are typically used to construct metamodels.

metafile

A vector based image format native to Windows. Supports high detail and excellent scaling. Typically used for saving diagram images for placement in documents. Comes in Placeable (an older format) and Enhanced (current standard format).

meta-metamodel

A model that defines the language for expressing a metamodel. The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model.

metamodel

A model that defines the language for expressing a model.

metaobject

A generic term for all metaentities in a metamodeling language. For example, metatypes, metaclasses, metaattributes, and metaassociations.

method

The implementation of an operation. It specifies the algorithm or procedure associated with an operation.

model [MOF]

An abstraction of a physical system with a certain purpose.

See also: physical system

Usage note: In the context of the MOF specification, which describes a meta-metamodel, for brevity the meta-metamodel is frequently to as simply the model.

model aspect

A dimension of modeling that emphasizes particular qualities of the metamodel. For example, the structural model aspect emphasizes the structural qualities of the metamodel.

model elaboration

The process of generating a repository type from a published model. Includes the generation of interfaces and implementations which allows repositories to be instantiated and populated based on, and in compliance with, the model elaborated.

model element [MOF]

An element that is an abstraction drawn from the system being modeled.

Contrast: view element. In the MOF specification model elements are considered to be metaobjects.

model library

A stereotyped package that contains model elements which are intended to be reused by other packages. A model library differs from a profile in that a model library does not extend the metamodel using stereotypes and tagged definitions. A model library is analogous to a class library in some programming languages.

modeling time

Refers to something that occurs during a modeling phase of the software development process. It includes analysis time and design time. Usage note: When discussing object systems, it is often important to distinguish between modeling-time and run-time concerns.

See also: analysis time, design time

Contrast: run time

module

A software unit of storage and manipulation. Modules include source code modules, binary code modules, and executable code modules.

See also: component

multiple classification

A semantic variation of generalization in which an object may belong directly to more than one classifier.

See also: static classification, dynamic classification

multiple inheritance

A semantic variation of generalization in which a type may have more than one supertype.

Contrast: single inheritance

multiplicity

A specification of the range of allowable cardinalities that a set may assume. Multiplicity specifications may be given for roles within associations, parts within composites, repetitions, and other purposes. Essentially a multiplicity is a (possibly infinite) subset of the non-negative integers.

Contrast: cardinality

multi-valued [MOF]

A model element with multiplicity defined whose Multiplicity Type:: upper attribute is set to a number greater than one. The term multi-valued does not pertain to the number of values held by an attribute, parameter, etc. at any point in time.

Contrast: single-valued

13.13 N (Glossary)

~N~



name

A string used to identify a model element.

namespace

A part of the model in which the names may be defined and used. Within a namespace, each name has a unique meaning.

See also: name

n-ary association

An association among three or more classes. Each instance of the association is an n-tuple of values from the respective classes.

Contrast: binary association

nesting

The nesting connector is an alternative membership notation used to indicate nested members within an element, for example, a package which has nested members. The nested members of a package could also be shown inside the packaged rather than linked by the nesting connection.

node

A node is classifier that represents a run-time computational resource, which generally has at least a memory and often processing capability. Run-time objects and components may reside on nodes.

13.14 O (Glossary)

~O~



object

An entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines. An object is an instance of a class.

See also: class, instance

object diagram

A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a collaboration diagram.

See also: class diagram, collaboration diagram

object flow

An Object Flow is a sub type of the State Flow or Transition. It implies the passing of an object instance between elements at run-time.

object flow state

A state in an activity graph that represents the passing of an object from the output of actions in one state to the input of actions in another state.

object lifeline

A line in a sequence diagram that represents the existence of an object over a period of time.

See also: sequence diagram

Object Management Group

The standards body responsible for the UML specification and management. Their website is www.omg.org - follow the links to the UML pages.

object toolbar

The main toolbar running down the centre of EA from which you can select model elements to insert into diagrams. This is also known as the UML Toolbox and the Toolbox.

occurrence

An occurrence relationship indicates that a collaboration represents a classifier. An occurrence connector is drawn from the collaboration to the classifier.

operation

A service that can be requested from an object to effect behavior. An operation has a signature, which may restrict the actual parameters that are possible.

13.15 P (Glossary)

~P~

**package**

1. A package is a namespace as well as an element that can be contained in other package's namespaces. Packages can own or merge with other packages, and its elements can be imported into a package's namespace.
2. A logical container of model elements. Groups elements and may also contain other packages.

The OMG UML specifications states:

"A package is a grouping of model elements. Packages themselves may be nested within other packages. A package may contain subordinate packages as well as other kinds of model elements. All kinds of UML model elements can be organized into packages."

Note that packages own model elements and are the basis for configuration control, storage, and access control. Each element can be directly owned by a single package, so the package hierarchy is a strict tree. However, packages can reference other packages, modeled by using one of the stereotypes «import» and «access» of Permission dependency, so the usage network is a graph. Other kinds of dependencies between packages usually imply that one or more dependencies among the elements exists.

A package is shown as a large rectangle with a small rectangle (a "tab") attached to the left side of the top of the large rectangle. It is the common folder icon.

package diagram

Package diagrams are used to reflect the organization of packages and their elements, and provide a visualization of their corresponding namespaces.

package import

A package import relationship is drawn from a source package to a package whose contents will be imported. Private members of a target package cannot be imported.

package merge

A package merge indicates a relationship between two packages whereby the contents of the target package are merged with those of the source package. Private contents of a target package are not merged.

parameter

The specification of a variable that can be changed, passed, or returned. A parameter may include a name, type, and direction. Parameters are used for operations, messages, and events.

Synonym: formal parameter

Contrast: argument

parameterized element

The descriptor for a class with one or more unbound parameters.

Synonym: template, parameterized class

parent

In a generalization relationship, the generalization of another element, the child.

See also: subclass, subtype

Contrast: child

part

Parts are run-time instances of classes or interfaces.

participate

The connection of a model element to a relationship or to a reified relationship. For example, a class participates in an association, an actor participates in a use case.

partition

1. activity graphs: A portion of an activity graphs that organizes the responsibilities for actions.

See also: swim lane

2. architecture: A set of related classifiers or packages at the same level of abstraction or across layers in a layered architecture. A partition represents a vertical slice through an architecture, whereas a layer represents a horizontal slice.

Contrast: layer

pattern

A template collaboration.

persistent object

An object that exists after the process or thread that created it has ceased to exist.

physical system

1. The subject of a model.
2. A collection of connected physical units, which can include software, hardware and people, that are organized to accomplish a specific purpose. A physical system can be described by one or more

models, possibly from different viewpoints.

Contrast: system

port

Ports define the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted by using the "Expose Interface" toolbox element.

postcondition

A constraint that must be true at the completion of an operation.

precondition

A constraint that must be true when an operation is invoked.

primitive type

A pre-defined basic datatype without any substructure, such as an integer or a string.

process

1. A heavyweight unit of concurrency and execution in an operating system.

Contrast: thread, which includes heavyweight and lightweight processes. If necessary, an implementation distinction can be made using stereotypes.

2. A software development process - the steps and guidelines by which to develop a system.
3. To execute an algorithm or otherwise handle something dynamically.

profile

A profile is a stereotyped package that contains model elements which have been customized for a specific domain or purpose using extension mechanisms, such as stereotypes, tagged definitions and constraints. A profile may also specify model libraries on which it depends and the metamodel subset that it extends.

project browser

The workspace window (top left) where the model contents are displayed in 'tree' format. Displays packages, diagrams, model elements, etc.

projection

A mapping from a set to a subset of it.

property

A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are predefined in the UML; others may be user defined.

See also: tagged value

pseudo-state

A vertex in a state machine that has the form of a state, but doesn't behave as a state. Pseudo-states include initial and history vertices.

published model [MOF]

A model that has been frozen, and becomes available for instantiating repositories and for the support in

defining other models. A frozen model's model elements cannot be hanged.

13.16 Q (Glossary)

~Q~



qualifier

An association attribute or tuple of attributes whose values partition the set of objects related to an object across an association.

13.17 R (Glossary)

~R~



realize

A source object realizes the destination object. Realize is used to express traceability and completeness in the model – a business process or requirement is realized by one or more use cases which are in turn realized by some classes which in turn are realized by a component, etc.

receive [a message]

The handling of a stimulus passed from a sender instance.

See also: sender, receiver

receive

A Receive element is used to define the acceptance or receipt of a request. Movement on to next action does occur until it has received what is defined.

receiver [object]

The object handling a stimulus passed from a sender object.

Contrast: sender

reception

A declaration that a classifier is prepared to react to the receipt of a signal.

recursion

A recursion is a type of message used in sequence diagrams to indicate a recursive function.

reference

1. A denotation of a model element.
2. A named slot within a classifier that facilitates navigation to other classifiers.

Synonym: pointer

region

UML 2 supports both expansion regions and interruptible activity regions. An Expansion Region defines the bounds of an region consisting of one or more sets of input collections, where an input collection is a

set of elements of the same type. An interruptible region contains activity nodes - when a token leaves an interruptible region, this terminates all of the regions tokens and behaviors.

refinement

A relationship that represents a fuller specification of something that has already been specified at a certain level of detail. For example, a design class is a refinement of an analysis class.

relationship

A semantic connection among model elements. Examples of relationships include associations and generalizations.

repository

A facility for storing object models, interfaces, and implementations.

represents

The Represents connector indicates a collaboration is used in a classifier. The connector is drawn from the collaboration to its owning classifier.

requirement

A desired feature, property, or behavior of a system.

responsibility

A contract or obligation of a classifier.

reuse

The use of a pre-existing artifact.

reverse engineering

The process of importing source code into the model as standard UML model elements (classes, attributes, operations, etc.).

rich text format

A standard mark-up language for creating word processor documents, frequently associated with Microsoft Word.

robustness diagram

Enterprise Architect supports business process modeling extensions from the UML business process model profile. Robustness diagrams are used in the Iconix Process - you can read more about this at www.sparxsystems.com.au/iconix/iconixsw.htm.

role

1. The named detail and rules associated with one end of an association. May indicate name, constraints, multiplicity and collection details.
2. The named specific behavior of an entity participating in a particular context. A role may be static (e.g., an association end) or dynamic (eg. a collaboration role).

role binding

Role Binding is the mapping between a collaboration occurrence's internal roles and the respective parts needed to implement a specific situation. The associated parts can have properties defined to enable the binding to occur, and the collaboration to take place.

run time

The period of time during which a computer program executes.

Contrast: modeling time

13.18 S (Glossary)

~S~

**scenario**

A specific sequence of actions that illustrates behaviors. A scenario may be used to illustrate an interaction or the execution of a use case instance.

See also: interaction.

scenario

A sequence of operations carried out in some order to produce a known result. May apply to use cases where it is the equivalent of a sequence diagram, or to other objects to describe how they are used at run-time.

schema [MOF]

In the context of the MOF, a schema is analogous to a package which is a container of model elements. Schema corresponds to an MOF package.

Contrast: metamodel, package

self-message

A self-message reflects a new process or method invoked within the calling lifeline's operation. It is a specification of a message.

semantic variation point

A point of variation in the semantics of a metamodel. It provides an intentional degree of freedom for the interpretation of the metamodel semantics.

send [a message]

The passing of a stimulus from a sender instance to a receiver instance.

See also: sender, receiver

sender [object]

The object passing a stimulus to a receiver object.

Contrast: receiver

sequence diagram

A diagram that shows object interactions arranged in time sequence. In particular, it shows the objects participating in the interaction and the sequence of messages exchanged. Unlike a collaboration diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and collaboration diagrams express similar information, but show it in different ways.

See also: collaboration diagram

signal

The specification of an asynchronous stimulus communicated between instances. Signals may have parameters.

signature

The name and parameters of a behavioral feature. A signature may include an optional returned parameter.

single inheritance

A semantic variation of generalization in which a type may have only one supertype.

Synonym: multiple inheritance [OMA]

Contrast: multiple inheritance

single valued [MOF]

A model element with multiplicity defined is single valued when its Multiplicity Type: upper attribute is set to one. The term single-valued does not pertain to the number of values held by an attribute, parameter, etc., at any point in time, since a single-valued attribute (for instance, with a multiplicity lower bound of zero) may have no value.

Contrast: multi-valued

specification

A declarative description of what something is or does.

Contrast: implementation

state

A condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.

Contrast: state [OMA]

state invariant

A State Invariant is a condition applied to a lifeline, which must be fulfilled for the lifeline to exist.

state machine

A behavior that specifies the sequences of states that an object or an interaction goes through during its life in response to events, together with its responses and actions.

state machine diagram

A State Machine diagram illustrates how an element, often a class, can move between states classifying its behavior, according to transition triggers, constraining guards, and other aspects of state machine diagrams that depict and explain movement and behavior.

state chart

diagram A diagram that shows a state machine.

See also: state machine

state continuation

The State/Continuation symbol serves two different purposes for interaction diagrams, as state invariants

and as continuations. A State Invariant is a condition applied to a lifeline, which must be fulfilled for the lifeline to exist. A Continuation is used in seq and alt combined fragments, to indicate the branches of continuation an operand follows.

state lifeline

A State Lifeline follows discrete transitions between states, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations.

static classification

A semantic variation of generalization in which an object may not change classifier.

Contrast: dynamic classification

stereotype

A new type of modeling element that extends the semantics of the metamodel. Stereotypes must be based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user defined. Stereotypes are one of three extensibility mechanisms in UML.

See also: constraint, tagged value

stimulus

The passing of information from one instance to another, such as raising a signal or invoking an operation. The receipt of a signal is normally considered an event.

See also: message

string

A sequence of text characters. The details of string representation depend on implementation, and may include character sets that support international characters and graphics.

structural diagram

Structural diagrams depict the structural elements composing a system or function. These diagrams can reflect the static relationships of a structure, as do class or package diagrams, or run-time architectures, such as object or composite structure diagrams. Structural diagrams include Class diagrams, Composite Structure diagrams, Component diagrams, Deployment diagrams, Object diagrams and Package diagrams.

structural feature

A static feature of a model element, such as an attribute.

structural model aspect

A model aspect that emphasizes the structure of the objects in a system, including their types, classes, relationships, attributes, and operations.

subactivity state

A state in an activity graph that represents the execution of a non-atomic sequence of steps that has some duration.

subclass

In a generalization relationship, the specialization of another class; the superclass.

See also: generalization

Contrast: superclass

submachine state

A state in a state machine which is equivalent to a composite state but its contents is described by another state machine.

subpackage

A package that is contained in another package.

substate

A state that is part of a composite state.

See also: concurrent state, disjoint state

subsystem

A grouping of model elements that represents a behavioral unit in a physical system. A subsystem offers interfaces and has operations. In addition, the model elements of a subsystem can be partitioned into specification and realization elements.

See also: package, physical system

subtype

In a generalization relationship, the specialization of another type; the supertype.

See also: generalization

Contrast: supertype

superclass

In a generalization relationship, the generalization of another class; the subclass.

See also: generalization

Contrast: subclass

supertype

In a generalization relationship, the generalization of another type; the subtype.

See also: generalization

Contrast: subtype

supplier

A classifier that provides services that can be invoked by others.

Contrast: client

swimlane

A partition on a activity diagram for organizing the responsibilities for actions. Swimlanes typically correspond to organizational units in a business model.

See also: partition

synch

A synch state is useful for indicating concurrent paths of a state machine will be synchronized. After bringing the paths to a synch state, the emerging transition will indicate unison.

synchronize code

The process of importing and exporting code changes to ensure the model and source code match

system

A top-level subsystem in a model.

Contrast: physical system

system boundary

A System Boundary element is used to delineate a particular part of the system. For example in the diagram below, the actor is outside the system and the use case within.

13.19 T (Glossary)

~T~**table**

A relational table (composed of columns).

tagged value

The explicit definition of a property as a name-value pair. In a tagged value, the name is referred as the tag. Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML.

See also: constraint, stereotype

template

Synonym: parameterized element

terminate

The terminate pseudostate indicates that upon entry of its pseudostate, the state machine's execution will end.

thread [of control]

A single path of execution through a program, a dynamic model, or some other representation of control flow. Also, a stereotype for the implementation of an active object as lightweight process.

See also: process

time event

An event that denotes the time elapsed since the current state was entered.

See also: event

time expression

An expression that resolves to an absolute or relative value of time.

timing diagram

The Timing diagram defines the behavior of different objects within a time-scale, with visual depictions of those objects changing state and interacting over time.

toolbox

The main toolbar running down the centre of EA from which you can select model elements to insert into diagrams. This is also known as the UML Toolbox and the Object Toolbar.

top level

A stereotype of package denoting the top-most package in a containment hierarchy. The topLevel stereotype defines the outer limit for looking up names, as namespaces "see" outwards. For example, opTopLevelSubsystem represents the top of the subsystem containment hierarchy.

trace

A dependency that indicates a historical or process relationship between two elements that represent the same concept without specific rules for deriving one from the other.

transient object

An object that exists only during the execution of the process or thread that created it.

transition

A relationship between two states indicating that an object in the first state will perform certain specified actions and enter the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to fire.

type

type A stereotyped class that specifies a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects. A type may not contain any methods, maintain its own thread of control, or be nested. However, it may have attributes and associations. Although an object may have at most one implementation class, it may conform to multiple different types.

See also: implementation class

Contrast: interface

type expression

An expression that evaluates to a reference to one or more types.

13.20 U (Glossary)

~U~

**UML**

The Unified Modeling Language, a notation and specification for modeling software systems in an Object-Oriented manner. You can read more about UML at the OMG home page or at our UML Tutorial

UML diagrams

UML diagrams are used to model different aspects of the system under development. They include various elements and connections, all of which have their own meanings and purposes. UML 2.0 includes 13 diagrams: Use Case diagram, Activity diagram, State Machine diagram, Timing diagram, Sequence diagram, Interaction Overview diagram, Communication diagram, Package diagram, Class diagram,

Object diagram, Composite Structure diagram, Component diagram and Deployment diagram.

UML toolbox

The main toolbar running down the centre of EA from which you can select model elements to insert into diagrams. This is also known as the Toolbox and the Object Toolbar.

uninterpreted

A placeholder for a type or types whose implementation is not specified by the UML. Every uninterpreted value has a corresponding string representation.

See also: any [CORBA]

usage

A dependency in which one element (the client) requires the presence of another element (the supplier) for its correct functioning or implementation.

use

A Use link indicates that one element requires another to perform some interaction. The Usage relationship does not specify how the target supplier is used, other than that the source client uses it in definition or implementation.

use case [class]

A *Use Case* is a UML model element that describes how a user of the proposed system will interact with the system to perform a discrete unit of work. It describes and signifies a single interaction over time that has meaning for the end user (person, machine or other system), and is required to leave the system in a complete state: either the interaction completed or was rolled back to the initial state.

See also: use case instance

use case diagram

A *Use Case diagram* captures Use Cases and actor interactions. It describes the functional requirements of the system, the manner that outside things (actors) interact at the system boundary, and the response of the system.

use case estimation

The technique of estimating project size and complexity based on the number of use cases and their difficulty.

use case instance

The performance of a sequence of actions being specified in a use case. An instance of a use case.

See also: use case class

use case model

A model that describes a system's functional requirements in terms of use cases.

utility

A stereotype that groups global variables and procedures in the form of a class declaration. The utility attributes and operations become global variables and global procedures, respectively. A utility is not a fundamental modeling construct, but a programming convenience.



13.21 V (Glossary)

~V~

value

An element of a type domain.

value lifeline

The Value lifeline shows the lifeline's state across the diagram, within parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

vertex

A source or a target for a transition in a state machine. A vertex can be either a state or a pseudo-state.

See also: state, pseudo-state

view

A projection of a model, which is seen from a given perspective or vantage point and omits entities that are not relevant to this perspective.

view element

A view element is a textual and/or graphical projection of a collection of model elements.

view projection

A projection of model elements onto view elements. A view projection provides a location and a style for each view element.

visibility

An enumeration whose value (public, protected, package or private) denotes how the model element to which it refers may be seen outside its enclosing namespace.

Visual Basic

A rapid application development programming language. Windows only scripting language based on COM.

Part

XIV

14 Acknowledgements

Some parts of this application include code originally written by various authors and modified for use in Enterprise Architect.

Marquet Mike

Print listview contents
mike.marquet@altavista.net

Davide Pizzolato

CXImage Library
© 7-Aug-2001
ing.davide.pizzolato@libero.it

Also, many thanks to all those who have made suggestions, reported bugs, offered feedback and helped with the beta testing of Enterprise Architect. Their help has been invaluableA0CB368DDF6FC830151104335A0C21A350D14D3269FEFAD0DDB315145C31A7225F

Index

- 2 -

2005 Style 45

- A -

About EA 34
 Abstract 244
 Acceptance Testing 784
 Acknowledgements 1144
 Action 264
 Action - Local Pre/Post Conditions 270
 Action Expansion Node 266
 Action Notation 264
 Action Pin 268
 Active Classes 281
 Active state configuration 328
 Activity 271
 Activity Diagram 217
 Activity Diagrams - Object Flows 387
 Activity Final 301
 Activity Group 141
 Activity Notation 273
 Activity Parameter Nodes 274
 Activity Partition 276
 Actor 277
 Add 812
 Delete ,Modify Glossary Entries 812
 Delete ,Modify Glossary Entries - Glossary Dialog 812
 Delete ,Modify Glossary Entries - Glossary Tab 813
 Add a Diagram 150
 Add a Pattern to Diagram 581
 Add a Task 801
 Add a UML Pattern to Diagram 581
 Add Additional Views 675
 Add an Instance Variable 316
 Add an Object 163
 Add Code Modules in MDG Technology Wizard 541
 Add Diagram Properties Note 411
 Add Enumeration tags to Stereotypes 557
 Add Expansion Region 300

Add Images in MDG Technology Wizard 542
 Add Ins 1080
 Add Interruptible Activity Region 311
 Add Key 30
 Add MDG Technologies to UML Toolbox 546
 Add MS Word Table of Contents 943
 Add MS Word Table of Figures 944
 Add Note to Connection 516
 Add Note to Link 516
 Add or Delete Line Points 507
 Add Packages to Your Document Object 964
 Add Pattern in MDG Technology Wizard 539
 Add Previously defined Version Control Configuration 712
 Add Profile in MDG Technology Wizard 538
 Add Properties Note 411
 Add Tagged Value types in MDG Technology Wizard 540
 Add Tagged Values 493
 Add UML Profile Connector to Diagram 565
 Add-In Broadcast Events 1091, 1092, 1093, 1094, 1095, 1096
 Add-In Events 1086, 1087, 1088, 1089, 1090
 Add-in MDG BuildProject 1098
 Add-In Tasks 1081
 Adding a Port to an Element 322
 Adding New Code Sections to Existing Features 883
 Adding New Features and Elements 884
 Adding New Stereotyped Templates 881
 Adding Packages 147
 Adding Use Case Attributes and Operations to Activity Diagrams 458
 Add-Ins 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096
 Add-Ins - Creating 1082
 Add-Ins - Defining Menu Items 1082
 Add-Ins - Deploying 1083
 Add-Ins - EA Events 1086
 Add-Ins - Overview 1081
 Add-Ins - Tricks and Traps 1085
 Add-Ins Broadcast Events 1091
 Add-Ins EA_Connect 1086
 Add-Ins EA_Disconnect 1087
 Add-Ins EA_FileOpen 1091
 Add-Ins EA_GetMenuItems 1088
 Add-Ins EA_GetMenuState 1089
 Add-Ins EA_MenuClick 1088
 Add-Ins EA_OnDeleteTechnology 1095

- Add-Ins EA_OnImportTechnology 1096
- Add-Ins EA_OnPostNewConnector 1094
- Add-Ins EA_OnPreNewConnector 1093
- Add-Ins EA_OnPreNewElement 1092
- Add-Ins EA_PostNewElement 1094
- Add-Ins EA_ShowHelp 1090
- Advanced Properties Dialog 429
- Advanced Settings 484
- Advanced Tag Management 491
- Aggregate 363
- Aggregation Link Form 363
- Alias 1020
- Align Elements 447
- Align Objects 447
- All Permissions 726
- Allow Duplicate Tagged Values 128
- Allow Duplicate Tags 128
- Alternate Image 419
- Analysis 314
- Analysis Diagram 255
- Analysis Group 135
- Analysis Stereotypes 342
- Appearance 453
- Appearance - Element Context Menu 432
- Appearance Menu Section 506
- Application Workspace 41
- Apply a Filter 931
- Apply a User Lock 734
- Applying a Stereotype 371
- Argument 473
- Arranging Connections 509
- Arranging Windows and Menus 42
- Artifact 278
- Assembly 364
- Assign People to Defects or Changes 799
- Assigning Information to tagged Values 127
- Assigning Tagged Values to an Item 125
- Associate 364
- Associated Files 495
- Association 365, 366
- Association Class 353, 366
- Association Details 520
- Association Properties 520
- Association Type - Change 510
- Attach a Note to a Link 516
- Attribute 1020
- Attribute Constraints 462
- Attribute Detail 461
- Attribute Tagged Values 463
- Attributes 458
- Attributes and Operations 457
- Attributes Main Page 459
- Authors 746
- Auto Counters 166
- Auto Element Naming 166
- Auto Routing Connector Style 507
- Autohide Windows 47
- Automation and Scripting 970
- Automation Interface 970, 971, 974, 975, 976, 977, 980, 981, 982, 990, 991, 992, 993, 994, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1006, 1007, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1022, 1023, 1024, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1035, 1037, 1038, 1039, 1042, 1043, 1044, 1051, 1052, 1053, 1054, 1055, 1058, 1059, 1060
- Automation Interface 981, 1055
- Automation Interface - App 980
- Automation Interface - Available Resources 976
- Automation Interface - Call from EA 975
- Automation Interface - Code Examples 1051
 - Add a Connector 1053
 - Add and Manage Diagrams 1054
 - Add and Manage Elements 1053
 - Add and Manage Packages 1052
 - Adding and Deleting Attributes and Methods 1055
 - Element Extras 1055
 - Iterate Through an EAP File 1052
 - Open the Repository 1051
 - RepositoryExtras 1058
 - Stereotypes 1059
 - Working with Attributes 1060
 - Working with Methods 1060
- Automation Interface - Connector 1031
 - Connector 1032
 - Connector Constraints 1035
 - Connector End 1035
 - Connector Tag 1037
 - Role Tag 1037
- Automation Interface - Diagram 1038
 - Diagram 1039
 - Diagram Links 1042
 - Diagram Objects 1043
- Automation Interface - Element 1004
 - Constraint 1006
 - Effort 1006
 - Element 1007

Automation Interface - Element 1004
 File 1012
 Issue 1013
 Metric 1014
 Requirement 1014
 Resource 1015
 Risk 1016
 Scenario 1017
 Tagged Value 1018
 Test 1019
 Automation Interface - Element Features 1019
 Attribute 1020
 Attribute Constraint 1022
 Attribute Tag 1023
 Custom Properties 1031
 Embedded Elements 1029
 Method 1024
 Method Constraint 1026
 Method Tag 1027
 Parameter 1027
 Partitions 1028
 Transitions 1030
 Automation Interface - Examples and Tips 974
 Automation Interface - Model and Package 977
 Automation Interface - Project Interface 1044
 Project 1044
 Automation Interface - Reference 976
 Automation Interface - Repository 983, 985
 Author 990
 Client 991
 Collection 992
 Datatype 993
 EventProperties 1002
 EventProperty 1002
 Package 994
 Project Issues 996
 Project Resource 997
 Property Type 1003
 Reference 998
 Stereotype 999
 Task 1000
 Term 1001
 Automation Interface - Set Up VB 972
 Automation Interface -Enumerations 981
 Autosize Elements 411
 Available Help File Formats 11

- B -

Basic Elements 263
 Basic UML Elements 263
 Batch Generate Selected Element(s) 54
 Batch Synchronize Selected Element(s) 54
 Batch XMI Export 689
 Batch XMI Import 690
 Behavioral Diagrams 215
 Bitmap Images in Diagrams 419
 Bookmarks 819
 Boundary 335, 343
 Boundary - Create 344
 Boundary Object Settings 498
 Boundary Properties 498
 Business Modeling Stereotypes 1 344
 Business Modeling Stereotypes 2 345
 Business Modelling 602
 Business Processes 602

- C -

C# 825
 C# Options 846
 C++ 825
 C++ Options 846
 Call 346
 Change Aggregation Link Form 363
 Change Connector Type 510
 Change Diagram Type 406
 Change Element Type 445
 Change the Source or Target Object 514
 Change Type 445
 Changes 793, 797
 Changes and Defects 796
 Changing Object Type 445
 Checking Data Integrity 654
 Checking In and Out 713
 Choice 278
 Class 244, 280
 Class Diagram 244
 Class Group 136
 Classes - Parameterized (Templates) 281
 Clients 747
 Close All Diagrams 163
 Close All Except Current Diagram 162

- Close Current Diagram 161
- Code - Reverse Engineer 821
- Code Engineering 821
- Code Engineering - Element Context Menu 432
- Code Engineering Settings 835
- Code Engineering Sub-Menu 99
- Code Generation 827
- Code Generation - Attributes 838
- Code Generation Constructor/Destructor Options 837
- Code Generation Dialog 829
- Code Generation Options 205, 836
- Code Generation Toolbar 85
- Code Page for Source Editing 839
- Code Templates 855
- Code Templates - Base Templates 856
- Code Templates - Custom Templates 859
- Code Templates - Execution 861
- Code Templates - Literal Text 862
- Code Templates - Macros 862
- Code Templates - Overview 856
- Code Templates - Variables 877
- Code Templates Syntax 862
- CodeTemplate Framework 855
- Collaboration 283
- Collaboration Color 201
- Collaboration Diagram 236
- Collaboration Diagrams in Color 238
- Collaboration Group 138
- Collaboration Messages 378
- Collaboration Occurrence 285
- Collisions 737
- Color Coded External Requirments 586
- Combined Fragment 286
- Combined Fragment - Creation 288
- Common Actions - Element Context Menu 433
- Common Actions Menu Section 505
- Common Connection Tasks 507
- Common Tasks 435
- Common Usage 380
- Communication 201
- Communication Diagram 236
- Communication Diagrams in Color 238
- Communication Group 138
- Communication Message 377
- Communication Message Creation 377
- Communication Message Properties 377, 378
- Compact a Project 673
- Compare Data 658
- Comparing Models 658
- Compartments 499
- Component 291
- Component Diagram 252
- Component Group 141
- Compose 368
- Composite Elements 346
- Composite Group 137
- Composite state 328
- Composite State Regions 222
- Composite Structure Diagram 248
- Composition 591
- Configuration Menu 79
- Configure Controlled Packages with XMI 685
- Configure Default Appearance 59, 453
- Configure Local Options 192
- Configure Package for Version Control 710
- Connect to a MSDE Server Data Repository 639
- Connect to a MySQL Data Repository 625
- Connect to a Oracle9i Data Repository 630
- Connect to a PostgreSQL Data Repository 633
- Connect to a SQL Server Data Repository 627
- Connect to an Adaptive Server Anywhere Data Repository 636
- Connecting Elements 507
- Connecting Objects 164, 507
- Connecting to a Data Repository 624
- Connecting to the Interface 971
- Connection - Add a Note 516
- Connection Constraints 522
- Connection Context Menu 504
- Connection Details 520
- Connection Properties 520
- Connections 360
- Connector 368
- Connector Styles 507
- Connector Type - Change 510
- Connector Visibility 512, 515
- ConstLayoutStyles Enum 981
- Constraint Compartment 501
- Constraint Status Types 751
- Constraint Types 752
- Constraints 487, 556
- Contents Sub-Menu 100
- Context Element Highlight 160
- Continuation 331
- Control 347, 698

- Control Element - Create 347
- Control Flow 369
- Control Macros 874
- Controlled Package 687
- Controlled Packages Menu -XMI 684
- Controlled Packages with XMI 683
- Convert Linked Element to Local Copy 405
- Copy Diagram Element 404
- Copy Diagram Image to Clipboard 406
- Copy Image to Clipboard 406
- Copy Image to Disk 406
- Copy Packages from One Project to Another 659
- Copy RTF Bookmark to Clipboard 98, 101, 102
- Copying Attributes and Operations Between Elements 442
- Copyright Notice 5
- Correcting Words 743
- Corrupt .EAP file 673
- Create a Boundary 344
- Create a Combined Fragment 288
- Create a Control Element 347
- Create a Document Object 964
- Create a Model 611
- Create a New Adaptive Server Anywhere Repository 647
- Create a New Model File 611
- Create a New MSDE Server Repository 651
- Create a New MySQL Repository 639
- Create a New Oracle9i Server Repository 644
- Create a New PostgreSQL Repository 645
- Create a New SQL Server Repository 642
- Create a Rich Text Document 928
- Create a Table 898
- Create a Timing Message 382
- Create a Virtual Report 964
- Create an Entity 349
- Create and Open Model Files 609
- Create and Open Model Files Discussion 651
- Create Columns 905
- Create Link 518
- Create MDG Technologies 534
- Create Messages 377
- Create UML Patterns 577
- Creating a Custom View with EA Add-Ins 1097
- Creating a HTML Report 957
- Creating a Repository 639
- Creating Add-Ins 1082
- Creating Documents 927
- Creating Notes 451
- Creating Objects 435
- Creating Patterns 577
- Creating Properties 464
- Creating Replicas 738
- Creating Requirements 584
- Creating Templates For Custom Languages 882
- Creating Text 451
- Creating UML Profiles 551
- Cross References 440
- CSV Export 694
- CSV Import 696
- CSV Import and Export 692
- CSV Specifications 692
- CTF 855
- Current Connector Toolbar 89
- Current Element Toolbar 89
- Custom Connector Style 507
- Custom Diagram 256
- Custom Group 143
- Custom Layouts 205
- Custom Tagged Values 124
- Custom Toolbars 66
- Custom Tools 70
- Custom Views 1096
- Customize Commands 66
- Customize Keyboard Shortcuts 75
- Customize Menu 78
- Customize Options 79
- Customize Toolbars 91
- Customize Visible Elements 450
- Customize Window 65
- CVS 687, 697, 698, 699, 700, 710, 712
- CVS Remote Repository 707

- D -

- Data Compare 658
- Data Integrity 654
- Data Model Diagram 897
- Data Modeling 897
- Data Transfer 656
- Data Types 758
- Database Schema 257
- Datastore 291
- Datatype 916
- Datatypes - DBMS 919
- DBMS 916

- DBMS Conversion Procedure 916
- DBMS Conversion Procedure Package 917
- DBMS Datatypes 919
- DDL 913
- Decision 292
- Default Diagram 399
- Default Editor Options 837
- Default Element Template 168
- Default Element Templates 168
- Default Hours 767
- Default Model Diagram 399
- Default Tools Toolbar 84
- Defaults and User Settings 192
- Defects 793
- Defects (Issues) 796
- Define a Run time Variable 316
- Define Stereotype Constraints 556
- Defining Menu Items 1082
- Delegate 370
- Delete 812
 - Add ,Modify Glossary Entries 812
 - Add ,Modify Glossary Entries - Glossary Dialog 812
 - Add ,Modify Glossary Entries - Glossary Tab 813
- Delete a Package 149
- Delete a Package From Document Object 966
- Delete a Task 801
- Delete an Instance Variable 318
- Delete Views 678
- Deleting Connections 511
- Deleting Diagrams 402
- Deleting Elements 449
- Deleting Objects 449
- Delphi 826
- Delphi Options 847
- Delphi Properties 848
- Denote Lifecycle of an Element 228
- Dependency 371
- Dependency Report 592, 949
- Deployment 372
- Deployment Diagram 253
- Deployment Group 142
- Deployment Spec 293
- Deprecated Export Profile 569
- Deprecated Load a UML Profile into EA 567
- Deprecated Profile Tasks 567
- Deprecated Save a Diagram as a UML Profile 567
- Deprecated UML Profile 567
- Design Master 653, 737
- Design Masters 738
- Destination Role 525
- Diagonal Representation 365
- Diagram Behavior 197
- Diagram Context Menu 398
- Diagram Context Menu - Project Browser 102
- Diagram Gate 294
- Diagram Menu 57
- Diagram Notes 411
- Diagram Only RTF Report 936
- Diagram Properties 402
- Diagram Properties Note 411
- Diagram Settings 196
- Diagram Tabs 92
- Diagram Tasks 149, 161, 400
- Diagram Toolbar 88
- Diagram View 145
- Differences Between Desktop 6
 - Professional and Corporate Editions 6
- Direct Connector Style 507
- Disconnect Controlled Package 687
- Displaying Inherited Attributes 466
- Displaying Inherited Operations 480
- Distributed Development 719
- Dockable Windows 42, 45, 105
- Dockable Windows 2005 Style 45
- Document a Single Element 929
- Document Object 964
 - Document Object - Adding Packages 964
 - Document Object - Delete a Package 966
 - Document Object - Generate 967
 - Document Object - Rearrange Package Order 965
- Documentation - HTML 957
- Documentation - RTF 927
- Documentation and Web 960
- Documentation Sub-Menu 98
- Documenting Projects 927
- Documents - HTML 927
- Documents - RTF 927
- Drag 411
 - Drag a Package onto a Diagram 148
 - Drop Elements from the Project Browser 411
 - Drop Objects from Tree 411
- Duplicate a Diagram 408

- E -

- EA Add-Ins 1097
- EA Add-Ins Custom View 1096
- EA Features 4
- EA_Connect 1086
- EA_Disconnect 1087
- EA_FileOpen 1091
- EA_GetMenuItems 1088
- EA_GetMenuState 1089
- EA_MenuClick 1088
- EA_OnDeleteTechnology 1095
- EA_OnImportTechnology 1096
- EA_OnPostNewConnector 1094
- EA_OnPreNewConnector 1093
- EA_OnPreNewElement 1092
- EA_PostNewElement 1094
- EA_ShowHelp 1090
- ECF Dialog 765
- Eclipse 855
- Edit Attribute Keyword 176
- Edit Attribute Keyword - Inplace Editing 176
- Edit Attribute Keyword and Operation Scope 175
- Edit Attribute Keyword and Operation Scope- Inplace Editing 175
- Edit Attribute or Operation Stereotype 173
- Edit Attribute or Operation Stereotype - Inplace Editing 173
- Edit Element 170
- Edit Element Name 172
- Edit Element Name Inplace Editing 172
- Edit Menu 50
- Edit Operation Parameter Keyword 177
- Edit Operation Parameter Keyword - Inplace Editing 177
- Edit Parameter Kind 179
- Edit Parameter Kind - Inplace Editing 179
- Effort and Risk Management 770
- Effort Types 775
- Element 184
- Element Appearance 453
- Element Browser 113
- Element Context Menu 427
- Element Context Menu - Appearance 432
- Element Context Menu - Code Engineering 432
- Element Context Menu - Common Actions 433
- Element Context Menu - Embedding and Features 430
- Element Context Menu - Multiple Selection 434
- Element Context Menu - Project Browser 101
- Element Context Menu - Properties 429
- Element Context Menu - Type 432
- Element Editing 170
- Element Lifecycle 228
- Element Maintenance 170
- Element Menu 59
- Element Properties 482, 799
- Element Relationship Matrix 593
- Element Tasks 163, 170
- Element Template 168
- Element Template Package 168
- Element Toolbar 87
- Elements 259
- Embedded Elements 430
- Embedding and Features - Element Context Menu 430
- Enabling Security 722
- Endpoint 295
- Enterprise Architect Connections 360
- Enterprise Architect Glossary 1112
- Entity 348
- Entity - Create 349
- Entry Point 296
- Enumeration 557
- EnumRelationSetType Enum 981
- Environment Complexity Factors 765
- Estimating Project Size 766
- Estimation 763
- Estimation Hour Rate 767
- Event 349
- Events 349
- Example UML Profile Diagram 564
- Example UML Profile File 575
- Exception 297
- Exclude Elements in RTF 932
- Exclude Objects in RTF 932
- exclusion 927
- Exit Point 300
- Expansion Region 298
- Expansion Region - Add 300
- Export Profile 558
- Export Reference Data 760
- Export to XMI 680
- Exporting to CSV 599

Expose Interface 373
Extend 373
Extended Class 256
Extended Stereotypes 342
Extension Points 338
External Editor 59
External Requirements 584
External Responsibilities 486

- F -

Favorites 110
Field Substitution Macros 863
File Menu 49
Find 166
Finding Your License Information 34
Flow Final 302
Foreign Keys 908
Fork 304
Fork/Join 302
Format Toolbar 90
Fragment 286
Free Sorting 103
Function Macros 871

- G -

general add-in methods 1082
General Code Options 835
General Properties 483
General Settings 193, 483
General Types 750
Generalize 374
Generate a Group of Classes 830
Generate a Package 831
Generate a Single Class 828
Generate Code 827
Generate DDL 913
Generate DDL for a Package 915
Generate Glossary Report 814
Generate Package Dialog 833
Generate Package Source Code 831
Generate RTF Report 935
Generate Source Code 827
Generate Virtual Documents 967
Generate XML schema 884
Get/Set Project Custom Colors 455

Getting Started 28
Glossary 810
Glossary - A 1112
Glossary - B 1114
Glossary - C 1115
Glossary - D 1119
Glossary - E 1121
Glossary - F 1122
Glossary - G 1123
Glossary - H 1124
Glossary - I 1124
Glossary - J 1126
Glossary - L 1126
Glossary - M 1126
Glossary - N 1129
Glossary - O 1129, 1130
Glossary - Q 1133
Glossary - R 1133
Glossary - S 1135
Glossary - T 1139
Glossary - U 1140
Glossary - V 1142
Glossary Dialog 811
Glossary of Terms 1112
Glossary Overview 810
Glossary Report 814
Glossary Report Sample 816
Glossary Tab 811

- H -

Headers and Footers 945
Height 448
Help for Tagged Values 127
Help Menu 82
Hide/Show Connectors 512
Hide/Show Labels 513
Hierarchy Window 116
Highlight Context Element 160
History 306
HTML and the Web 960
HTML Documentation 957
HTML Documents 927
HTML Reports 927, 957
Hyperlinking diagrams 349
Hyperlinks 349

- I -

Image Handling 939
 Image Manager 419
 Images 419
 Implementation 591
 Implementation Report 950
 Implementation Report - Target Types 952
 Import a Directory Structure 824
 Import a Pattern 580
 Import a UML Pattern 580
 Import and Export Reference Data 760
 Import C# 825
 Import C++ 825
 Import Database Schema from ODBC 920
 Import DDL Schema from ODBC 920
 Import Delphi 826
 Import from XML 682
 Import Java 826
 Import MDG Technologies 545
 Import PHP 826
 Import Reference Data 761
 Import Scenario as Test 787
 Import Source Code 822
 Import Test 788
 Import VB.Net 826
 Import Visual Basic 826
 Import/Export Sub-Menu 100
 Imported Class Elements 924
 Importing and Exporting Code Templates 882
 Importing UML Profiles 561
 Inbuilt and Extended Stereotypes 342
 Include 374
 Include or Exclude a Package from RTF Report 937
 Indexes and Triggers 912
 Initial 308
 Insert Attribute or Operation 180
 Insert Attribute or Operation - Inplace Editing 180
 Insert Diagram Properties Note 411
 Insert Element at Cursor 451
 Insert Element at Cursor - Boundary 498
 Insert Item Feature 184
 Insert Maintenance Feature 184
 Insert Maintenance Feature - Inplace Editing 184
 Insert Operation Parameter 182
 Insert Operation Parameter - Inplace Editing 182

Insert Properties Note 411
 Insert Related Elements 433
 Insert Sub-Menu 98
 Insert Testing Feature 188
 Insert Testing Feature - Inplace editing 188
 Installation 28
 Installing Enterprise Architect 28
 Instance 314
 Instance Classifier 315
 Instance Runtime State 318
 Integration Testing 782
 Integrity of Data 654
 Interaction Diagrams 225
 Interaction Group 139
 Interaction Occurrence 308
 Interaction Operators 288
 Interaction Overview Diagram 239
 Interface 309
 Internal Requirements 587
 Interrupt Flow 375
 Interruptible Activity Region - Add 311
 Interruptible Activity Region 310
 Introducing Diagrams 396
 Introducing UML Objects 259
 Introduction 3
 Introduction to Enterprise Architect 3
 Issues - Adding 805
 Deleting ,Modifying 805
 Deleting ,Modifying (Model Issues Tab) 806
 Deleting ,Modifying (Project Issues Dialog) 805
 Issues - Report Sample Output 809
 Issues Report 807
 Issues Report (Model Issues Tab) 807
 Issues Report (Project Issues Dialog) 807

- J -

Java 826
 Java Options 850
 Join 305
 Junction 311

- K -

Keyboard Shortcuts 209

- L -

Label Visibility 513
Language Macros 842
Language Options 845
Language - Spelling 744
Layout a Diagram 155
Layout of Sequence Diagrams 229
Layouts 205
Licence Management 29
License Agreement 8
License Information 34
Life Cycle 793
Lifecycle of an Element 228
Lifeline 313
Lifeline Self-Message Hierarchy 231
Limitations of XMI 683
Line Angles - Tidy 507
Line Points - Add or Delete 507
Line Style 507
Link - Add a Note 516
Link a New Class to an Existing Association 367
Link a Note to an Element 502
Link Class to Association 366
Linking Notes 501
Linking Notes to Internal Documentation 501
Links 489
Load a Package 689
Local Directories 841
Local Options 192
Local Path Dialog 841
Local Paths 840
Local Pre/Post Conditions 270
Locate in Browser - Diagram 57
Locate in Browser - Element 59
Lock Diagram 154
Locking Model Elements 733
Locking Packages 733
Logical Diagram 244
Logical Group 136

- M -

Macros 842
Main Menu 48
Maintaining Users 723

Maintenance 184, 754, 793, 794
Maintenance - Problem Types 754
Maintenance - Testing Types 755
Maintenance Element Properties 794
Maintenance Script 794
Maintenance Support 794
Maintenance Workspace 793
Make Same 59
Making the Output Available on the Web 960
Manage Bookmarks 819
Manage Locks 730
Manage My Profile 39
Manage Views 675
Managing Groups 727
Managing Inherited and Redefined Ports 322
Managing Models 608
Managing Your Locks 736
Manifest 376
Manual Version Control with XMI 691
Mapper 916
Master 737
Matrix 592
Matrix Options 597
MDG 1097
MDG Add-In 1099, 1100, 1101, 1102, 1104, 1105, 1106, 1107, 1108, 1109
MDG Add-Ins 1097, 1098
MDG Events 1098
MDG Link 855
MDG Link Code Engineering 855
MDG Menus Enumeration 981
MDG Technologies 533, 534
MDG Technology 533, 538, 539, 540, 541, 542
MDG_BuildProject 1098
MDG_Connect 1099
MDG_Disconnect 1100
MDG_GetConnectedPackages 1101
MDG_GetProperty 1101
MDG_Merge 1102
MDG_NewClass 1104
MDG_PostGenerate 1105
MDG_PostMerge 1106
MDG_PreGenerate 1106
MDG_PreMerge 1107
MDG_PreReverse 1108
MDG_RunExe 1108
MDG_View 1109
Mentor 11, 207

Message 376, 380
 Message (Communication) 377
 Message (Sequence Diagram) 380
 Message (Timing Diagram) 381
 Message (Timing Diagram) Creation 382
 Message Creation 377
 Message Endpoint 384
 Message Label 385
 Message Scope 526
 Message Sequencing 377, 378
 Message Source and Target 526
 Metric Types 776
 Metrics 772
 Metrics and Estimation 756
 Microsoft Word 939
 Microsoft Word - Special Considerations 943
 Model and Project Issues 802
 Model Authors 746
 Model Context Menu 95
 Model Context Menu - Root Node Package Control Submenu 96
 Model Element Cross References 440
 Model Glossary Overview 810
 Model Glossary Tab 811
 Model Integrity 654
 Model Issues Tab 804
 Model Maintenance 672
 Model Management 608
 Model Task List 800
 Model Task Tab 800
 Models 259
 Modifying Tagged Values 125
 Modify 812

- Add ,Delete Glossary Entries 812
- Add ,Delete Glossary Entries - Glossary Dialog 812
- Add ,Delete Glossary Entries - Glossary Tab 813

 Modify a Task 801
 Modifying Relationships in Matrix 597
 Move Down 103
 Move Internal Responsibility to External Requirement 588
 Move Up 103
 Moving Elements between Packages 445
 Moving Elements in Diagrams 446
 Moving Objects and Packages 165
 Moving Objects between Packages 445
 Moving Objects in Diagrams 446

MS Word tables 946
 Multiple Select 414
 MySQL Data Repository 625
 MySQL ODBC Driver 613
 MySQL Repository 639

- N -

Namespaces 834
 N-Ary Association 353
 Nesting 386
 New Diagrams 401
 New Elements 437
 New File 49
 Node 313
 Note - Add to Link/Connection 516
 Notes Window 110, 451

- O -

Object 314
 Object Appearance 199
 Object Classifiers 496
 Object Connections 489
 Object Diagram 246
 Object Flow 386
 Object Flows in Activity Diagrams 387
 Object Links 489
 Object Properties 482
 Object Runtime State 318
 Object Scenarios 489
 Object Tasks 163
 Object Toolbar 132
 Object TypeEnum 982
 Object Visibility 201
 ObjectType 994
 Occurrence 308, 388
 ODBC 913
 Open a Package 147, 407
 Open a Report in Microsoft Word 939
 Open an Existing Project 610
 Open File 49
 Open Project Dialog 609
 Open Source in External Editor 59
 Open the Relationship Matrix 594
 Opening External Tools 73
 Operation Detail 471

- Operation Parameters 473
- Operation Parameters by Reference 475
- Operation Tagged Values 477
- Operations 468
- Operations Constraints 477
- Operations Main Page 469
- Options - C# 846
- Options - C++ 846
- Options - Code Generation 836
- Options - Code Generation - Attributes 838
- Options - Code Generation Constructor/Destructor 837
- Options - Collaboration Colors 201
- Options - Communication Colors 201
- Options - Default Editors 837
- Options - Delphi 847
- Options - Diagram Settings 196
- Options - General Settings 193
- Options - Java 850
- Options - Local 192
- Options - Object Appearance 199
- Options - Object Visibility 201
- Options - Outlook Toolbar 203
- Options - PHP 851
- Options - Sequence Diagrams 198
- Options - Standard Colors 195
- Options - VB.Net 852
- Options - Visual Basic 851
- Options - Visual Styles 206
- Options - XML Specifications 202
- Oracle9i Data Repository 630
- Oracle9i Server Repository 644
- Order Package Contents 103
- Ordering Enterprise Architect 10
- Other Documents 949
- Other Elements 342, 353
- Other UML Elements 342
- Outlook Toolbar 203
- Override Implementation 478
- Override Parent Operations 478
- Overriding Default Templates 880
- P -**
- Package 319
- Package Contents - Update 833
- Package Context Menu 96
- Package Control Sub-Menu 97
- Package Diagram 242
- Package Import 389
- Package Merge 389
- Package Tasks 146
- Parameter 473
- Parameterized Classes (Templates) 281
- Parents and Interfaces 438
- Part 319
- Partition 320
- Passing Parameters to External Tools 74
- Password Encryption 731
- Paste 414, 416
- Paste Elements 415
- Paste Object as Link 413
- Paste Object as New 413
- Pasting Activities 416
- Pasting Composite Elements 415
- Pasting from the Project Browser 413
- Pasting Multiple Items from the Project Browser 414
- Patches - SQL 656
- Patterns 577
- People 746
- Perform a Data Transfer 657
- Permission List 729
- PHP 826
- PHP Options 851
- Pin End Point 89
- Pin Start Point 89
- Pkg Import 389
- Pkg Merge 389
- Place Related Elements on Current Diagram 416
- Port 321
- Predefined Reference Data 121
- Predefined Tagged Values 120
- Primary Key 907
- Primary Keys 907
- Printing the Matrix 600
- Problem Types 754
- Process 354
- Profile 557
- Profile Constraints 556
- Profile Group 144
- Profile References 571
- Profile Tags 554
- Profiles 601
- Profiles Add Elements 552
- Profiles Add Metaclass 552

- Programming Languages Data Types 758
 Project and Model Issues 802
 Project Browser 93
 Project Clients 747
 Project Explorer 93
 Project Interface - Automation Interfaces 1062
 Project Interface - Enumerating 1065
 Project Interface - Enumerating Diagrams 1066
 Project Interface - Enumerating Elements 1068
 Project Interface - Enumerating Links 1077
 Project Interface - Enumerating Views 1064
 Project Interface - Get Diagram Image 1080
 Project Interface - Get Diagrams 1067
 Project Interface - Get Element 1069
 Project Interface - Get Link 1078
 Project Interface - Opening a Project 1063
 Project Interface - Read Only Interface 1062
 Project Interface (XML) 1062
 Project Issues 802
 Project Management 763
 Project Menu 54
 Project Roles 749
 Project Toolbar 85
 Properties 250, 319
 Properties - Element 482
 Properties - Element Context Menu 429
 Properties - Object 482
 Properties Menu Section 505
 Properties Note 411
 Property 250
 Property View 402
 Property Window 106, 402
 Pseudo-States 224
- Q -**
- Qualifiers 323
 Quick Add of Tagged Values 493
- R -**
- Read Only Interface 1062
 Realize 390
 Rearrange Package Order 965
 Receive 325
 Rectangle Notation 340, 432
 Recursion 354
 Reference Data 745
 Reference Data - Clients 747
 Reference Data - Import/Export 760
 Reference Data - Model Authors 746
 Reference Data - People 746
 Reference Data - Resources 748
 Reference Data - Roles 749
 Region 326, 328
 Region - Expansion 298
 Region - Interruptible Activity 310
 Regions 222
 Register Add-In 206
 Register Enterprise Architect 32
 Register Zicom Mentor 207
 Registration Key 34
 Relation Visibility 515
 Relationship Matrix 592
 Relationships Window 114
 Reload Current Diagram 162
 Remove a Defined Variable 318
 Remove Package from Version Control 715
 Remove Replication 740
 Removing Recent Models 39
 Rename a Project 673
 Rename Views 676
 Renaming Diagrams 404
 Renaming Packages 148
 Re-Order Messages 378
 Repair a Project 673
 Repeat Last Connector 57
 Repeat Last Element 57
 Replication 737
 Report - Glossary 814
 Report Sample - Glossary 816
 Report Templates 936
 Report View 146
 Reports - HTML 927, 957
 Reports - RTF 927
 Represents 391
 Requirement Properties 590
 Requirement Types 750
 Requirements 354
 Requirements Hierarchy 592
 Requirements Management 583
 Reset Options 853
 Resolving Conflicts 740
 Resource Management 769
 Resource Report 772

- Resource View 109
- Resource Window 108
- Resources 748, 768
- Responsibilities 485
- Responsibility Compartment 500
- Reverse Connector 511
- Reverse Engineer a Directory Structure 824
- Reverse Engineer ODBC data sources 920
- Reverse Engineer Source Code 821
- Review version control history 714
- Rich Text File 928
- Risk Management 771
- Risk Types 777
- Robustness Diagram 258
- Role Binding 391
- Role Tagged Values 525
- Roles 749, 773
- Root Node 96
- RTF Bookmarks 940
- RTF Custom Language Settings 955
- RTF Diagram Format 932
- RTF Documentation 927
- RTF Documents 927
- RTF Model Include 933
- RTF Options 933
- RTF Properties 930
- RTF Report Wizard 929
- RTF Reports 927
- RTF Selections 934
- RTF Templates 953
- Rule Window 115
- Running SQL Patches 656
- Runtime State 316
- Runtime State - Add an Instance Variable 316
- Runtime State - Delete an Instance Variable 318
- Run-time Variable - Defining 316

- S -**

- Sample Glossary Report 816
- Save a Package with XML 688
- Save All Modified Diagrams 162
- Save as Document 938
- Save as UML Pattern 577
- Save Diagram 57
- Save Image to File 406
- Save UML Pattern 577
- Scale Image to Page Size 153

- SCC 697, 698, 699, 700, 702, 706, 712
- Scenario Testing 786
- Scenario Types 753
- Scenario Window 117
- Scenarios 489
- Screen 355
- Searching a Project 166
- Security Group Permissions 728
- Security Policy 721
- Security Users 723
- Select a Data Source 922
- Select a Diagram 150
- Select Alternate Image 419
- Select Spelling Language 744
- Select Tables 923
- Self-Message 357
- Self-Message Hierarchy 231
- Send 326
- Sequence Communication Messages 377, 378
- Sequence Diagram 226
- Sequence Diagram Layout 229
- Sequence Diagrams 198
- Sequence Element Activation 230
- Sequence Elements 235
- Sequence Group 139
- Sequence Message 380
- Set Appearance Options 158
- Set Appearance Options - Visible Class Members 159
- Set as Model Default 399
- Set as User Default 57
- Set Connector Style 507
- Set Connector Visibility 515
- Set Diagram Page Size 152
- Set Element Parent 438
- Set Element Template Package 168
- Set Feature Visibility 410
- Set Group Permissions 728
- Set Instance State 318
- Set Message Source and Target 526
- Set MySQL Table Type 903
- Set Object State 318
- Set Parents and Interfaces 438
- Set Relation Visibility 515
- Set Table Owner 901
- Set Table Properties 899
- Set the Default Diagram 399
- Set the Main RTF Properties 930

- Set Up Single Permissions 725
 - Set Up User Groups 724
 - Set Up Version Control 698
 - Setting Collection Classes 843
 - Setting Default Tree Behavior 103
 - Setting Element Type 595
 - Setting Source and Target Package 595
 - Setting the Link Type and Direction 596
 - Setting Up an ODBC Driver 613
 - Setting Up Database Model Files 612
 - Setup a MySQL ODBC Driver 613
 - Setup a PostgreSQL ODBC Driver 616
 - Setup an Adaptive Server Anywhere ODBC Driver 619
 - Shared Keys 30
 - Sharing a Project 719
 - Sharing an Enterprise Architect Project 718
 - Show Element Usage 439
 - Show or Hide Attributes and Operations 155
 - Show or Hide Package Contents 148
 - Show Other References 440
 - Show/Hide Connectors 512
 - Show/Hide Labels 513
 - Single Permissions 725
 - Single User 841
 - Size Elements 448
 - Size Objects 448
 - Source and Target - Set for Message 526
 - Source Code Control 697
 - Source Code Viewer 111
 - Source Editor 111
 - Source Role 523
 - Space Evenly 59
 - Spell Checking 741
 - Spelling 742
 - Spelling Language 744
 - SQL 913
 - SQL Patches 656
 - SQL Server Data Repository 627
 - SQL Server Repository 642
 - Standard Colors 195
 - Standard Element Stereotypes 530, 757
 - Start Page 37, 39
 - Start Page Options 38
 - Starting Enterprise Architect 29
 - State 327
 - State Diagram 221
 - State Group 140
 - State Invariant 331, 332
 - State Lifeline 329
 - State Machine Diagram 221
 - State Machine Regions 222
 - State/Continuation 331
 - Status Bar 93
 - Status Types 751
 - Stereotype 554
 - Stereotype - Applying 371
 - Stereotypes 103, 527, 564
 - Stereotypes with Alternate Images 533
 - Structural Diagrams 241
 - Structure Group 136
 - Style 1020
 - Style Menu Section 506
 - SubActivity 333
 - Sub-Activity 271
 - SubMachine 334
 - Sub-Machine 327
 - Sub-states 328
 - Support 10
 - Supported Attributes in UML Profiles 575
 - Supported Tags in UML Profiles 571
 - Swimlanes 417
 - Synch 335
 - Synchronization 737
 - Synchronize Model and Code 854
 - Synchronize Package Tree 833
 - Synchronize UML Profile Tags and Constraints 566
 - Synchronizing Code 882
 - Synchronizing Existing Code Sections 883
 - Synchronizing Replicas 739
 - System Boundary 335
 - System Clients 774
 - System Tab 111
 - System Testing 783
 - System Users 723
- T -**
- Table 357
 - Tag Compartment 501
 - Tagged Value Macros 870
 - Tagged Values 118, 121, 128, 490, 757
 - Tagged Values in UML Profiles 564
 - Tags 121, 554
 - Target Types 952
 - Task Details 801

Task List 800
TCF Dialog 763
Team Development 717
Tear Off Menus 48
Technical Complexity Factors 763
Template - default element 168
Template Substitution Macros 863
Templates - Parameterized Classes 281
Terminate 336
Test Details 790
Test Details Dialog 779
Test Documentation 792
Test Scripts 791
Testing 184
Testing Report 952
Testing Support 778
Testing Types 755
Testing Workspace 779
Text 451
The Automation Interface 970
The Code Template Editor 879
The Generate HTML Report Dialog 958
The Project Interface (XML) 1062
The RTF Dialog 927
The Start Page 37
The Start Page Options 38
The UML Language 213
Tidy Line Angles 507
Timing Diagram 225
Timing Group 139
Timing Message 381
Timing Message Creation 382
Toggle Line Points 507
Toolbox 132
Toolbox Shortcut menu 134
Tools Window 63
Trace 392
Trademarks 5
Transition 392
Tree Style Hierarchy 517
Triangle 819
Tricks and Traps 1085
Type - Element Context Menu 432
Type Hierarchy 438
Type Specific Menu Section 505

- U -

UI Element 357
UML 756
UML Behavioral Diagrams 215
UML Connections 360
UML Diagrams 214, 255, 396
UML DTD 683
UML Elements 259, 263
UML Elements Toolbar 87
UML Language 213
UML Mentor 11, 207
UML Pattern - Add to Diagram 581
UML Patterns 577
UML Profile for XML 887
UML Profile Structure 574
UML Profiles 548
UML Stereotypes 527
UML Structural Diagrams 241
UML Tagged Values 757
UML Toolbox 132, 134, 135, 136, 137, 138, 139, 140, 141, 142, 144
Undo Last Action 160
Unit Testing 780
Update links in Word 948
Update Package Contents 833
Update Package Status 817
Upgrade an Existing License 33
Upgrade Current License Key 82
Upgrade Replicas 653
Upgrade SCC for Version 4.5 716
Upgrade Wizard 652
Upgrading Models 652
Upgrading Projects 652
Upgrading Replicas 740
Upsizing Models 661
Upsizing to MySQL 662
Upsizing to Oracle 666
Upsizing to PostgreSQL 668
Upsizing to SQL Server 664
Upsizing to SQL Server Desktop Engine (MSDE) 672
Upsizing to Sybase Adaptive Server Anywhere (ASA) 670
Use 393
Use a Pattern 581
Use Case 337

Use Case Arrowhead 520
 Use Case Diagram 219
 Use Case Extension Points 338
 Use Case Group 135
 Use Case Metrics 766
 User Dictionaries 744
 User Groups 723
 User Security 720
 User Security Groups 724
 User Settings 841
 Using Classifiers 497
 Using Enterprise Architect 37
 Using MS Word 939
 Using Object Flows in Activity Diagrams 387
 Using Profiles 560
 Using Rectangle Notation 340
 Using the Automation Interface 971
 Using the Spell Checker 742
 Using UML 395
 Using Version Control 699

- V -

Value Lifeline 340
 Variable Definitions 877
 Variable References 878
 VB - Set up for Automation Interface 972
 VB.Net 826
 VB.Net Options 852
 Version Controlled Package 710
 Version Control 697
 Version Control - Get Shared File from Repository Dialog 687
 Version Control Configuration 712
 Version Control History SCC 703
 Version Control Menu 700
 Version Control Options 697
 Version Control Options CVS 709
 Version Control Options Dialog 700
 Version Control Options SCC 706
 Version Control Other users packages 716
 Version Control Overview 698
 Version Control Providers Dialog 702
 Version Control Reference 699
 Version Control SCC Show interface 706
 Version Control Setup Menu 700
 View Last and Next Diagram 151
 View Menu 51

View Options 145
 Views 675
 Virtual Documents 963
 Virtual Documents - Adding Packages 964
 Virtual Documents - Delete a Package 966
 Virtual Documents - Generate the Document 967
 Virtual Documents - Rearranging Package Order 965
 Virtual Report - Document Object 964
 Visible Class Members 159
 Visual Basic 826
 Visual Basic - Set up for Automation Interface 972
 Visual Basic Options 851
 Visual Layouts 205
 Visual Studio.NET 855
 Visual Styles 206

- W -

Web Documentation 960
 Web Stereotypes 359
 Web Templates 960
 What is UML 213
 Why Compare Models? 658
 Width 448
 Word - Special Considerations 943
 Worker 360
 Working with MDG Technologies 543
 Working with UML Connections 504
 Working with UML Elements 425
 Working with UML Profiles 551
 Workspace Toolbars 83
 Workspace Views 91

- X -

XMI Import and Export 679
 XML schema generation 884
 Example 885
 Getting Started 884
 Steps to Generate XSD 885
 UML Profile for XML 887
 XML Specifications 202

- Y -

Your Comments and Feedback are Welcome 11

- Z -

Z Order Elements 405
Zicom Mentor 11
Zicom UML Mentor 207
Zoom a Diagram View 151

Enterprise Architect User Guide
www.sparxsystems.com.au